Stream Learning with Entity-Centric Models and Active Feature Acquisition

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von M. Sc. Christian Beyer

geb. am 10.08.1987 in Haldensleben

Gutachterinnen/Gutachter

Prof. Dr. Myra Spiliopoulou
Prof. Dr. Allan Tucker
Prof. Dr. Sławomir Nowaczyk

Magdeburg, den 21.11.2024

Abstract

Data stream mining addresses environments where data arrives continuously, often at high volumes and velocities, and requires models that adapt incrementally to changes in the data. In many cases, data stream instances are linked to specific entities, such as product reviews (instances) being linked to a particular product (entity). However, current practices generally utilize a single model per stream, disregarding the relationship between entities and their instances. Another common assumption is that arriving instances are feature-complete or that missing features must be imputed due to their unavailability.

This thesis is organized into two parts. The first part introduces the concept of entity-centric learning on data streams, where the relationship between entities and their instances is explicitly modeled in the stream's data space. Entity-centric learning involves creating individual prediction models for each entity, allowing model learning and adaptation to account for each entity's unique characteristics. This approach contrasts with traditional methods that apply a single model across the entire stream. The analysis demonstrates that entity-centric learning can improve prediction quality in data streams.

Initial experiments focused on comparing an entity-ignorant (global) model, which accessed features but pooled data from all entities, with basic entity-centric models that relied solely on an entity's labels, such as a moving average. Results highlighted that, for some entities, even a simple entity-centric model outperformed a more complex entity-ignorant one. However, entities with fewer instances showed better results with the entity-ignorant model.

A hybrid ensemble approach combining entity-centric and entity-ignorant models was proposed to address this. This ensemble significantly improved prediction quality across most entities. However, maintaining separate models for each entity posed a challenge for memory management. To resolve this, a memory management system was introduced, differentiating between active and inactive entities. Models for active entities were kept in primary memory, while those for inactive entities were moved to secondary storage. This makes entity-centric learning feasible on data streams with a large number of entities.

The second part of the thesis shifts focus from the data space to the feature space, tackling the problem of missing features. One of the first studies on active feature acquisition (AFA) for data streams is presented, exploring scenarios where missing features can be acquired at a cost under budget constraints.

The initial approach assumed equal feature costs and allowed the acquisition of one missing feature per instance. Using a stream feature selection metric to assess feature importance, it was shown that acquiring the most valuable feature outperformed random acquisition, usually a competitive baseline in the field of active learning.

Subsequent work extended this method to handle varying acquisition set sizes and differing feature costs. Experiments across multiple cost distributions and levels of missingness demonstrated that the proposed methods consistently outperformed a random baseline. The experiments also highlighted that a feature importance metric that considers each feature independently can lead to superfluous acquisitions in the

case of feature correlations, which wastes the available budget.

This was addressed in the final study, which introduces a two-stage decision process for budget optimization, where imputation is considered before feature acquisition. A specialized imputation method, the feature-pair-imputer, was developed to predict missing features based on available ones, tracking imputation performance. This approach allowed for informed decisions on whether to impute or acquire missing features, achieving budget savings without significantly compromising prediction accuracy compared to an AFA-only strategy.

In conclusion, this thesis presents novel methods for both entity-centric learning on data streams and active feature acquisition. Various experimental frameworks were developed to support these investigations, including synthetic dataset generators for exploring aspects such as concept drift, complex label dependencies, and feature correlations. The results demonstrate that entity-centric models can enhance prediction quality with minimal computational cost, while active feature acquisition enables cost-effective handling of missing features in data streams.

Zusammenfassung

Das Lernen auf Datenströmen befasst sich mit Umgebungen, in denen Daten kontinuierlich, oft in großen Mengen und mit hoher Geschwindigkeit eintreffen, und erfordert Modelle, die sich inkrementell an Veränderungen in den Daten anpassen. In vielen Fällen sind Datenstrominstanzen mit bestimmten Entitäten verknüpft, wie z.B. Produktbewertungen (Instanzen), die einem bestimmten Produkt (Entität) zugeordnet sind. Die derzeitigen Praktiken verwenden jedoch in der Regel ein einzelnes Modell pro Datenstrom und ignorieren dabei die Beziehung zwischen Entitäten und ihren Instanzen. Eine weitere häufige Annahme ist, dass die eintreffenden Instanzen keine fehlenden Attribute haben oder dass fehlende Attribute aufgrund ihrer Nichtverfügbarkeit imputiert werden müssen.

Diese Dissertation ist in zwei Teile gegliedert. Der erste Teil führt das Konzept des entitätszentrierten Lernens auf Datenströmen ein, bei dem die Beziehung zwischen Entitäten und ihren Instanzen explizit im Datenraum des Stroms modelliert wird. Das entitätszentrierte Lernen umfasst die Erstellung individueller Vorhersagemodelle für jede Entität, was eine Anpassung und das Lernen der Modelle an die spezifischen Charakteristiken jeder Entität ermöglicht. Dieser Ansatz steht im Gegensatz zu traditionellen Methoden, die ein einziges Modell über den gesamten Strom anwenden. Die Analyse zeigt, dass entitätszentriertes Lernen die Vorhersagequalität in Datenströmen verbessern kann.

Die ersten Experimente konzentrierten sich auf den Vergleich eines entitätsignoranten (globalen) Modells, das auf Attribute zugriff, aber die Daten aller Entitäten verarbeitete, mit naiven entitätszentrierten Modellen, die sich ausschließlich auf die Labels einer Entität stützten, wie z.B. einen gleitenden Durchschnitt. Die Ergebnisse zeigten, dass für einige Entitäten selbst ein einfaches entitätszentriertes Modell ein komplexeres entitätsignorantes Modell übertraf. Allerdings erzielten Entitäten mit weniger Instanzen bessere Ergebnisse mit dem entitätsignoranten Modell.

Um dies zu lösen, wurde ein hybrider Ensemble-Ansatz vorgeschlagen, der entitätszentrierte und entitätsignorante Modelle kombiniert. Dieses Ensemble verbesserte die Vorhersagequalität für die meisten Entitäten signifikant. Die Aufrechterhaltung separater Modelle für jede Entität stellte jedoch eine Herausforderung für das Speichermanagement dar.

Um diese Herausforderung zu adressieren, wurde ein Speichermanagementsystem eingeführt, das zwischen aktiven und inaktiven Entitäten unterscheidet. Modelle für aktive Entitäten wurden im Primärspeicher gehalten, während Modelle für inaktive Entitäten in den Sekundärspeicher verschoben wurden. Dies macht entitätszentriertes Lernen auf vielen Datenströmen praktikabel.

Der zweite Teil der Dissertation verlagert den Fokus vom Datenraum auf den Attributsraum und befasst sich mit dem Problem fehlender Attribute. Es wird eine der ersten Studien zur aktiven Attributsakquisition für Datenströme vorgestellt, die Szenarien untersucht, in denen fehlende Attribute zu Kosten unter Budgetbeschränkungen erworben werden können.

Der anfängliche Ansatz ging von gleichen Attributkosten aus und ermöglichte den Erwerb eines fehlenden Attributs pro Instanz. Mithilfe einer Metrik zur Attribu-

tauswahl im Datenstrom, die die Wichtigkeit der Attribute bewertete, wurde gezeigt, dass der Erwerb des wertvollsten Attributs eine zufällige Akquisition übertraf.

Darauf aufbauend wurde diese Methode erweitert, um unterschiedliche Akquisitionsgrößen und variierende Attributkosten zu berücksichtigen. Experimente über verschiedene Kostenverteilungen und Ausmaße des Fehlens zeigten, dass die vorgeschlagenen Methoden eine zufällige Baseline konstant übertrafen. Die Experimente zeigten auch, dass eine Metrik zur Attributwichtigkeit, die jedes Attribut unabhängig betrachtet, zu überflüssigen Akquisitionen führen kann, wenn Attributkorrelationen vorliegen, was das verfügbare Budget verschwendet.

Dies wurde in der letzten Untersuchung aufgegriffen, die einen zweistufigen Entscheidungsprozess zur Budgetoptimierung einführt, bei dem die Imputation vor der Attributakquisition in Betracht gezogen wird. Eine spezialisierte Imputationsmethode, der "Feature-Pair-Imputer", wurde entwickelt, um fehlende Attribute basierend auf verfügbaren vorherzusagen und die Imputationsleistung zu verfolgen. Dieser Ansatz ermöglichte fundierte Entscheidungen darüber, ob fehlende Attribute imputiert oder akquiriert werden sollten, wodurch Budgeteinsparungen erzielt wurden, ohne die Vorhersagegenauigkeit im Vergleich zu einer rein auf aktiver Attributsakquisition basierenden Strategie signifikant zu beeinträchtigen.

Diese Dissertation präsentiert Methoden sowohl für das entitätszentrierte Lernen auf Datenströmen als auch für die aktive Attributsakquisition. Verschiedene Software wurden entwickelt, um diese Untersuchungen zu unterstützen, einschließlich synthetischer Datengeneratoren zur Erkundung von Aspekten wie Concept Drift, komplexen Label-Abhängigkeiten und Attributskorrelationen. Die Ergebnisse zeigen, dass entitätszentrierte Modelle die Vorhersagequalität bei minimalen Rechenkosten verbessern können, während die aktive Attributsakquisition eine kosteneffiziente Handhabung fehlender Attribute in Datenströmen ermöglicht.

Contents

List of Figures

List of Tables

List of Acronyms

AED Average Euclidean Distance

AFA Active Feature Acquisition

AL Active Learning

IG Information Gain

IPF Incremental Percentile Filter

MAR Missing At Random

MCAR Missing Completely At Random

MNAR Missing Not At Random

RA Random Acquisition

RQ Research Question

SBM Simple Budget Manager

SU Symmetric Uncertainty

1. Introduction

The advent of smartphones and Web 2.0 led to a dramatic increase in user-generated data. People engage with each other on social networks, streaming platforms, and various apps that track or monitor user behavior, such as weight and fitness trackers. Therefore, many businesses are confronted with a continuous influx of new data at a high speed, which changes as user behavior adapts to current trends and events. High volume, high velocity, and high volatility are the characteristics of data streams. In contrast to classical machine learning methods designed for static data, analyzing data streams poses special challenges that need to be considered. In most streaming scenarios it is unfeasible to retrain a machine learning model every time a new instance arrives on the stream, but we need models that can be updated efficiently in an incremental manner. Furthermore, these models need to be able to forget outdated information in order to present up-to-date predictions to the users of these models. The users, also called entities, are often diverse in their behavior, likes, dislikes, and needs. This diversity was mostly ignored in the stream mining community where a one-model-fits-all approach is most common. The first part of this thesis explores an alternative approach, where global/entity-ignorant models, which see the data from all entities, are supplemented with entity-centric models that were solely trained on an entity's data to improve prediction quality; see Figure 1.1. Storing an ever-growing number of entity-centric models can put huge strains on our memory requirements, which is a challenge that needs to be addressed, as well as the question of how entity-centric models can be incorporated with entity-ignorant ones.

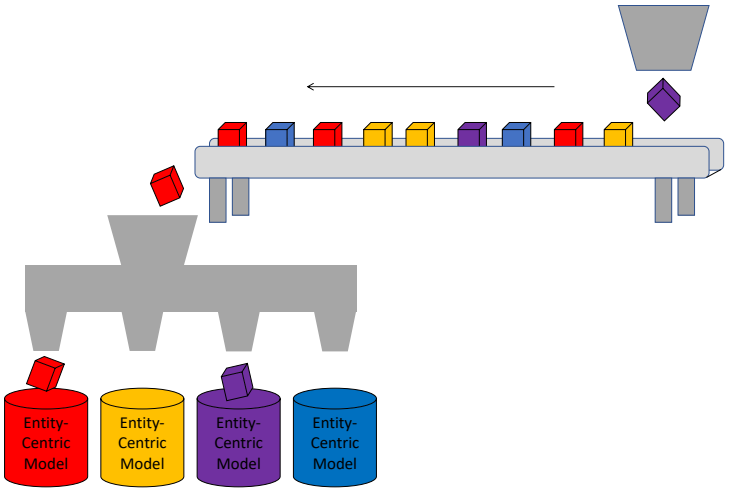

Figure 1.1.: Instances depicted as packages arrive on a data stream, and the color codes the entity they belong to. Each entity has a dedicated model trained on the instances belonging to that entity, called the entity's substream. In contrast, an entity-ignorant model would be trained on all instances.

Another common assumption is that all the instances used to train our machine learning models are feature-complete, which is rarely true in real-world applications. People skip questions in a questionnaire, sensors feeding a data stream can break, and doctors do not send their patients to all available diagnostic tests but only a select few. If we know that a particular feature is crucial to making a reliable prediction and is missing, we often could actively query for that feature. For instance, a user could be prompted to fill out a missing question with a note explaining that it is crucial in their assessment. This process of active feature acquisition often invokes a cost, as users get tired and hesitant or just the financial burden of demanding an additional diagnostic test. Therefore, the second part of this thesis will focus on cost-sensitive methods that can alleviate the problem of missing features in a data stream and facilitate data stream mining in real-world scenarios.

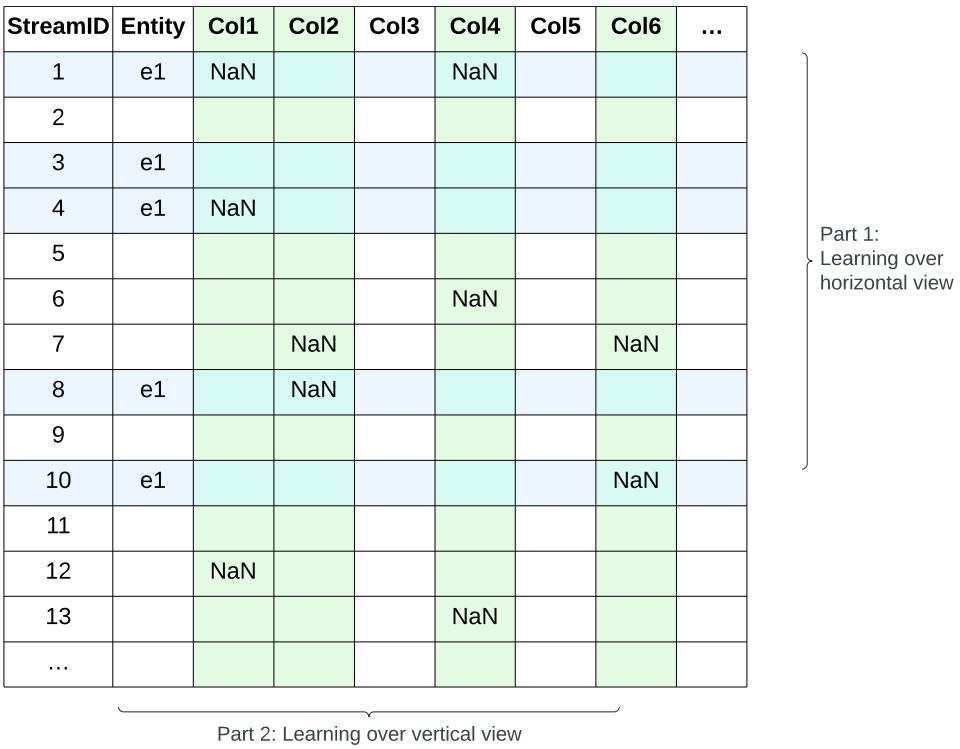

Figure 1.2.: In the horizontal view, we can see that instances belong to particular entities, which we can exploit to enrich our models with this information. The vertical view concerns the features of the instances and what role they play in the data mining task. This information can be exploited when we can acquire missing features (NaN) for a cost.

Each part of the thesis deals with data streams, but the first part focuses on the horizontal view and the second on the vertical view of the data; see Figure 1.2. Both parts aim to improve data quality to enhance stream mining methods. This follows a current trend in the data mining sphere, where famous figures like Andrew Ng urge data scientists to be 'More Data-Centric And Less Model-Centric'[1]. In the next section, the research questions that are addressed in this thesis are introduced.

[1]Article on data quality, visited 20.08.2024, `https://analyticsindiamag.com/intellectual-ai-discussions/big-data-to-good-data-andrew-ng-urges-ml-community-to-be-more-data-centric-and-less-model-centric/`

1.1. Research Questions

The thesis is divided into two main parts. Part I addresses the entity-instance relationship in the data space of a data stream, covered by research questions 1 and 2. Part II focuses on the feature space, specifically exploring how Active Feature Acquisition (AFA) can manage missing feature values, as examined in research questions 3 and 4. Each Research Question (RQ) will be highlighted in more detail before moving to the main contributions of this thesis in the next section.

RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model? Entity-centric models typically receive far less training data than entity-ignorant models, as the latter are trained on all instances within the data stream, while the former are limited to the substream of instances specific to a particular entity. A smaller amount of training data is often associated with poorer performance during testing, so the performance of entity-centric models is generally expected to be inferior to that of entity-ignorant models. RQ1 explores this assumption by examining the percentage of entities that benefit from entity-centric models, the impact of the number of available training instances on their performance, and the classification performance of methods that combine entity-centric and entity-ignorant predictions.

RQ2: How can the memory footprint of entity-centric models be reduced? Creating a prediction model for each entity in a data stream can result in prohibitively high memory demands, which must be mitigated. RQ2 focuses on experiments aimed at reducing the memory footprint of entity-centric models while preserving potential performance gains

RQ3: How can Active Feature Acquisition be realized in a data stream setting? Active Learning (AL) on data streams typically assumes that labels, rather than features, are missing, that all missing labels have the same acquisition cost, and that one label can be obtained per instance. However, the presence of missing features introduces new challenges, as multiple features can be missing for a given instance, and the cost of acquiring each feature may vary. Additionally, unlike static AFA, decisions regarding which features to acquire must be made immediately as instances arrive. RQ3 investigates how AFA can be effectively implemented in a data stream setting

RQ4: How can varying feature costs be considered during AFA on streams? RQ4 focuses on the integration of feature costs in AFA for data streams and its impact on budgeting. This includes decisions related to budget allocation and strategies for optimizing budget usage across the stream. Additionally, it explores how AFA and imputation can be intelligently combined to minimize costs further.

1.2. Summary of Scientific Contributions

Part I focuses on the data space and covers entity-centric learning, where a prediction model is created for each entity appearing in a data stream. The goal is to improve prediction quality with models tailored to an entity's characteristics and behavior.

A good example of entity-centric learning is how LLMs like ChatGPT 4o are increasingly used as personal assistants, which can now recall specific memories of interactions with specific users (entities) - 'We're testing memory with ChatGPT. Remembering things you discuss across all chats saves you from having to repeat information and makes future conversations more helpful...You've explained that you prefer meeting notes to have headlines, bullets and action items summarized at the bottom. ChatGPT remembers this and recaps meetings this way.'[2].

This work addresses entity-centric learning in a data stream setting, which has its particular challenges, and the main contributions are:

1. Introduction of an entity-entric, ensemble-based learning approach that combines entity-centric and entity-ignorant models' predictions, improving prediction quality.
2. Introduction of methods for managing the growing memory requirements resulting from creating new models as new entities arrive on the data stream.
3. Showing that even very simple entity-centric models that solely rely on labels can improve prediction quality while having a very low impact on memory requirements.

Part II focuses on the feature space. It addresses the issue of missing features on a data stream, as most machine learning methods require feature-complete instances to make predictions. In data stream settings, it is common that missing features are imputed (guessed) using various models that either use other features to predict missing ones or rely on the distribution of a feature and then sample from that distribution.

This work investigates scenarios where missing features can be acquired under budget constraints, which can be a preferable alternative in many scenarios. For example, if the blood type of an incoming patient is crucial to the diagnosis, it is better to run a costly lab test than to use imputation, but depending on the symptoms and other factors, it might be unnecessary to run the blood test on every incoming patient. The main contributions in this part are:

1. Introduction of methods that realize AFA under equal feature costs when only a single missing feature can be acquired.
2. Development of methods that allow for bigger acquisition sets and can handle varying feature costs while aiming for optimal usage of the available budget.
3. Introduction of a hybrid approach that relies on strategic imputation to further reduce costs.

The next section will briefly outline the thesis's structure before moving to the underpinnings, which are necessary for both parts of the thesis.

[2] `https://openai.com/index/memory-and-new-controls-for-chatgpt/`, visited 14th September 2024

1.3. Outline of the Thesis

The structure of the thesis is as follows. First, the foundational concepts of data stream mining, such as concept drift and common evaluation metrics, are introduced in chapter 2. This sets the stage for the first main part of the thesis, which focuses on entity-centric learning (chapter 3).

Chapter 4 begins by presenting the initial work using simple, text-unaware entity-centric models to predict the polarity of product reviews. This approach is extended in chapter 5, where text-aware entity-centric models are combined with an entity-ignorant model in various ensemble configurations. Chapter 6 then addresses the challenge of reducing memory requirements in entity-centric learning. The first part concludes with additional reflections in chapter 7 and a general conclusion, including limitations, in chapter 8.

The second main part of the thesis (chapter 9) focuses on Active Feature Acquisition (AFA) on data streams. Chapter 10 introduces AFA on data streams, allowing for the acquisition of one feature per instance, assuming equal feature costs. This work is expanded in chapter 11, which presents methods for handling acquisition sets of varying sizes and budget management strategies for handling variable feature costs. Chapter 12 presents a hybrid approach that integrates AFA with strategic imputation to reduce budget expenditures. Additional reflections follow in chapter 13, and the second part concludes with limitations and future work in chapter 14.

The thesis ends with chapter 15, which summarizes the results, synthesizes the findings from both parts of the thesis, and explores more ambitious directions for future research.

2. Stream Mining Underpinnings

This chapter introduces general concepts of data streams and data stream mining. We first discuss what characteristics are typical for a data stream in contrast to static data and will introduce typical stream evaluation schemes and metrics.

2.1. Data Streams

The introduction mentioned the three 'Vs' that characterize data streams: Velocity, Volume, and Variety, but the main difference to static data is that new instances constantly arrive and need to be incorporated into our models [6]. The three 'Vs' describe additional characteristics that emphasize why we often cannot simply apply solutions developed for handling static data iteratively.

Velocity: In a data stream, instances typically arrive with a high velocity and must be processed by our stream learning models immediately [45]. Consequently, our models must be able to process incoming instances faster than the stream's velocity so that we do not build a stack of unprocessed instances. Furthermore, the stream mining models must support fast updates to incorporate the latest information. Most static models can fulfill the first requirement but struggle to meet the second, as each new instance would require retraining the whole model, which can be incredibly demanding concerning time and computational resources. The growing volume of data aggravates this challenge.

Volume: Stream mining usually assumes an infinite number of instances to arrive, which means we cannot store all incoming data in primary memory, and we also cannot consider all available data for training or re-training our models, as the temporal and computational demands would increase with each incoming instance. Therefore, stream mining methods must be able to work under memory constraints [6]. Due to the potentially high velocity and volume of the arriving data, most stream mining algorithms work incrementally, which means they can be trained one instance at a time. This is sometimes called *online* training. For example, very fast decision trees [22] can be trained incrementally in contrast to the conventional CART algorithm [14].

Variety and Concept Drift: The third 'V' addressed the challenge that our target variable's distribution and the distributions of our features might change over time. These changes in distribution are called concept drift [2], see Figure 2.1. For example, music genres might rise and fall in popularity on a streaming platform, the average complexity of song lyrics changes over time [64], and the average amount of money a user spends might change according to the economic situation. These changes must be detected and then incorporated into our models. Consequently, our models should be able to forget outdated knowledge. For example, in [51], the authors present a forgetting mechanism for the k-nearest-neighbor classifier.

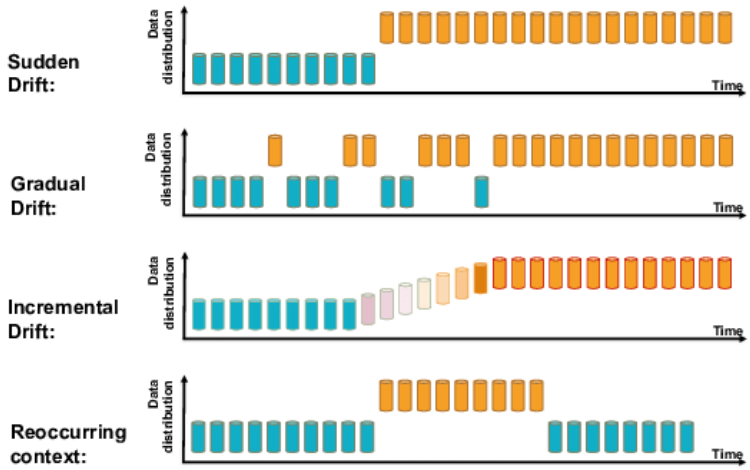

Figure 2.1.: Depiction of different kinds of concept drift. Reproduced with permission from [84] ©CC0 1.0 Universal.

Additional challenges for stream mining that have been discussed in [6] are potential **label delay**, which impacts the training of our models as we have to wait for an update till the label arrives or work with surrogate labels in the meantime. Furthermore, **high-dimensional data** can bring extra challenges as it typically requires more computational resources and processing time. This can impact the speed of our model regarding incremental updates and degrade performance due to the curse of dimensionality. Lastly, **imbalanced classes** can be problematic as models designed to generalize might ignore the minority class, which can be exacerbated by employing forgetting techniques.

2.2. Evaluation Schemes and Performance Metrics for Supervised Stream Mining

The dynamic nature of data streams and their potential infinite length pose a challenge when evaluating a model's performance. Models and their performances change across the data stream, which makes splitting a stream into training and test streams a sub-optimal solution [29]. Instead the most common way to do evaluation is to use a combination of prequential evaluation and a rolling window of the chosen performance metric [29, 88].

Prequential Evaluation means that each incoming instance is first used for testing and subsequently for the training of our model. This ensures that all instances are used for testing and that concept drift can be picked up by our model as each instance is also used for training. With all the predictions available one can calculate the overall performance across the whole stream but because of the assumed dynamic nature it is more common to move a window over the stream and to calculate the performance at a certain time point, given the instances in the current window. The window can be rolling or realized as non-overlapping chunks of data. The size of the window is typically hand-chosen but can also be calculated and doesn't have to be fixed [88]. When we use such an evaluation method we often see low performance scores at the beginning of the stream as few instances have been available for training yet, which is called the cold start problem.

Table 2.1.: Confusion matrix of a binary classifier.

Confusion Matrix	**predicted positive**	**predicted negative**
real positive	true positive (TP)	false negative (FN)
real negative	false positive (FP)	true negative (TN)

Evaluation Metrics: In this work, we use accuracy, balanced accuracy, κ and κ_+ for classification tasks. Given a confusion matrix over a window or the whole data stream, see Table 2.1, accuracy is defined as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Accuracy has the issue that it is not suited for imbalanced classes. If the classifier is good at predicting the majority class, it can vastly overestimate the classification performance.

For example, if we have 100 people and we want to predict their cancer risk, and only 10 people in our dataset have cancer (90 are cancer-free), then a simple majority classifier that labels every instance as not having cancer would reach an accuracy of 90% (90/100).

Balanced accuracy aims to remedy this issue by giving equal weight to all classes.

$$sensitivity = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

$$balancedAccuracy = \frac{sensitivity + specificity}{2} \tag{2.2}$$

Another issue with using accuracy on data streams is that it ignores temporal patterns. This is why Bifet et al. [13] propose an adjustment to the popular Cohen's Kappa performance metric, which adjusts the performance estimates by comparing a model's performance against a classifier that propagates the last observed label (no-change classifier). This ensures that the performance is not overestimated, as it will be 0 or negative if our model performs similarly to or worse than the no-change classifier. If the model's prequential accuracy is p_0 and a no-change classifier would reach a prequential accuracy of $p_e^{'}$ then κ^+ is defined as:

$$\kappa^+ = \frac{p_0 - p_e^{'}}{1 - p_e^{'}} \tag{2.3}$$

The κ^+ statistic is more reliable when facing imbalanced classes and simple temporal patterns whereas κ can be used when such temporal patterns are absent.

For numerical or ordinal labels, we employ RMSE as a metric that penalizes predictions $\hat{y}$, which are further off from the true label y.

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{2.4}$$

The basic concepts introduced here provide a sufficient overview to follow our proposed solutions to learn entity-centric models on streams in Part I (chapters: 4, 5 and 6), and acquiring the best features to learn from over a data stream with missing values in Part II (chapters: 10, 11 and 12).

Part I.

Dealing with the Data Space in Stream Mining

3. Motivation & Background - Entity-Centric Learning on Data Streams

The first part of this thesis covers the work on the data space of data streams, particularly when data points/instances are tied to specific entities. We investigate if and how the knowledge of an entity-instance relationship can be leveraged to improve prediction quality on a data stream. In many scenarios, stream mining algorithms are employed on data streams where the incoming instances belong to particular entities. For example, online posts belong to a specific social media platform user, product reviews belong to a specific product, and incoming temperature measurements belong to a specific weather station. Knowing which entity an instance belongs to can inform a stream learner and give particular context, especially in the case of opinion stream mining, which we consider in our work. For example, 'thin' may be positive when referring to a cell phone but negative when referring to a winter coat.

Different entities and their instances often show varying label and feature distributions. One common approach to dealing with different distributions on static data is to apply clustering to the instances first so that similar instances are grouped together. Afterwards, classifiers can be built for each individual cluster, forming a Multi-Classifier System (MSC), which can apply different knowledge to a test instance depending on which cluster it belongs to.

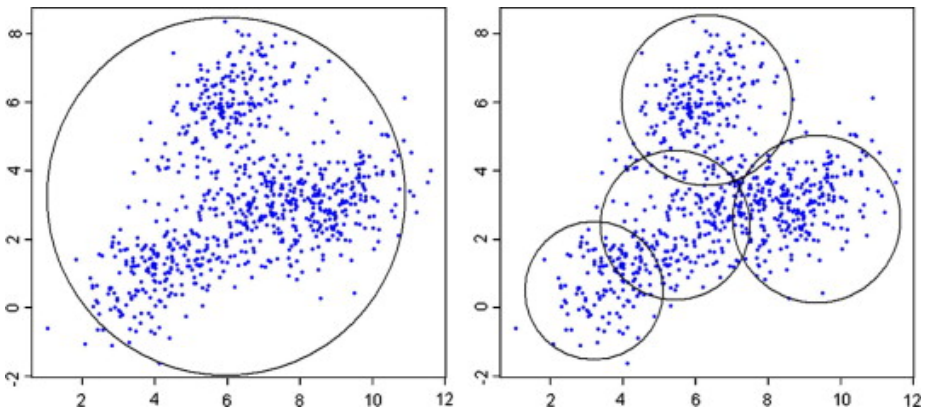

Figure 3.1.: The figure depicts the decision boundary of a one-class-classifier that sees all instances of a class on the **left** and a group of four classifiers applied on the same data after clustering to the **right**. It shows that the white space within the decision boundary, where errors can occur, becomes smaller when the cluster-based approach is used. Reproduced with permission from [43] ©Science Direct 2020

In [43], Krawczyk et al. propose a MSC in which they first filter all training instances by class, then they cluster all the instances of each class into multiple groups, next they train a one-class-classifier on each individual group and lastly

they fuse the predictions of their classifiers. Figure 3.1 shows that the cluster-based approach captures the distribution of training instances better than a model that sees all the instances of a class. The authors report that the cluster-based approach outperformed a single-model-multiclass approach in 13 out of 20 experiments and was superior to an ensemble of one-class-classifiers without clustering in 16 out of 20 experiments.

Inspired by the findings in [43] and similar ones [53, 75], we want to investigate the potential of building one model per entity in a data stream setting so that an entity-centric model is only trained on the instances belonging to the substream of a particular entity; see Figure 3.2. Clustering entities and building models for groups of similar entities is out of the scope of this work. We will start with the most extreme case, which employs one model per entity.

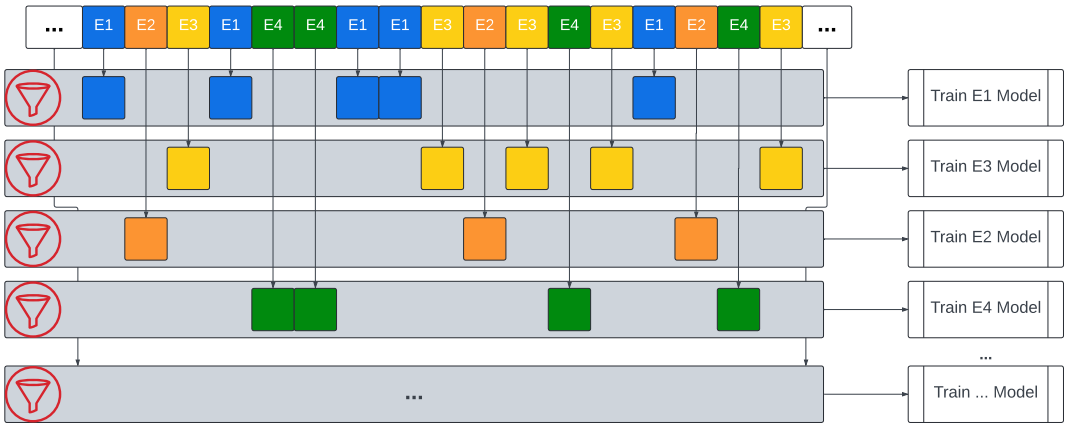

Figure 3.2.: Each instance on the data stream at the top belongs to a certain entity signified by its color. The instances of an entity build a substream, which is the sole input to the entity-centric models seen on the right.

The association between an opinionated document and its “target entity" is being studied intensively. For example, in [38], Jakob and Gurevych elaborate on the potential of Conditional Random Fields for the identification of the entity to which an opinion refers. Qiu, et al. combine the task of target extraction with the exploitation of the words in an opinionated document to assess polarity [67]. Zhang and Liu elaborate on the aspects of an entity, which are conveyed in opinionated documents, and distinguish further between opinions on an entity and comparative opinions, which involve more than one entities [86]. In [19], Deng and Wiebe use probabilistic modeling to predict the polarity of an opinionated document and of the target of this opinion simultaneously, whereas Fudholi et al. [26] propose the use of a BERT-based language model for the highlighting of named entities in the form of tourist attractions from search results.

While much research focuses on recognizing and extracting target entities in the form of Named Entity Recognition (NER) [47], our research concerns scenarios where the entity-instance relationship is known in advance. For example, the product ID contained in the metadata of a product review identifies the product (entity) it refers to, so there is no need to extract the target entity from the review text.

Furthermore, this thesis focuses on entity-centric learning on data streams, which brings unique challenges.

3.1. Challenges for Entity-Centric Learning

Deploying one model per entity can be quite challenging because, in many scenarios, the substreams for each entity are orders of magnitude smaller than the stream covering instances from all entities. For example, while there are millions of product reviews on Amazon, most individual products receive only very few reviews. Consequently, the amount of training data for each entity is much smaller as well, which will affect the quality of the resulting entity-centric models. Furthermore, outliers have a bigger impact on entity-centric models, as each sample will have a higher weight if the number of samples is reduced, i.e., an outlier among 10 instances is worse than an outlier among 1000 instances. This can then lead to entity-centric overfitting. Another challenge is the growing memory footprint of storing entity-centric models if we work in a scenario with many entities or potentially unlimited entities. This effect also strongly depends on the chosen machine learning models which will be applied on an entity level, while having one decision tree per entity might be feasible, it might not be computationally or monetarily feasible to train and keep a deep learning model for each entity.

Considering the above-mentioned challenges of building robust entity-centric models, we early on considered merging entity-centric models, which have a local perspective, with an entity-ignorant model, which has a global perspective. This brings the challenge of combining entity-centric and entity-ignorant predictions.

This work will address the challenges of incorporating entity-centric models on a data stream and managing their memory footprint. This concerns the following research questions posed in section 1.1:

- RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model?
- RQ2: How can the memory footprint of entity-centric models be reduced?

Chapters 4 and 5 present proposed solutions for RQ1, while chapter 6 contains solutions to RQ2. In chapter 4, entity-centric models are introduced that operate solely on the substream of labels belonging to an entity, without accessing the features of incoming instances. Chapter 5 relaxes this assumption, exploring how an ensemble of entity-ignorant and entity-centric models improves performance compared to a single global model applied to the entire dataset. Chapter 6 addresses the issue of the prohibitive memory demands associated with maintaining numerous entity-level models. It also combines the approaches from chapters 4 and 5 by introducing an ensemble that includes an entity-ignorant model with access to features, alongside a simple entity-centric model that relies solely on labels.

Part I concludes with additional reflections on entity-centric learning in chapter 7, followed by a final summary in chapter 8.

4. Entity-Centric Learning without Features

This part covers the work published in [7], additional unpublished material, and addresses:

RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model?

The initial intuition behind entity-centric learning was that it would be most beneficial for entities that significantly deviate from the general population and, as a result, cannot be effectively predicted by an entity-ignorant model. It was expected that such entities would exhibit different label patterns compared to the majority of instances in the data stream, which motivated the focus on the substream of labels associated with each entity.

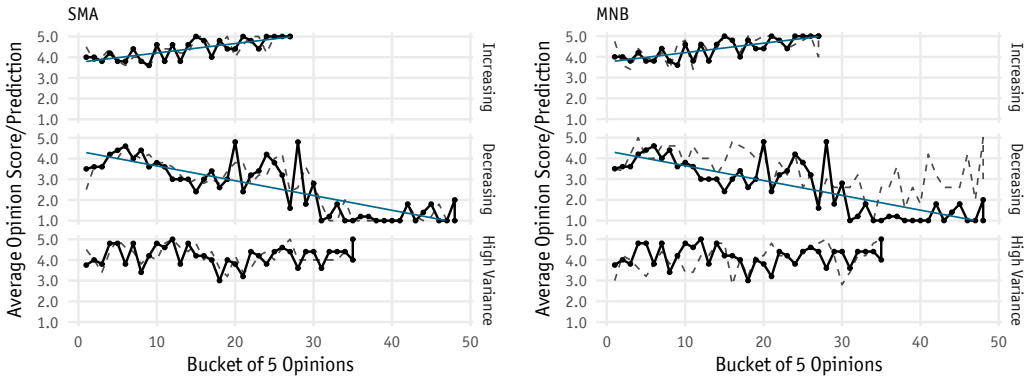

Figure 4.1.: Development of reviews scores from example Amazon products. The solid line represents the bucket average of review scores; dashed lines show predictions of entity-centric, text-unaware Simple Moving Average (SMA, left) and entity-ignorant, text-aware Multinomial Naive Bayes (MNB. right) using a window length of 5. Especially for the product with decreasing scores, entity-centric SMA performs better than the MNB.

Figure 4.1 illustrates that entity-centric, text-ignorant predictors, which rely solely on past labels, can accurately approximate the arriving labels for certain entities. In fact, these models can even outperform text-aware approaches in some cases. For instance, the product displayed in the middle of the figure demonstrates that the simple-moving-average (SMA) model over the substream of labels exhibits a smaller error than a Multinomial Naive Bayes classifier (MNB) that had access to the full review text. This chapter specifically examines how well the labels of opinionated texts referring to a particular entity can be predicted when only the substream of labels associated with that entity is available and no access to textual features is provided. The Amazon review dataset is used for this investigation, treating each

product as a distinct entity that receives a substream of opinionated reviews, with the star rating serving as the target label for prediction.

The evaluation compares simple, text-unaware entity-centric models, which only use labels from the substream of a particular product, against a text-aware, entity-ignorant model that is trained and tested on all arriving reviews, regardless of the product or entity to which they belong. The sections 4.6 and 4.7 are based on their counterparts in [7].

4.1. Related Work

The stated objective is to predict the next label of an opinion associated with a specific entity, without utilizing the opinion text itself. This task intersects with several research areas, including polarity learning, target entity analysis, and stream-based recommenders.

4.1.1. The Polarity Learning Problem

In [63], the authors decompose a document into individual sentences, treating each sentence as a distinct instance, and apply weighted Multiple Instance Regression (MIR) to predict the overall polarity of the document. Unlike these sentence-focused approaches, the present work does not consider the document's semantics or its sentences. Instead, the focus is exclusively on the target entity, assuming that the entity is known beforehand.

As summarized by [78], earlier research in static opinion mining generally achieved accuracy rates ranging from 60% to 90%, primarily addressing binary or three-class classification problems (positive, negative, neutral). In contrast, the task explored here involves a multi-class classification problem with five classes, similar to the work of [62], where Support Vector Machines (SVMs) were employed to predict star ratings for movie reviews. However, while [62] developed models for each individual review author, the approach in this work is more closely aligned with stream recommender systems, a topic further elaborated in section 4.1.4. The advent of large language models (LLMs) and text embeddings revolutionized text mining [60, 4] in recent years, but they are some of the most computationally expensive models to train and also to adjust to concept drift [83, 21], which is why they are not yet commonly used on data streams [6].

4.1.2. Target Entity Analysis

In [19], the authors employ probabilistic models to predict both the polarity of an opinionated document and the polarity of the document's target entity. Jakob and Gurevych, in [38], explore the potential of Conditional Random Fields for identifying the specific entity to which an opinion refers. Similarly, Qiu et al. combine target extraction with an analysis of the words in an opinionated document to assess its polarity [67]. Zhang and Liu extend this further by examining the aspects of an entity conveyed in opinionated texts, differentiating between direct opinions on a single entity and comparative opinions that involve multiple entities [86]. Vakulenko et al. formalize the problem of linking sentiment to a target in opinionated documents. They define "sentiment-target linking" as the task of pairing sentiment tokens with target tokens extracted from a sentence, ensuring that the sentiment is correctly attributed to the target entity [80].

While these works focus on target extraction or linking sentiment to targets, our approach differs significantly. We concentrate on predicting the polarity of opinionated documents for which the target entity is already known—such as products on rating platforms, or opinions on hotels, restaurants, and events. Additionally, this work addresses this task in a stream setting, where the goal is to predict the polarity of the most recent opinion on a given entity without considering the content of the opinion.

4.1.3. Learning on a Data Stream

In a data stream setting, consider the timepoints $t_1, t_2, \ldots, t_j, \ldots$, at which opinionated documents—referred to as *instances*—arrive. At each timepoint, t_j, a model M_j is built based on all instances observed up to that point. When a new instance arrives at t_{j+1}, the model processes the incoming instance, predicts its label, and updates itself based on the actual label, producing M_{j+1}.

The goal of stream classifiers is to quickly adapt to changes in the data, known as concept drift [12]. Stream learning algorithms differ in the amount of historical data they retain and the types of information they prioritize. A key consideration is memory retention, or the application of *forgetting* mechanisms. For a comprehensive discussion on forgetting strategies in stream learning, refer to [30].

Research has shown the advantages of online linear and discriminant analysis for handling streaming data [3]. In [82], both aggressive and gradual forgetting strategies were examined for the classification of opinionated documents, with our work referencing the gradual method from that study, which will be employed as the text-aware entity-ignorant model.

4.1.4. Stream Recommenders

From a technical perspective, the approach presented here shares several characteristics with stream-based recommender systems. Recommenders typically predict whether a user will like an item by analyzing the user's prior ratings, the item's rating history, and the similarities between both users and items. In systems utilizing matrix factorization, this prediction task is framed as a regression problem, where a model of the rating matrix is learned to predict missing values. In streaming environments, recommendation systems adopt strategies to handle outdated ratings, as outlined in [81] and [57]. These strategies frequently adjust the retention of information, preserving more data about infrequently rated items compared to frequently rated ones.

Despite these technical similarities, the techniques used in stream recommenders are not directly applicable to our task. The objective here is not to generate recommendations, so factors such as a user's past opinions or their similarity to other users are irrelevant. Likewise, item-item similarity is not a key consideration; for instance, even if two hotels offer nearly identical services or two watches share the same features, the opinions received for these items may differ significantly due to factors like brand perception or geographical location.

4.2. Polarity Predictors on Entity-Level Substreams

A stream of opinions explicitly linked to predefined entities, such as products, movies, hotels, and restaurants, is considered. Let $E = \{e_1, e_2, ..., e_n\}$ represent the set of entities, and $t_1, t_2, \ldots, t_j, \ldots$ denote the timepoints at which these opinions arrive. Each opinion, or *instance*, is represented as a pair $i_{k,j}$, where $e_k \in E$ refers to the associated entity and t_j indicates the time of arrival. Similar to traditional opinion stream classifiers, the task is to predict the polarity $label(i_{k,j}) \in \mathcal{L}$, where $\mathcal{L}$ signifies the set of possible labels. However, the approach presented here differs in terms of the data used for learning and whether this data is processed within a single model or distributed across multiple models.

4.2.1. Entity-Level Substream and Models

For each entity $e_k \in E$, a substream $substream(e_k, t_j)$ is constructed and maintained, storing all instances observed for e_k up to time t_j. Thus, at t_j, this substream contains a list of instances $substream(e_k, t_j) = [i_{k,k_1}, \ldots, i_{k,k_l}]$, where $t_{k_l} \leq t_j$ represents the most recent timepoint at which an opinion about e_k was received. The number of instances associated with different entities can vary significantly, leading to substantial differences in the length of each entity's substream and the amount of training data available for model learning.

It has been postulated that for entities benefiting from entity-centric learning, the substream of labels would differ from the overall population. This led to the conjecture that an entity's label history alone might suffice to predict future labels effectively.

To investigate this postulation, entity-centric predictors are developed, and a model for each entity e_k is learned and adapted. These predictors are divided into two categories: those that only utilize the labels of the instances in $substream(e_k, t_j)$ and those that also consider each instance's content (features). Additionally, models are built that generalize across all entities, again distinguishing between those that access only the labels and those that also exploit instance content. A conventional opinion stream classifier is used as a baseline. This classifier builds a model using all incoming instances, accessing their features but disregarding the entities they refer to.

4.3. Entity-Centric and Entity-Ignorant Predictors

A family of *entity-centric predictors* (denoted as "E") is proposed, where a distinct model is developed for each entity, considering only the labels of the opinions associated with that entity. The primary distinction between these predictors lies in the extent of historical information they retain. Among the various forgetting mechanisms commonly applied in stream classification [30], a sliding window approach is adopted, thereby distinguishing between *window-based* models and those that retain *all-past* data.

This family of predictors is contrasted with two baseline groups: the *baselines* (denoted as "B"), which construct a single model from the entire dataset, and the *entity-centric baselines*, which also utilize the content of the opinions for each entity.

(E1) Entity-centric, all-past-based predictors.

Each predictor of this subfamily builds a model M_k per entity e_k. More precisely, a predictor builds a model M_{k,k_j} with all data remembered until t_{k_j} and adapts it upon arrival of the next instance on e_k, at $t_{k_{j+1}}$. Each model of this subfamily is only updated when a new instance arrives for the respective entity.

The predictors are:

Prior Let $i_{k,j+1}$ be the opinion arrived at t_{j+1}. Since it refers to e_k, the label assigned to it is drawn from $substream(e_k, t_j)$.

This model predicts the next label to be the most frequently observed label in the substream of e_k before t_{j+1}. Alternatively, it is called the majority-label classifier.

HMM-based predictors For each entity e_k, a Gaussian HMM with three states is learned on $substream(e_k, t_j)$ and adapted whenever a new opinion comes for e_k. The states are intended to capture (1) the tendency for entities to receive mostly negative opinions, (2) the tendency for entities to receive mostly positive opinions, and (3) other tendencies.

The motivation behind these states is that people's opinions towards an entity may drift towards more positive/negative values. For example, an eagerly expected release of a cellphone may provoke enthusiastic opinions at first, later shifting towards negative ones, e.g. if some expected functionalities are unavailable or if some flaws or construction errors emerge. Improvements may lead again to positive opinions. Such transitions are also expected in opinions on hotels or restaurants, where temporary changes at the service level may lead to more positive or negative ones. To predict $label(i_{k,j+1})$, each candidate label from $\mathcal{L}$ is added to the sequence of already observed labels ($label(i_{e_k,k_1})$, $label(i_{e_k,k_2}), \ldots, label(i_{e_k,k_j})$) and we compute for each potential label the likelihood of being observed in this sequence of labels given the current HMM parameters. The candidate label with the highest score is then selected as the prediction $label(i_{k,j+1})$.

(E2) Entity-centric, window-based predictors.

For each of the predictors mentioned earlier, a window-based variant is derived, applying a sliding window of length w (in terms of instances) over each substream. These models consider only the instances within the window for learning, resulting in the *WindowedPrior* and *WHMM* predictors, which utilize only the labels.

In the specific case where labels are numerical, as can be the case with ordinal labels like the considered star-ratings, two additional predictors are introduced:

Regression The label for $i_{k,j+t}$ is predicted using a linear regression model trained within the window. Since the regression model returns real values, the prediction is rounded to the nearest ordinal label.

Simple Moving Average(SMA) This predictor sets as $label(i_{k,j+1})$ the average over all labels within the window, after rounding to the closest integer.

(B) Baselines for the Predictors.

To evaluate our approach, various baselines are introduced as counterparts to the aforementioned methods.

Entity-centric, all-past-based one-nearest neighbor ("KNN") This algorithm exploits the contents of the instances in $substream(e_k, t_j)$, as well as the contents of the instance $i_{k,j+1}$. For vectorization of the opinions, a bag-of-words approach is used after stopword removal. Then, the label assigned to $i_{k,j+1}$ is the label of the most similar instance to it among the instances in $substream(e_k, t_j)$.

Entity-centric, window-based one-nearest neighbor ("WKNN") The counterpart of the aforementioned baseline, considering only the instances that are inside the window.

Global, window-based one-nearest-neighbor ("GKNN") This simplistic baseline considers the N most recently observed opinions and assigns to $i_{k,j+1}$ the label of its nearest neighbor among those N. In our experiments, we set $N = 1000$.

Global, all-past-based baseline "Global Prior" A baseline that forgets none of the past data is atypical in stream learning because it is irresponsive to drift. We still consider such a baseline, which ignores the texts but remembers the labels and assigns to an opinion the most likely label observed thus far in the stream.

Reference Strategy: "Multinomial Naive Bayes with Forgetting" All prediction methods are compared against a multinomial Naive Bayes (MNB) classifier with forgetting introduced in [82], which uses the review text to make label predictions. The MNB classifier was optimized for the use on data streams and uses occurrence counts for each label and conditional counts, tracking the occurrences of each tracked feature with each label to make predictions using Bayes' theorem. The 1000 most frequently occurring words make up the features[1]. Of the two forgetting MNB variants proposed in [82], this work uses the the 'fadingMNB' approach, which decays word counts more gradually than the 'aggressiveFadingMNB'. A traditional MNB, without forgetting, is also employed for reference but takes much longer to execute as it is not optimized for data streams.

The review text was subject to basic preprocessing steps, such as removing non-alphanumeric characters, stemming (reducing derived words to their roots), and stopword elimination (removing words like 'and', 'the', 'a', etc.).

4.4. Evaluation Framework

The proposed entity-centric learning approach differs greatly from conventional opinion stream mining due to the nature of the information being exploited. To ensure fair comparisons, the stream alignment method and evaluation settings of the framework are outlined next.

[1] In case the data for the whole period does not fit in memory, the frequent words are computed on the first half of the dataset

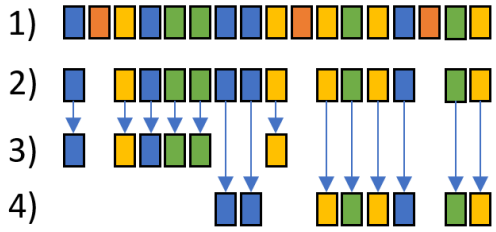

Figure 4.2.: 1) A stream of opinions with four entities, time is progressing from left to right; 2) removal of entities having fewer than $2 \cdot w$ opinions, where $w = 2$ (the orange entity is removed as $3 < 2 \cdot 2$); 3) the first $w = 2$ opinions per entity are placed in the *training stream*, and 4) the remaining opinions are placed in the *evaluation stream*. Figure reproduced with permission from [7] ©ACM 2018 Copyright held by the owner/author(s).

4.4.1. Aligning a Stream of Opinions to their Target Entities

A predictor that leverages the target entity e of an opinion i for label prediction requires at least one prior opinion on e. For evaluation purposes, a sufficient number of opinions per entity must be available for both training and testing. The stream is therefore divided into a *training stream* and an *evaluation stream* as follows:

1. A threshold $w \geq 1$ is set as the minimum number of opinions (instances) required for prediction. In our experiments, this threshold is varied.
2. Entities with fewer than $2 \cdot w$ opinions are excluded from the stream.
3. For each remaining entity e, the first w opinions are allocated to the training stream.
4. The remaining opinions for e are allocated to the evaluation stream.

This process is illustrated in Figure 4.2, where time progresses from left to right. In the figure, two boxes of the same color represent opinions on the same entity. The example includes four entities (depicted in blue, orange, yellow, and green).

Figure 4.2, line 1) shows the original stream. The gaps in line 2) indicate removed entities: $w = 2$, therefor the opinions on the orange entity were skipped. In line 3), we see the first $w = 2$ opinions of each retained entity as part of the training stream. The remaining ones are placed in the evaluation stream, cf. line 4).

This setup allows to compute prediction performance for each entity individually. It also ensures that all entities included in the evaluation provide at least w instances for training and testing so that the models can actually learn and predict.

4.4.2. Entity-Centric Evaluation

The framework provides two distinct types of evaluation: one that mirrors conventional stream evaluation and another that operates at the entity level.

The first approach uses prequential evaluation on the evaluation stream, excluding all instances assigned to the training stream. This filters out the w instances used to initialize the entity-centric models. The goal is to assess the performance of these models after initialization with w instances and at least w test instances. In this evaluation, all entity-centric models are treated as a single aggregated model, and a

unified performance score is calculated. The performance metrics considered include accuracy, balanced accuracy, and κ^+ for nominal labels, and RMSE for ordinal or numeric labels. For κ^+, an entity-centric no-change classifier is employed, which propagates the last seen label of an entity to the next incoming instance of that entity. Performance is calculated over the whole stream instead of employing a window or chunk-based evaluation, as the overall performance is of interest, not the performance at certain points in time.

The second evaluation method operates at the entity level, where performance is calculated separately for each entity. For each prediction method M and entity e, performance is evaluated on the entity's substream S_e, based on the true labels y_{S_e} and the predicted labels $\hat{y}_{M_e,S_e}$ on the evaluation stream:

$$performance_e(S_e, M_e, metric) = metric(y_{S_e}, \hat{y}_{M_e,S_e}) \tag{4.1}$$

Following this, a pairwise comparison is conducted between each pair of prediction models $M_1, M_2 \in Models$. The percentage of entities where $performance_e(S_e, M_{1_e}, metric) > performance_e(S_e, M_{2_e}, metric)$ is then calculated.

4.5. Experiments

The predictors and baselines outlined in 4.3 are compared in this experiment. For all entity-based predictors and baselines, the minimum number of opinions per entity, denoted as w, is varied with values of $5, 10, 15$, and 20. For the window-based subfamily, window sizes of $5, 10, 15$, and 20 instances are considered, with the constraint that the window size cannot exceed the value of w. Given that w is relatively small in this set of experiments, the values of w and W are varied simultaneously, i.e., $w = W$.

4.5.1. Datasets of the Experiments

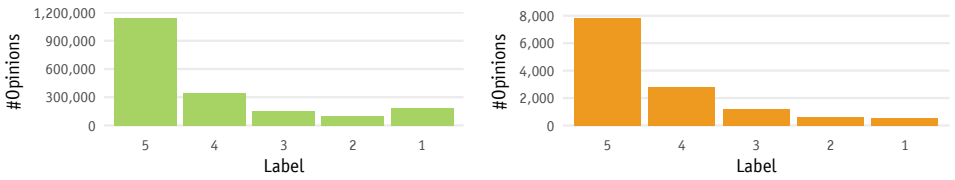

Figure 4.3.: Distribution of opinion labels for **tools** (green) and **watches** (orange)

For the experiments, subsets of the Amazon review dataset [34] are used, namely the 'Tools and Home Improvement' denoted as **tools** hereafter, and the set of opinions on **watches**, extracted from the 5-core version[2] of the 'Clothing, Shoes and Jewelry' subset and denoted as **watches** hereafter. An overview of the two datasets can be found in Table 4.1.

Collection	*#entities*	*#opinions*
tools	260,659	1,926,047
watches	1,221	13,027

Table 4.1.: Entities and reviews in **tools** and **watches**

[2] http://jmcauley.ucsd.edu/data/amazon/links.html

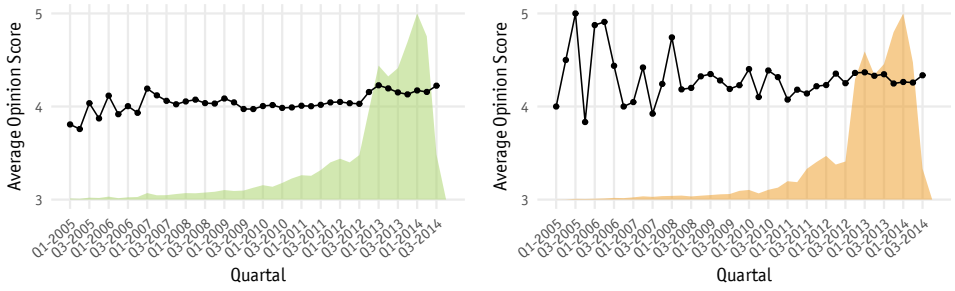

Figure 4.4.: Average product rating over time for **tools** (green) and **watches** (orange). The height of the background histogram depicts the rating density within the quarter.

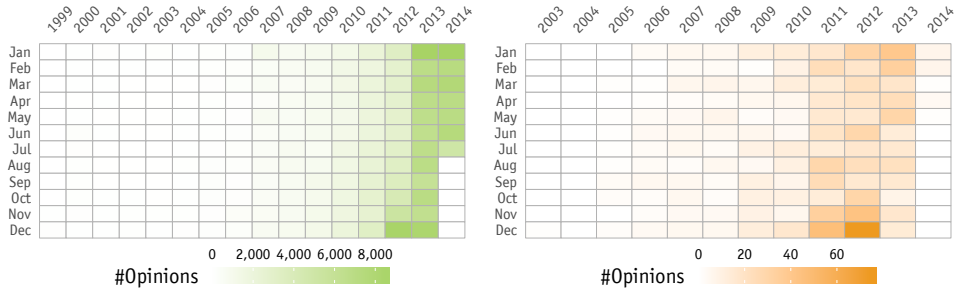

Figure 4.5.: First opinions on an entity for **tools** (green) and **watches** (orange).

Figure 4.3 illustrates the distribution of opinion labels (1 to 5 stars) for the **tools** (green) and **watches** (orange) datasets. The class distribution is clearly skewed, with the label "5" being more frequent than all other labels combined in **tools** and the most frequent label in **watches**. Treating the labels as numerical values, the average rating exceeds 4, reflecting a strong tendency towards positive opinions.

Figure 4.4 further confirms this trend by showing the average product rating for each quarter from 2005 onwards. After some initial fluctuations, particularly in **watches**, the average rating stabilizes slightly above 4, while the volume of posted opinions increases significantly after 2010.

Figure 4.5 presents calendar heatmaps for both datasets, illustrating the number of *first* opinions on entities over time. The increasing intensity in the heatmaps indicates a growing number of entities receiving opinions. For **tools**, this proliferation begins steadily in late 2012, whereas for **watches**, a peak is observed in December 2012, followed by a subsequent decline.

Although Figure 4.5 indicates that more and more entities have received an opinion

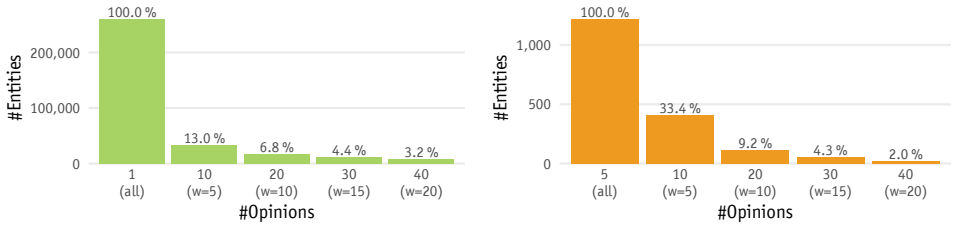

Figure 4.6.: Entities with at least $2 \cdot w$ opinions for **tools** (green) and **watches** (orange).

in the last years, most entities do not go beyond this number. As can be seen in Figure 4.6, only around 12% of the `tools` and of the `watches` have at least $2 \cdot 5$ opinions (for $w = 5$), and this percentage drops rapidly as we increase the w. Hence, the evaluation is done on a small part of the original stream. This detailed description of the datasets was part of unpublished work concerning [7].

4.6. Results and Discussion

The overall prediction quality is first evaluated by analyzing performance across the entire data stream, followed by a comparison of model performance at the entity level. Since the results are similar across both datasets, the discussion will focus on the `tools` dataset, which includes a larger number of instances and entities. The results for the `watches` can be found in the appendix.

The upper section of Figure 4.7 presents the key metrics of interest, including accuracy, balanced accuracy, RMSE, κ^+, and execution time. The lower section displays pairwise comparisons between the models based on κ^+.

Prediction Quality

Entity-centric models are aggregated into a single *conglomerated* global model (see subsection 4.4.2). Predictably, text-aware global models outperform these conglomerated models, primarily due to the integration of historical texts and labels.

Among the entity-centric methods, the conglomerated text-ignorant *SMA* exhibits the lowest RMSE, surpassing even text-aware models such as *KNN* and *WKNN*, as well as the global model *GKNN*. Additionally, *SMA* has a notably faster execution time (see the last column), making it particularly suited for numerical label tasks.

For categorical labels, the conglomerated version of *WPrior* emerges as the best entity-centric method that does not incorporate text data, though it is still surpassed by the text-aware models. The variation in performance across methods—especially when comparing RMSE with accuracy and balanced accuracy—suggests that RMSE is a more effective metric in scenarios with skewed class distributions, such as the 5-star ratings in our datasets.

The underperformance of the conglomerated entity-centric *HMM* and *WHMM*, combined with their excessive execution times, renders HMMs a less appealing choice for text-ignorant entity-centric learning. A more detailed analysis of model-specific errors is presented in Appendix A.1.

Finally, the conglomerated models yield a κ^+ value of zero, as κ^+ is computed over the entire data stream, while entity-centric algorithms are optimized for individual entities. As illustrated in Figure 4.1, entity-centric models can surpass global models for specific entities. Therefore, κ^+ is further examined on an entity-by-entity basis in the following section, as it was designed for streams and considers the temporal trends of each entity. For example, a product that received a 1-star review is more likely to receive a 1-star review in the near future, compared to a product, which recently received a 5-star review.

Pairwise Comparison of Prediction Quality for each Entity

The lower section of Figure 4.7 provides a pairwise comparison of predictors, showing the percentage of entities where the row predictor achieves a higher κ^+ than the column predictor. For instance, *Prior* (first row) outperforms *Regression* (fourth

column) for 44% of the entities, as indicated by a green bar filling 44% of the cell. Conversely, *Regression* outperforms *Prior* for only 1% of the entities (see fourth row, first column), with both performing equally for the remaining 55% of the 33,989 entities.

Notably, *MNB* and *MNBF* outperform other predictors for up to 58% of entities. However, for more than 40% of entities, their κ^+ is zero or matches that of another predictor. Specifically, *MNB* and *MNBF* are surpassed by *SMA* for 6% of the entities, by *Prior* for 18%, and by *WPrior* for 13%. These results suggest that entity-centric models offer distinct advantages for certain entities, highlighting the importance of identifying these characteristics in future research.

#predictions: 1247724			Acc.	RMSE	Bal. Acc.	κ+ loc. p.	Exec. Time [min:sec]
Entity-Centered	Text-ignorant	Prior	0.592	1.505	0.242	0.000	00:06
		WPrior	0.521	1.567	0.259	0.000	00:08
		SMA	0.385	1.332	0.244	0.000	00:07
		Reg	0.370	1.648	0.229	0.000	05:12
		HMM	0.580	1.530	0.242	0.000	443:09
		WHMM	0.471	1.638	0.247	0.000	1193:26
	Text-aware	KNN	0.512	1.439	0.290	0.000	67:40
		WKNN	0.482	1.544	0.269	0.000	36:52
Global	Text-ign.	GPrior	0.598	1.552	0.200	0.001	00:05
	Text-aware	MNB	0.652	1.184	0.409	0.224	294:06
		MNBF	**0.653**	**1.180**	**0.409**	**0.227**	12:48
		GKNN	0.484	1.580	0.278	0.000	244:20

#e: 33989	Prior	WPrior	SMA	Reg	HMM	WHMM	KNN	WKNN	GPrior	MNB	MNBF	GKNN
Prior		.22	.37	.44	.20	.40	.35	.37	.01	.18	.18	.42
WPrior	.01		.26	.33	.09	.29	.24	.26	.01	.13	.13	.32
SMA	.01	.01		.12	.03	.10	.08	.08	.01	.06	.06	.13
Reg	.01	.01	.02		.01	.02	.03	.03	.01	.02	.02	.06
HMM	.03	.17	.31	.37		.33	.28	.31	.02	.15	.15	.36
WHMM	.02	.02	.13	.18	.03		.10	.11	.01	.05	.05	.19
KNN	.04	.08	.19	.24	.07	.19		.15	.04	.08	.08	.23
WKNN	.03	.05	.15	.20	.05	.14	.08		.03	.06	.06	.19
GPrior	.09	.28	.42	.48	.26	.45	.39	.42		.20	.20	.46
MNB	.37	.43	.53	.58	.42	.54	.50	.52	.35		.04	.55
MNBF	.38	.44	.54	.58	.42	.54	.51	.53	.35	.06		.56
GKNN	.05	.07	.11	.13	.07	.10	.09	.10	.04	.04	.04	

Figure 4.7.: The top subtable compares the performance and execution time of all predictors on `tools` with $w = 5$. Higher scores are preferable for (balanced) accuracy and κ^+, while lower values are favored for RMSE. The best results are highlighted in bold, and the top-performing text-ignorant predictors are underlined. The bottom subtable shows the percentage of entities where the κ^+ of the predictor in the row exceeds that of the predictor in the column. Figure reproduced with permission from [7] ©ACM 2018 Copyright held by the owner/author(s).

4.7. Conclusion

In this chapter, the impact of incorporating entity-specific information into the prediction of opinion labels is explored. A framework for *entity-centric* document polarity prediction is introduced, where a distinct model is developed for each entity, utilizing the *substream* of instances associated with that entity. Some of the proposed predictors are text-aware, while others rely solely on historical label data. This chapter focuses on the performance of entity-centric methods that rely on labels. This is based on the intuition that entities that are hard to predict by an entity-ignorant model probably show label distributions that differ from the overall trend. Experiments were conducted on two Amazon review datasets where entities are products and instances are reviews (opinions) pertaining to these products.

Depending on the performance metric, up to 18% of the entities benefit from simple, text-ignorant predictors, which outperform more complex methods that consider the full opinion stream without factoring in the connection between the opinion and the corresponding entity. However, for the majority, models constructed from the entire opinionated stream yield better outcomes.

The framework includes both basic entity-based predictors, such as those that fit a regression line to past labels or select the most frequently observed label, and more sophisticated methods like Hidden Markov Models (HMMs). Nonetheless, experimental results show that the extended processing time required for more complex models does not always lead to improved predictions, with simpler predictors often being more competitive.

These findings are subject to certain caveats. The two datasets used in the experiments exhibit a class distribution heavily skewed toward the most positive label, which may affect the relative performance of entity-centric models when compared to traditional stream classifiers. Moreover, variations in the number of opinions and class distribution across entities were not taken into account.

Entity-centric predictors require a minimum number of opinions before they can predict future opinions, whereas traditional stream classifiers are able to predict opinion labels even without prior data for a specific entity, as long as they have been initialized with data from other sources. Consequently, this entity-centric approach should be viewed not as a replacement for traditional stream learning methods, but as a low-resource complement.

These findings have prompted a deeper investigation into the characteristics of entities for which entity-centric models perform well. Here, it was first assumed that an entity's label distribution is indicative of its model performance. Unfortunately, initial investigations in this direction were inconclusive. The results were very sensitive to the distributions themselves and also to the degree to which two distributions need to differ before they can be called 'different'.

A more obvious characteristic that is known to impact model performance is the length of an entity's substream, i.e., the amount of training data available for constructing the entity-centric model. The next chapter extends this work by developing an evaluation framework focused on substream length and combining the predictions from entity-centric and entity-ignorant models to enhance performance even when only a few instances are available for a given entity.

5. Entity-Centric Ensemble

This chapter describes the work published in [8]. It addresses the following RQs:

RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model?
RQ2: How can the memory footprint of entity-centric models be reduced?

Its primary focus is RQ1, but it also highlights the growing memory demands of employing entity-centric models, which relates to RQ2.

The initial work on entity-centric learning, discussed in chapter 4, demonstrates that simple entity-centric models can enhance performance for specific entities. However, it also emphasizes that the traditional entity-ignorant stream mining algorithm, which processes instances from all entities without differentiation, achieves the best overall performance. It is assumed that the length of an entity, and thus the amount of available training data, plays a significant role in the applicability of entity-centric learning for that entity.

To test this assumption, a new entity-centric evaluation framework is proposed. This framework focuses on the number of instances in an entity's substream available for training and assesses how many training instances are required for an entity-centric model to reach an RMSE comparable to that of an entity-ignorant model.

Additionally, several ensemble models that combine predictions from both entity-centric and entity-ignorant models are introduced. The newly developed framework is used to identify the point at which an ensemble's predictions surpass those made by either purely entity-centric or purely entity-ignorant models. Given that entity-centric models relying only on label data were frequently outperformed by the text-aware entity-ignorant model, the best-performing text-aware model (MNBF) from chapter 4 is utilized for the entity-centric predictions as well.

This chapter begins with a review of related work, followed by a detailed explanation of the evaluation framework in section 5.2. Next, the various ensemble models that combine entity-centric and entity-ignorant predictions are introduced in section 5.3. The chapter ends with a discussion of the experimental results in section 5.4 and concluding remarks in section 5.5. The sections in this chapter are based on their counterparts in [8].

5.1. Related Work

Stream algorithms that relax the assumption of independence between incoming reviews and product ratings are particularly relevant to our research, especially those that take into account the relationship between a rating and its corresponding product for both learning and forgetting.

One of the earliest references to the connection between an entity, such as a product, and its associated temporal events is captured in the concept of 'context.' In [36], Hong et al. define 'Context as any information that can be used to describe the situation of an entity.' Similarly, [61] demonstrates how context-aware features can enhance the

performance of recommender systems. Unlike context-aware learning, which defines an entity within a particular, predefined or learned context, our approach emphasizes the relationship between an entity and the various observations associated with it over time, using this connection to inform the model.

Several works align more closely with our notion of entity-centric processing over a stream of observations, including those by [25, 79, 58] and [7]. In [25], Fafalios et al. present entity-specific metrics, such as popularity and sentiment, extracted from the Twitter social stream to track the evolution of individual entities. [58] introduce a text stream classification algorithm that manages changes in the feature space by tracking the historical data of each feature. Their ensemble approach predicts future values by identifying patterns such as regular, seasonal, autoregressive, and short-term bursts, all handled by individual base predictors. In this context, each feature acts similarly to an entity with a corresponding historical record.

In [79], Unnikrishnan et al. propose connecting all observations related to an entity into an ongoing time series, leveraging similarities across entities to develop a predictor capable of forecasting labels for both near-term and long-term observations. Although their method also handles a stream of observations, they assume only an initial set of labels is provided, aiming to predict future labels far into the future. In contrast, our method adheres to a conventional stream classification framework, where new labels are revealed incrementally as the stream progresses.

In this chapter, the MNB with gradual forgetting [82] is employed as both the entity-centric and entity-ignorant model, which gives the entity-centric models access to the review text. Next, a description of the motivation and design of the evaluation framework will be provided before going deeper into the ensemble methods.

5.2. Entity-Centric Evaluation Scheme

As previously mentioned, it is hypothesized that the amount of available training data, particularly the length of an entity's substream, is a crucial factor influencing the performance of entity-centric models. This observation led to the development of an entity-centric evaluation framework aimed at determining how many entities benefit from entity-centric learning based on the given quantity of training data.

To ensure fair comparisons among entity-centric models, it was necessary to control the amount of training and test data per entity, allowing performance scores to be calculated from the same number of predictions across all entities.

Thus, the evaluation framework was designed to begin with a conventional prequential evaluation, followed by an assessment in an entity-centric context, focusing on performance relative to a specific amount of training and test data.

This chapter uses RMSE as the performance metric due to its suitability for ordinal labels, where larger prediction errors are penalized more heavily than smaller ones. For example, predicting a 5-star review as 4 stars is more favorable than predicting it as 1 star.

The evaluation framework is depicted in Figure 5.1, where $ECCE$ represents an ensemble method based entirely on entity-centric predictions, while $EIGC$ is an entity-ignorant classifier. Further details on $ECCE$ and other ensemble models are provided in section 5.3, but for now, it is important to note that $ECCE$ generates entity-centric predictions, whereas $EIGC$ delivers predictions that do not incorporate entity-specific information.

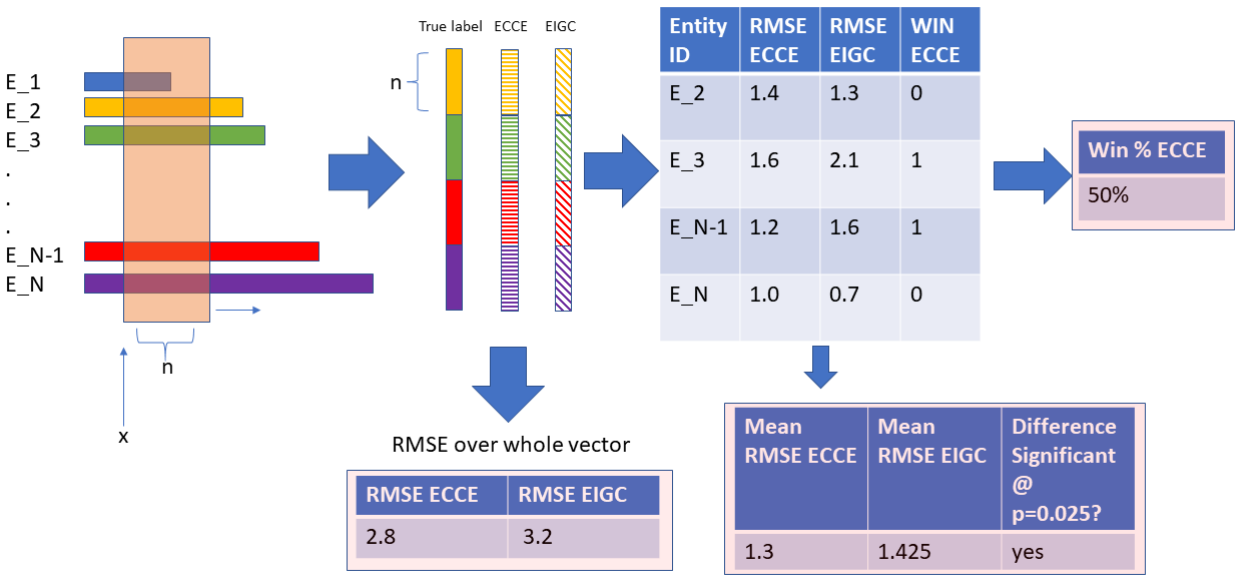

Figure 5.1.: This figure illustrates the entity-centric evaluation process, using *ECCE* as an example, which delivers solely entity-centric predictions. On the left, entities are aligned, and the next n observations are extracted according to a specified threshold x, which represents the number of training instances per entity. Entity E_1 in blue does not meet the criteria, as it lacks the n required observations at threshold x. Moving along the arrow to the right, the n observations of each entity are used to form vectors of true labels and predictions from *ECCE* (dashed lines for entity-centric predictions) and *EIGC* (diagonal dashed lines for entity-ignorant predictions). Following the downward arrow, the result of the first component, the RMSE at threshold x, is displayed. To the right, the RMSE for each entity is shown, which contributes to the second component (percentage of wins) and the third component (statistical significance test). Figure reproduced with permission from [8] ©2019 ACM.

The framework is organized around three key components:

First Component - The Effect of Available Training Data per Entity: The process begins by aligning the substreams of entities and generating a vector of n predictions for each entity at a given threshold x, which corresponds to the amount of training data available for the entity-centric models. This prediction vector is denoted as V_P, with V_T representing the true labels. The RMSE between these two vectors is then computed (e.g., $RMSE(V_{ECCE}, V_T)$), and the result is compared to the entity-ignorant baseline ($RMSE(V_{EIGC}, V_T)$). This process is illustrated in Figure 5.1, following the first arrow to the right and then the downward arrow.

Second Component - Evaluating Error Reduction per Entity: To assess how many entities benefit from entity-centric and ensemble-based methods, the RMSE is calculated for the next n predictions for each entity, as shown in the table on the right side of Figure 5.1. If the entity-centric ensemble results in a lower RMSE compared to the baseline for a given entity, it is marked as a "win" for the ensemble. This enables the calculation of the percentage of entities for which the entity-centric approach outperforms the baseline.

Third Component - Statistical Testing Over Time: At each threshold x, the table from the second component is used to determine whether the performance of the ensemble method at the entity level is statistically superior to the baseline using a Wilcoxon signed-rank test. This process is depicted under the downward arrow on the right side of Figure 5.1.

This framework is employed to explore the following questions:

- What percentage of entities benefits from entity-centric stream learning, offering improved classification of opinionated texts compared to conventional stream classification given a specific amount of training data x?
- How much training data is necessary before entity-centric models match the performance of entity-ignorant models, and what are the differences in performance when limited training instances are available (the cold-start problem)?

In addition to these evaluation methods, performance metrics are presented in a conventional format, showing the RMSE across various methods using non-overlapping chunks, with each chunk consisting of 1000 predictions.

Through this framework, it was observed that between 400 to 500 training instances per entity were required before the performance of the entity-centric models began to match that of the entity-ignorant model on our datasets; see section 5.4.2. This number is substantially higher than the average number of instances per entity. For example, on the bigger dataset `tools` only 3.2% of the entities have at least 40 instances, see section 4.5.1. This makes learning approaches relying solely on entity-centric models infeasable, which led to the idea of combining the predictions from both entity-centric and entity-ignorant models into various ensembles, as discussed in the next section.

5.3. An Ensemble with Two Voting Members

The proposed ensemble methods comprise two primary components: a traditional entity-ignorant stream classifier, which treats all observations as independent, and a set of Single-Entity Classifiers (SECs), with each SEC being responsible for processing the substream of data related to a specific entity. The functioning of the SECs is explained first, followed by a description of how the overall ensemble operates.

The Entity-Centric Ensemble Member

For each entity $e \in E$, a *Single-Entity Classifier* (SEC_e) is trained and continuously updated to process only the observations associated with that entity, denoted by $S_e = \{obs_{e,1}, obs_{e,2}, \ldots\}$. Whenever a new entity e is encountered, a corresponding SEC_e is created. The SEC_e is activated to classify and update itself only when an observation related to that entity arrives.

Each SEC utilizes a Multinomial Naive Bayes classifier with a 'gradual fading' mechanism, as described in [82]. This mechanism decreases the word count for each class as time progresses since the word was last observed for that class. Within the substream T_e, this fading causes the word count per class to decrease depending on when the word was last seen for that specific entity. Since each SEC operates exclusively on its designated substream T_e, the fading process and word counts vary from one entity to another.

The Entity-Ignorant Ensemble Member

The second component is a traditional stream classifier that does not differentiate between entities and is referred to as the 'Entity IGnorant Classifier' (EIGC).

The EIGC works within the same feature space F as the SECs but processes all the observations from the stream D. It is initialized with the first observation and is continuously updated as new data arrives, both for learning and prediction purposes. Similar to the SECs, the EIGC also employs the gradual fading Multinomial Naive Bayes method outlined in [82], but the fading is applied to the entire stream rather than entity-specific substreams.

Next, the different strategies for combining the SECs and EIGC into ensembles are discussed.

5.3.1. Ensemble Variants Based on Weighting

Following the descriptions in sections 3.2 and 3.3 of [8], three distinct weighting strategies are explored for combining the SECs and EIGC, each leading to a different ensemble variant.

Variant 1: The Entity-Centric Classifier Ensemble (ECCE) The $ECCE$ variant is based on the premise that a classifier requires a certain minimum number of training observations, denoted by x, before it can produce reliable predictions. At the start of the stream, $ECCE$ initializes both the EIGC and a SEC for the entity e associated with each observation. Once the SEC for an entity e has accumulated x observations, it becomes eligible to make predictions, and $ECCE$ transitions from using the EIGC to using the specific SEC_e for future predictions concerning that entity. As the EIGC is trained on the entire stream, it becomes the first operational component,

addressing the cold-start problem for new or infrequently observed entities. The EIGC continues to be updated alongside the SECs as more data arrives.

Variant 2: The Entity-Centric Weighted Ensemble (ECWE) In the $ECWE$ variant, even after the cold-start phase, the EIGC continues to contribute to predictions of all entities. Specifically, $ECWE$ assigns a weight w to the SECs and a weight of $1-w$ to the EIGC, with the final prediction determined by the weighted combination of votes from the SECs and EIGC.

Variant 3: The Entity-RMSE Weighted Ensemble (ERWE) The $ERWE$ variant differs from $ECWE$ by using dynamic weights based on the prediction error of each ensemble member, giving greater voting power to the model with the lower error rate. For each new observation o, let e represent the corresponding entity.

The weight assigned to SEC_e is calculated as:

$$wSEC(e) = \frac{RMSE(EIGC_e)}{RMSE(EIGC_e) + RMSE(SEC_e)}$$

The weight assigned to $EIGC$ for entity e is:

$$wEIGC(e) = \frac{RMSE(SEC_e)}{RMSE(EIGC_e) + RMSE(SEC_e)}$$

In this context, the RMSE is used as the classification error metric, assuming the labels are ordinal. For binary classifications (positive or negative labels), a standard misclassification error can be used instead of RMSE.

The weight assigned to the EIGC for a given observation depends on the performance of the SEC for the corresponding entity. This enables $ERWE$ to assign higher weight to the EIGC when the SEC has not yet achieved good performance, while prioritizing the SECs as soon as they outperform the EIGC. Furthermore, unlike $ECWE$, $ERWE$ does not require predefined weights, making it parameter-free.

5.4. Experiments and Results

The first RQ of this thesis asks: **To what extent can entity-centric models improve performance compared to an entity-ignorant model?**, which this chapter aims to address using the proposed evaluation framework and the proposed methods. For the evaluation, the performance of the proposed entity-centric ensembles is compared with that of the entity-ignorant classifier ($EIGC$), using the same datasets: `tools` (33,989 entities and 1,416,766 instances) and `watches` (8,162 entities and 487,741 instances), as described in Section 4.5.1 of the previous chapter.

5.4.1. Evaluation Procedure

Following the evaluation framework outlined in Section 5.2, the $RMSE_{Cl}$ is calculated for each classifier $Cl \in \{ECCE, ECWE, ERWE, EIGC\}$ once each entity has received at least $x \geq 2$ instances. Although $EIGC$ is capable of making predictions earlier, the aim of this evaluation is to assess the contributions made by the entity-centric models. Therefore, performance is only measured after the Single-Entity Classifiers (SECs) have accumulated enough observations to begin making predictions. To maintain fairness, the first prediction from each classifier is excluded, as it is not based on sufficient prior data.

The threshold x, which represents the number of observations available for training each SEC, is incrementally adjusted for the ensembles $ECCE$, $ECWE$, and $ERWE$.

Additionally, the evaluation tracks the number of entities that benefit from entity-centric learning, as well as those excluded from the evaluation as the threshold x increases. While the $RMSE$ values for $ECCE$, $ECWE$, $ERWE$, and $EIGC$ are based on the same number of predictions, it is important to note that $EIGC$ has access to a larger volume of data at each point in time. This is due to the fact that $EIGC$ starts learning from the beginning of the stream, while the SECs need to collect a minimum number of observations before they can make predictions. Consequently, it is expected that $EIGC$ will show lower $RMSE$, particularly when the threshold x is small.

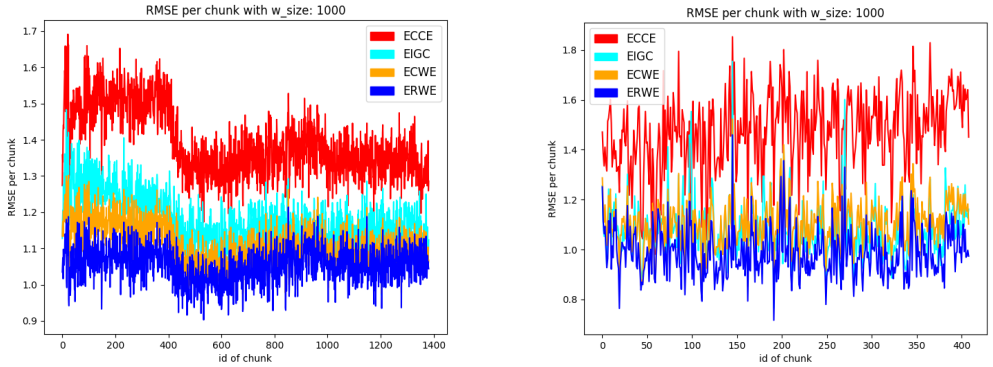

Figure 5.2.: Datasets **tools** (left) and **watches** (right): $RMSE$ on non-overlapping chunks (chunk size = 1000 predictions); lower values represent better performance. The error-weighted ensemble $ERWE$ (dark blue) shows the best performance ($EIGC$ in light blue). Figure reproduced with permission from [8] ©2019 ACM.

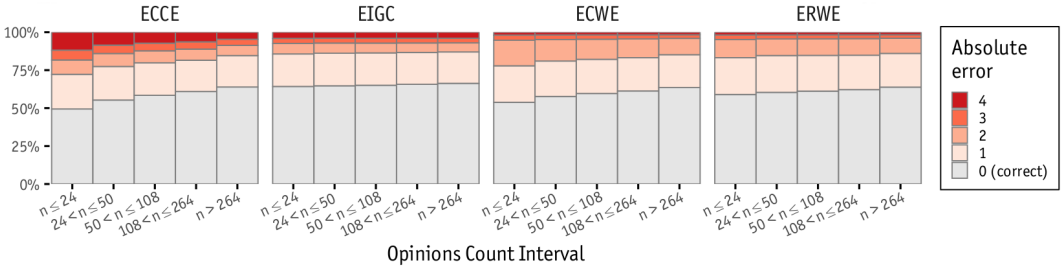

Figure 5.3.: Dataset **tools**: The "Absolute error," showing the gap between predicted and actual star ratings, is charted against the number of predictions per entity, divided into 5 bins. Lower percentages of error are more desirable, particularly for the largest errors (4-star discrepancy). Figure reproduced with permission from [8] ©2019 ACM.

5.4.2. RMSE and Number of Entities

The evaluation begins by analyzing the conventional $RMSE$ across non-overlapping chunks of data. The results, presented in Figure 5.2, compare performance on the **tools** dataset (left) and the **watches** dataset (right). For both datasets, $ECCE$

exhibits the weakest performance, whereas $ERWE$ consistently achieves the best results. On the **tools** dataset, $ECWE$ outperforms $EIGC$, although it falls slightly behind on the **watches** dataset.

Figure 5.3 offers a more detailed analysis for the **tools** dataset by grouping entities according to the number of predictions they contribute. Entities are divided into five bins, ranging from those with fewer than 24 predictions to those with over 264 predictions. Within each bin, the percentage of errors is displayed, based on the gap between the true and predicted labels, using a star-rating scale. The legend indicates the severity of the error, from 0 to 4 stars.

As shown in Figure 5.3, $ECCE$ performs poorly for entities contributing fewer predictions but demonstrates similar performance to the other models for entities with more than 264 predictions. This may explain why $ECWE$ and $ERWE$ generate fewer accurate predictions compared to $EIGC$ but still improve overall RMSE as in Figure 5.2. The entity-centric component mitigates larger errors (differences of 3 or 4 stars) more effectively than the entity-ignorant one but leads to more minor errors (1 or 2 stars difference), which is supported by Figure 5.3.

5.4.3. Impact of Entity-Length on Performance

The impact of entity-centric learning is further evaluated by analyzing the percentage of entities where $SECs$ improve predictions, varying the threshold x. The results of the smaller dataset **watches** are presented in the Appendix.

The entity-centric $RMSE$ for the **tools** dataset is shown in Figure 5.4, while Figure A.2 shows the results for **watches**.

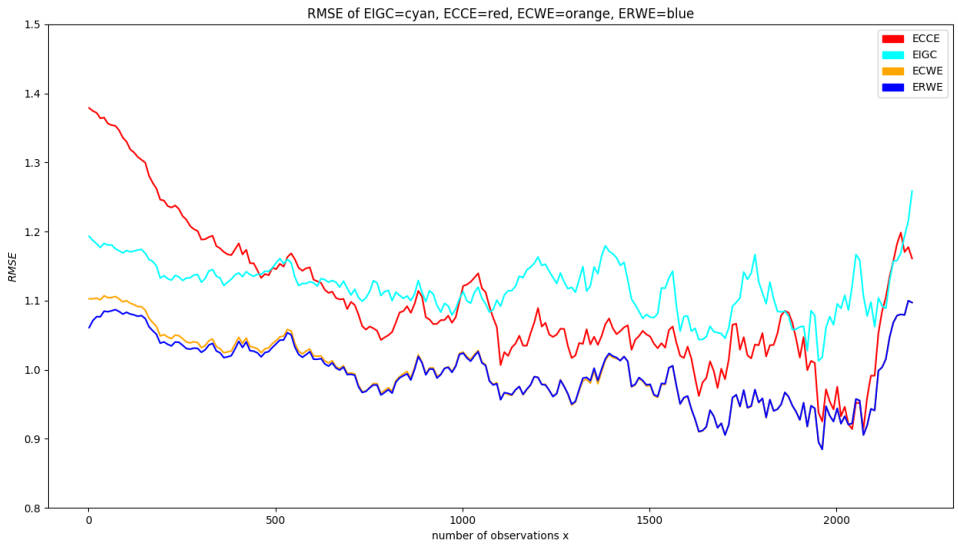

Figure 5.4.: Dataset **tools**: Entity-centric $RMSE$ given x training instances per entity; lower values indicate better performance. $ECCE$ starts to show better performance than $EIGC$ when around $x = 500$ instances are available for training per entity. Figure reproduced with permission from [8] ©2019 ACM.

The percentage of entities benefiting from entity-centric learning is displayed in the upper sections of Figures 5.5 and A.3, while the absolute number of entities is shown in the lower sections.

For the **tools** dataset (Figure 5.4), $ECCE$ begins to show a lower RMSE than $EIGC$ when approximately $x = 500$ observations per entity are available for training, and $ERWE$ shows the best overall performance. A similar trend is observed for **watches**, where $ECCE$ starts to outperform $EIGC$ at around $x = 400$ instances (Figure A.2).

The top section of Figure 5.5 indicates that for **tools**, the majority of entities benefit from $ECCE$ once $x > 700$ instances are available. For $ECWE$ and $ERWE$, more than 50% of entities show improved results compared to $EIGC$.

The intersections occur at different points because the number of wins is more restrictive than the entity-centric RMSE. An entity-centric learning approach may improve entity-centric RMSE even before these improvements are reflected for the majority of entities. For instance, consider a scenario with 5 entities: if 2 of them are perfectly predicted by $ERWE$, but for 3, the RMSE of $EIGC$ is slightly better than that of $ERWE$, the entity-centric RMSE of $ERWE$ across all 5 entities may already outperform that of $EIGC$. However, the number of wins would only be 2 out of 5, thus remaining below 50

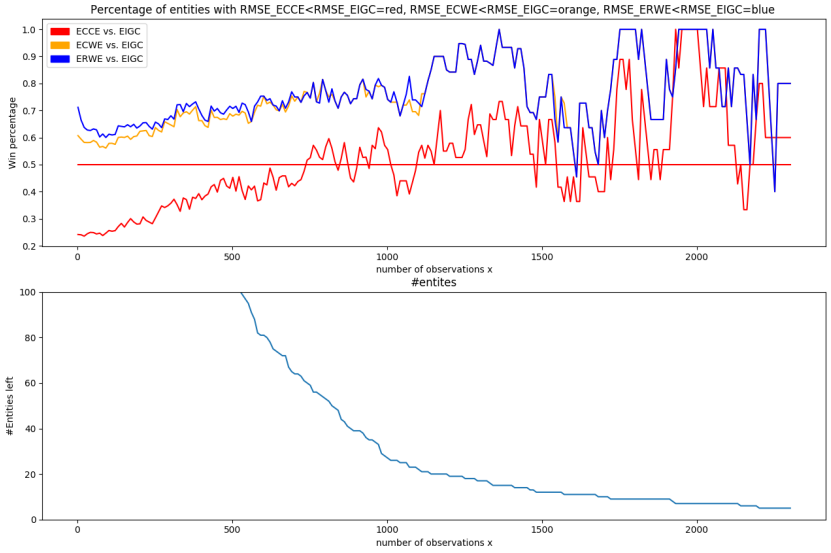

Figure 5.5.: Dataset **tools** - **Top**: Proportion of entities at threshold x where the $RMSE$ of entity-centric models surpasses that of $EIGC$: $ECCE$ vs $EIGC$ (red), $ECWE$ vs $EIGC$ (orange), and $ERWE$ vs $EIGC$ (blue). Higher Values are better. The Straight red line marks 50%.
Bottom: Count of entities still present at threshold x. Figure reproduced with permission from [8] ©2019 ACM.

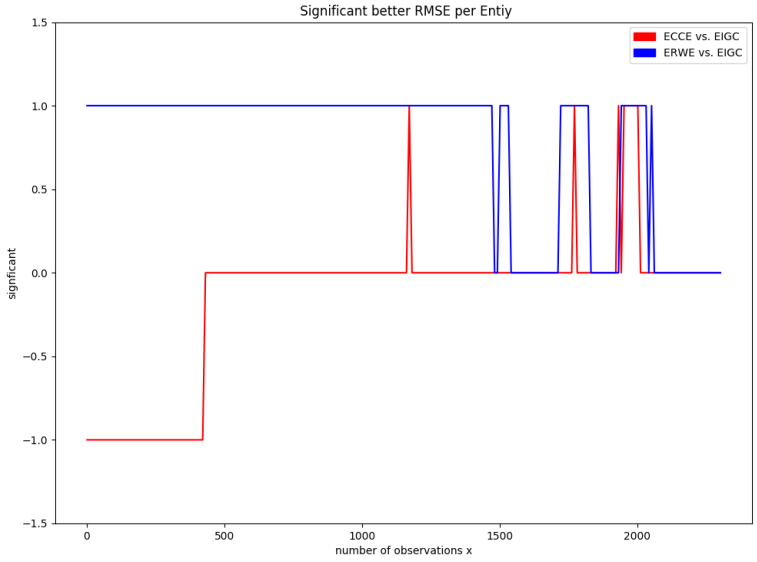

Figure 5.6.: Significance testing using the Wilcoxon signed-rank test ($p = 0.025$) comparing the $RMSE$ of $EIGC$ with $ECCE$ and $ERWE$. A value of 1 means the entity-centric ensemble outperforms $EIGC$, while -1 means $EIGC$ performs better. $ERWE$ demonstrates superior results for most x values and is never significantly worse. Figure reproduced with permission from [8] ©2019 ACM.

5.4.4. Significance Testing

Statistical significance tests were performed using the Wilcoxon signed-rank test (with $p = 0.025$) to compare the performance of $ECCE$ and $ERWE$ against $EIGC$. The results, displayed in Figure 5.6, reveal that for the `tools` dataset, $ECCE$ initially underperforms compared to $EIGC$ until approximately $x = 490$, but demonstrates improvement beyond this threshold. In contrast, $ERWE$ consistently outperforms $EIGC$ across nearly all values of x.

For the `watches` dataset, $ECCE$ does not significantly outperform $EIGC$ at any point, likely due to the smaller number of entities with a high volume of observations. However, $ERWE$ outperforms $EIGC$ for most values of x, as shown in Figure A.4.

5.4.5. Overhead of the Entity-Centric Ensembles

The memory consumption and execution time of $EIGC$ were compared against the entity-centric ensembles. Figure 5.7 presents these metrics for the `tools` dataset. While no significant difference in execution time was observed, the memory usage of the entity-centric ensembles was notably higher, largely dependent on the number of entities. On average, the entity-centric models required 6GB of memory, with a peak of 9.3GB, compared to 2GB (peaking at 2.9GB) for $EIGC$. Running the entity-centric approach on the complete `tools` dataset (w=0) even led to system crashes, which impacts the viability of entity-centric learning on data streams with many entities and will be addressed in the chapter 6.

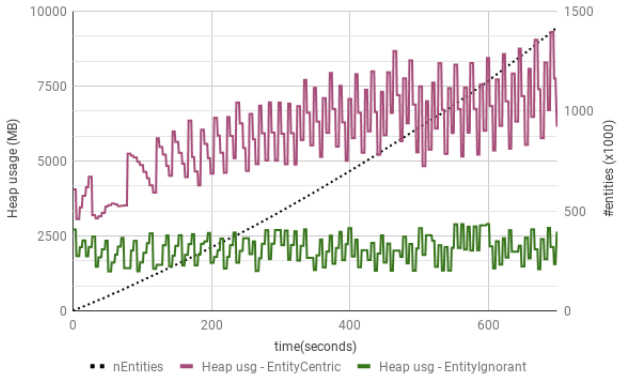

Figure 5.7.: Dataset `tools`: Comparison of memory consumption and execution time between the entity-centric ensemble and $EIGC$, along with the number of entities encountered (dotted line). The entity-centric model shows growing memory demands, but its execution time is similar to $EIGC$. Figure reproduced with permission from [8] ©2019 ACM.

5.5. Conclusion

This chapter examines how the availability of training data impacts the effectiveness of entity-centric models. It also proposes an ensemble method that combines an entity-ignorant learner, which processes the entire data stream, with a set of single-entity learners, each trained on the substream corresponding to a specific entity. The results show that entity-centric learning becomes more effective as the amount of training data per entity increases. Furthermore, classification performance in data streams is enhanced when entity-centric and entity-ignorant models are combined in an ensemble, particularly when weighted voting is employed, as in the $ERWE$ ensemble.

While $ERWE$ consistently achieves strong performance across both datasets, the simpler $ECCE$ ensemble seldom surpasses the entity-ignorant baseline, highlighting the importance of combining both entity-ignorant and entity-centric models. Notably, the $ERWE$ voting scheme maintains its advantage even for entities with relatively few observations, making entity-centric learning feasible in many scenarios. This addresses **RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model?**

An evaluation framework was developed to conduct this comparative analysis that goes beyond traditional window-based RMSE typically used in stream classification. This framework aligns entities, disregarding absolute time, and incorporates a training phase that filters out entities with very few observations. At least five entities, each with a minimum of $n = 100$ observations, are required in our experiments. This framework provides valuable insights by indicating how many entities benefit from entity-centric learning, offering a clearer understanding of its applicability to a given dataset. Once this determination has been made, further performance evaluations can be conducted by calculating the RMSE over non-overlapping chunks or rolling windows.

A major challenge for deploying entity-based learning in practice is the increasing memory demands as more entities require personalized models to be stored. The next chapter introduces two methods to reduce the memory footprint of entity-centric models while preserving the performance improvements demonstrated here.

6. Resource Management of Entity-Centric Models

This chapter covers the work published in [10] addressing:

RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model?
RQ2: How can the memory footprint of entity-centric models be reduced?

Chapter 5 demonstrated that entity-centric learning can enhance prediction performance on data streams when combined with an entity-ignorant model in an ensemble. However, one of the challenges observed was the significant increase in memory demand, resulting from creating individual prediction models for each entity; see Figure 6.1.

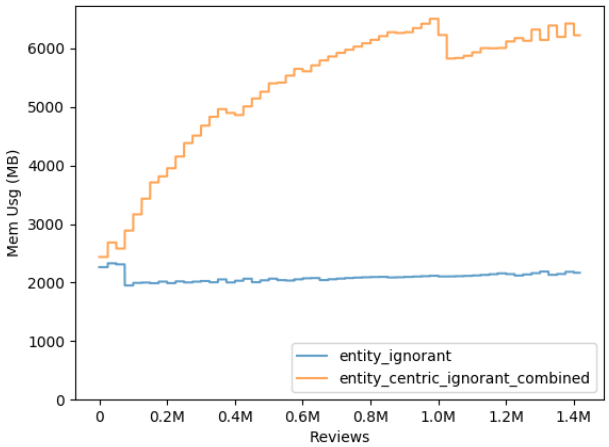

Figure 6.1.: Comparison of memory consumption between an entity-ignorant model and a combined model incorporating both entity-ignorant and entity-centric predictions, without memory optimization. The entity-ignorant model maintains a steady memory usage of approximately 2GB, while the combined model's memory consumption rises to nearly 6.5GB. Figure reproduced with permission from [10] under CC BY 4.0.

Additionally, as discussed in chapter 4, the majority of entities in the datasets analyzed have short substreams, meaning many of the entity-centric models remain inactive and do not receive new instances for prediction.

This chapter proposes two methods to address the increasing memory requirements. The first approach employs the Lossy Counting algorithm [56] to detect inactive entities and store their corresponding models in secondary memory, thereby freeing up primary memory while retaining access to the models if needed later.

The second approach integrates the methods from chapters 4 and 5, replacing the memory-intensive entity-centric MNBF with a simpler, lightweight majority-label classifier. This drastically reduces the memory consumption of the entity-centric models while still enhancing prediction performance, albeit to a lesser extent compared to using MNBFs.

The chapter begins with a review of related work, followed by a detailed explanation of the proposed memory-reduction methods in Section 6.2. The experimental setup is described in Section 6.3, with the results and conclusions presented in Sections 6.4 and 6.5. The sections in this chapter are based on their counterparts in [10].

6.1. Related Work

This section provides an overview of additional relevant literature. It begins by examining work on entity-centric learning, followed by a technique aimed at efficiently tracking frequent items in streams to optimize memory usage.

6.1.1. Learning at the Entity Level

Spitz et al. [77, 76] introduced an entity-centric method for processing news articles in streams. They constructed an entity-centric network, a graph structure that links named entities (such as locations and organizations) to sentences and documents. This graph also connects entities that co-occur within the same text, allowing for entity-centric tasks like topic exploration, as detailed in [76].

Liu and Hauskrecht [54] applied a similar entity-centric approach in the context of patient data. Initially, they developed an entity-ignorant model to capture population-wide patterns. They then introduced a *multivariate residual time series* for each patient, representing the deviation from the population model's predictions, which allowed for personalized forecasting using a multi-task Gaussian process.

6.1.2. Error-Weighted Predictions and Clustering Entities

An approach similar to the error-weighted ensemble proposed in this work comes from the time series domain, published in 2019 [69]. The authors combined entity-ignorant and entity-centric Gaussian Processes to model time series data. They found that individual entity-centric models performed poorly, but integrating them with entity-ignorant models led to significant improvements. Unlike the present work, their integration was achieved through a regression model. A related approach from the data panel field used performance-weighted ensembles based on AUC instead of RMSE [24]. Here, the weighting was based on model performance over different time steps, applied to financial data across multiple years.

Dynamic ensemble classifiers utilizing error-based weighting have also been explored in the context of data streams [70], although they did not take entity-instance relationships into account. In [50], a weighted ensemble was developed to manage concept drift, combining models based on mean squared error, but entities were not considered.

Clustering entities to enhance model performance has also been investigated. Lu et al. [55] clustered entities to develop more specialized models, while [68] proposed an algorithm to cluster evolving substreams, which could be relevant for future applications of this work.

6.1.3. Memory Efficient Item Set Mining on Data Streams

Lossy Counting, introduced by Manku et al. [56], is a technique designed to reduce memory consumption by tracking only the most frequent items in a stream, while allowing an acceptable error margin in frequency estimates. The algorithm maintains a data structure, D, that tracks recently encountered items, determining whether to retain or discard them based on their frequency.

This technique will be employed to identify entities that frequently receive new instances. It will ensure that their models remain in primary memory, while inactive entities will have their models moved to secondary storage.

The following section details how lossy counting, combined with label-only entity-centric models, is applied to reduce memory usage in entity-centric learning.

6.2. Methods For Memory Footprint Reduction

This section briefly introduces how Lossy Counting works for frequent item set mining, before showing how it can be applied to identify active and inactive entities. An alternative approach to reduce memory requirements is then described, which involves using lightweight, text-ignorant, entity-centric models.

Lossy Counting minimizes the number of retained item sets on a data stream, while ensuring that their frequency estimates remain within a user-specified error margin.

The algorithm uses a data structure, D, to store the counts of recently observed item sets. An item set remains in D as long as its frequency meets predefined criteria; otherwise, it is removed. Users specify two parameters: the minimum support threshold s and an error tolerance ϵ. The memory required by the algorithm is guaranteed to be at most $\frac{1}{\epsilon}\log(\epsilon M)$, where M is the length of the stream up to the current point. To manage memory, the stream is divided into buckets of size $w = \left\lceil \frac{1}{\epsilon} \right\rceil$, with each bucket assigned an ID starting from 1. The current bucket ID is denoted as $b_{current}$.

Each entry in D is represented by a triple $\langle e, f, \Delta \rangle$, where e denotes the item set, f is the estimated frequency of the set, and Δ is the maximum error in this estimate. When a new item set e is encountered, its frequency f is updated if the item already exists in D. Otherwise, a new entry is created with $\langle e, 1, b_{current} - 1 \rangle$. When the current bucket is filled, any item sets in D where $f + \Delta \leq b_{current}$ are deleted to conserve memory.

6.2.1. Entity Management with Lossy Counting

In this approach, instead of tracking frequent item sets, the percentage of instances in a stream corresponding to specific entities (e.g., Amazon product IDs) is monitored. At the end of each bucket, inactive (infrequent) entities are identified and removed from D. Their associated models are saved to secondary memory, freeing up primary memory resources. Models for entities still present in D remain in primary memory. If an entity stored in secondary memory reappears in the stream, its model is reloaded into primary memory for further use.

To implement this mechanism, a secondary data structure L is introduced to track whether an entity has appeared previously. If an entity is found in L but not in D, this indicates that its model needs to be retrieved from secondary storage.

6.2.2. Memory Reduction through Text-Ignorant Models

The second strategy for reducing memory consumption is motivated by the work presented in chapter 4, where simplified entity-centric models based solely on labels were introduced, disregarding features like review texts. This label-only method improved prediction accuracy for some entities, although the entity-ignorant MNBF model generally demonstrated better performance.

In chapter 5, it was shown that combining entity-ignorant MNBF with entity-centric MNBF models enhanced prediction outcomes. Expanding on this, lightweight, label-only, entity-centric models are combined with an entity-ignorant MNBF. These label-only models significantly reduce memory requirements while maintaining predictive performance. The most effective label-only model from chapter 4, predicts the most frequently observed label for each entity. This majority-label model is applied due to its simplicity and efficiency in reducing memory overhead.

6.3. Experiments

In the experiments,[1] a comparison of the memory usage between the two strategies outlined in section 6.2 was conducted using the `tools` and `watches` datasets described in section 4.5.1. For the `tools` dataset, entities with fewer than 10 reviews were excluded to align with previous experiments that lacked memory management. This exclusion was necessary as earlier tests of entity-centric learning without memory optimization led to RAM exhaustion and system crashes.

Additionally, a subset of the Yelp dataset,[2] specifically focusing on reviews related to 'bars and restaurants,' was used. Due to its large size, the scope was narrowed to include reviews from five cities (Toronto, Las Vegas, Phoenix, Montréal, and Calgary) to enable testing of memory management strategies while maintaining performance comparisons with the full dataset. The full Yelp dataset consists of 4,198,061 reviews and 59,372 entities.

An overview of the datasets used in our experiments is provided in Table 6.1.

Name	#Ent.	#Inst.	#Feat.	#Classes
tools	33,990	1,417,499	10,000	5
watches	78,220	487,907	10,000	5
bars5	25,110	2,224,710	10,000	5
barsFull	59,372	4,198,061	10,000	5

Table 6.1.: Overview of datasets: `tools`, `watches`, `bars5`, and `barsFull`. Table reproduced from [10].

6.3.1. Evaluation

A prequential evaluation setup is employed, where each new observation is first used to predict the label, followed by updating both the entity-ignorant and entity-centric models with the correct label.

The primary aim of this evaluation is to assess how effectively the two proposed methods reduce memory usage in entity-centric models. Memory consumption is monitored over time and compared against two baselines: an entity-ignorant

[1] Code is available at: `https://github.com/m-vishnu/entity-memory-management`
[2] The dataset is available at: `https://www.yelp.com/dataset`

model and a combined entity-ignorant and entity-centric model without memory management. For the Lossy Counting method, the parameter ϵ is set to 0.001.

A secondary objective is to determine whether replacing entity-centric MNBFs with majority-label classifiers continues to provide performance improvements over the entity-ignorant model, and how this approach compares to using a separate MNBF for each entity. Although memory reduction is the main focus, it is also important to ensure that simplifying the model does not result in a substantial drop in performance.

Performance is evaluated using $RMSE$, as outlined in Section 2.2. RMSE is selected because it reflects the ordinal nature of the labels and assigns greater penalties to predictions that deviate significantly from the actual label.

RMSE is calculated in non-overlapping chunks of 10,000 observations, with the first prediction for each entity excluded, as at least one observation is required to initialize an entity-specific model.

6.4. Results

In this section, the predictive performance of entity-centric models utilizing MNBFs is compared to that of models based on the majority label for each entity. The latter part of the section focuses on the primary objective of this research: reducing memory usage.

To clarify the results, the ensemble acronyms are reintroduced. The evaluation compares the performance of a model that uses only entity-centric predictions (ECCE) against a baseline model relying solely on entity-ignorant predictions (EIGC), a model that averages both predictions (ECWE), and a model that incorporates error weighting (ERWE).

6.4.1. Entity-Centric MNBF vs. Majority-Label

The central question is whether majority-label classifiers can replace entity-centric MNBFs while still improving RMSE compared to an entity-ignorant MNBF. Results across all datasets confirm that this is achievable with the error-weighted ensemble (ERWE). The majority-label approach performs only marginally worse than MNBFs, as illustrated in Figures 6.2 and 6.3. On the `watches` dataset, the majority-label method even slightly outperforms MNBFs. Similarly, on the `barsFull` dataset, the majority-label classifier performs effectively in an error-weighted ensemble, resulting in an overall performance improvement. In this dataset, which experiences periodic bursts of new entity arrivals (see Figure 6.5), a temporary increase in RMSE above the entity-ignorant model (EIGC) is observed when many entities arrive simultaneously, but this effect diminishes quickly.

The ensemble relying solely on entity-centric models (ECCE) consistently performs worse than all other methods, consistent with findings of chapter 5. This again emphasizes the necessity of integrating the entity-centric models with a strong entity-ignorant model.

Importantly, the majority-label classifier demonstrates that it can compete with the more complex MNBF model while using considerably less memory, as further explored in the following section.

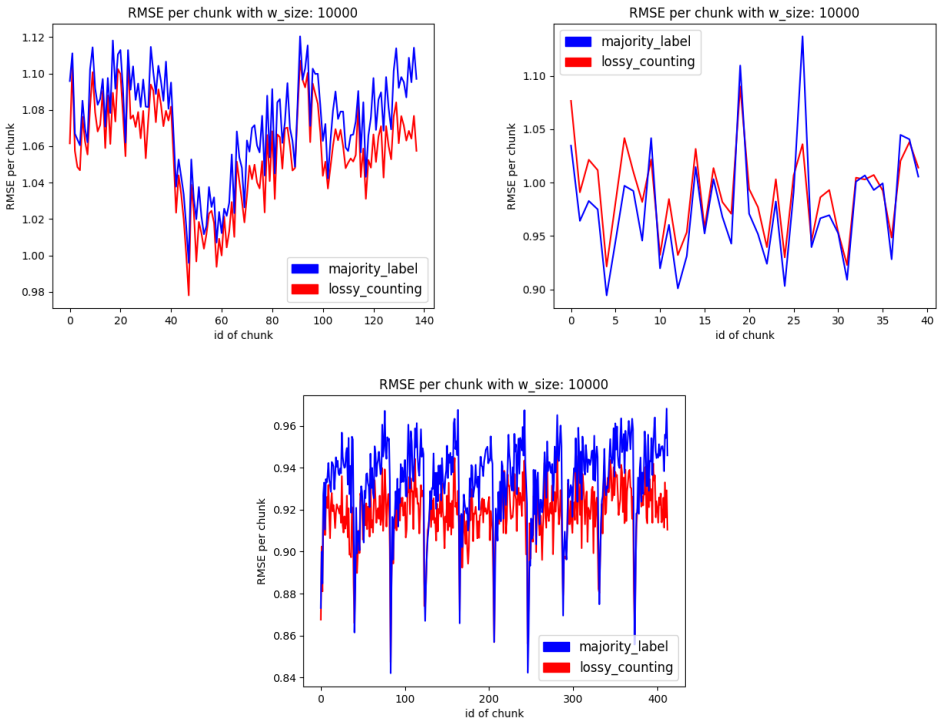

Figure 6.2.: RMSE comparison between entity-centric MNBFs using Lossy Counting and majority-label models on 10k-review non-overlapping chunks. **Top Left**: `tools`; **Top Right**: `watches`. **Bottom:** `barsFull`. RMSE values are close, with Lossy Counting slightly outperforming on `tools` and `barsFull` and majority-label models performing better on `watches`. Figures reproduced with permission from [10] under CC BY 4.0.

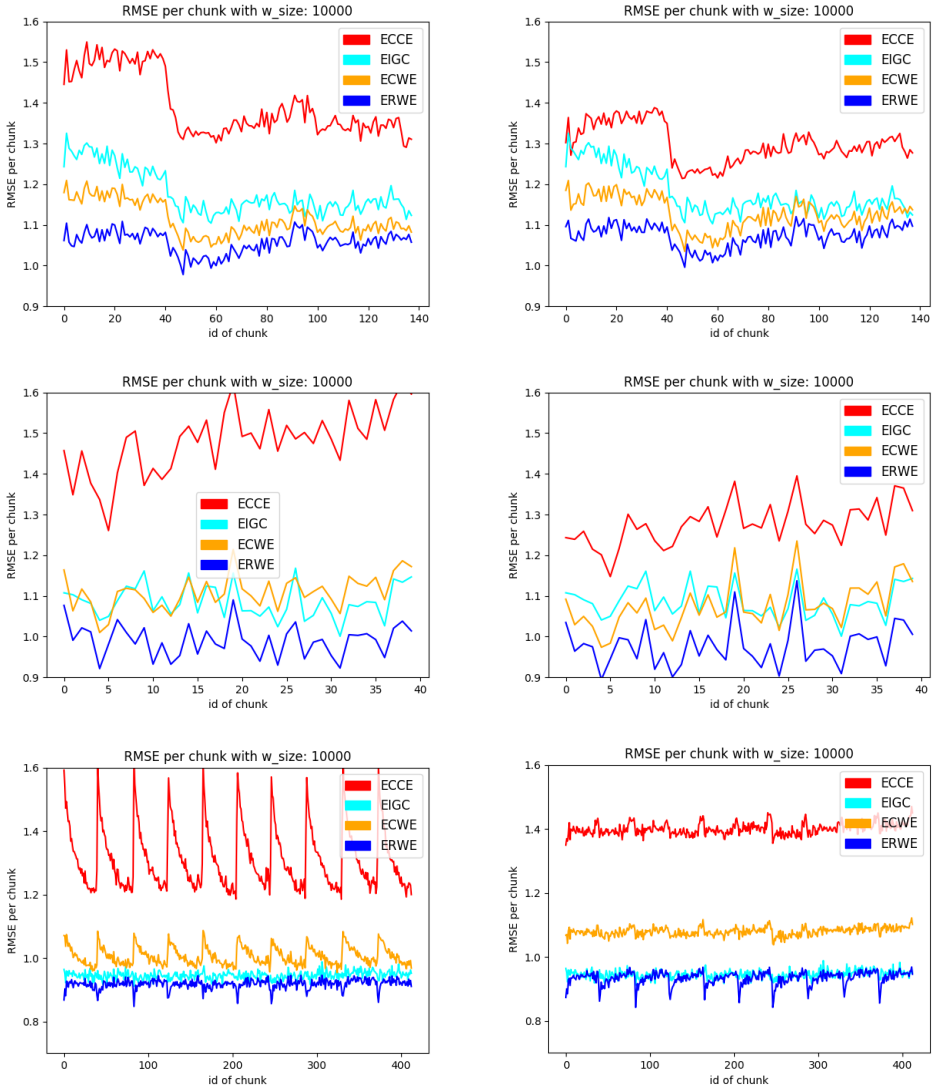

Figure 6.3.: RMSE comparison across models: entity-centric predictions (ECCE), entity-ignorant predictions (EIGC), a combined model averaging both (ECWE), and an error-weighted ensemble (ERWE). RMSE is measured in non-overlapping chunks of 10k reviews. Top row: **`tools`**; middle row: **`watches`**; bottom row: **`barsFull`**. Left panel: results with MNBFs; right panel: results with majority-label classifiers. The error-weighted ensemble consistently outperforms the entity-ignorant model. Figure reproduced with permission from [10] under CC BY 4.0.

6.4.2. Memory Footprint Comparison

Both methods lead to significant reductions in primary memory usage, as shown in Figure 6.4. The majority-label approach has the smallest memory footprint, nearly matching the memory usage of the entity-ignorant model. With Lossy Counting, the number of entities stored in memory drops quickly and remains nearly constant thereafter, although memory usage continues to increase slightly. This increase is due to the L data structure, which tracks all previously seen entities, even if they are no longer in primary memory.

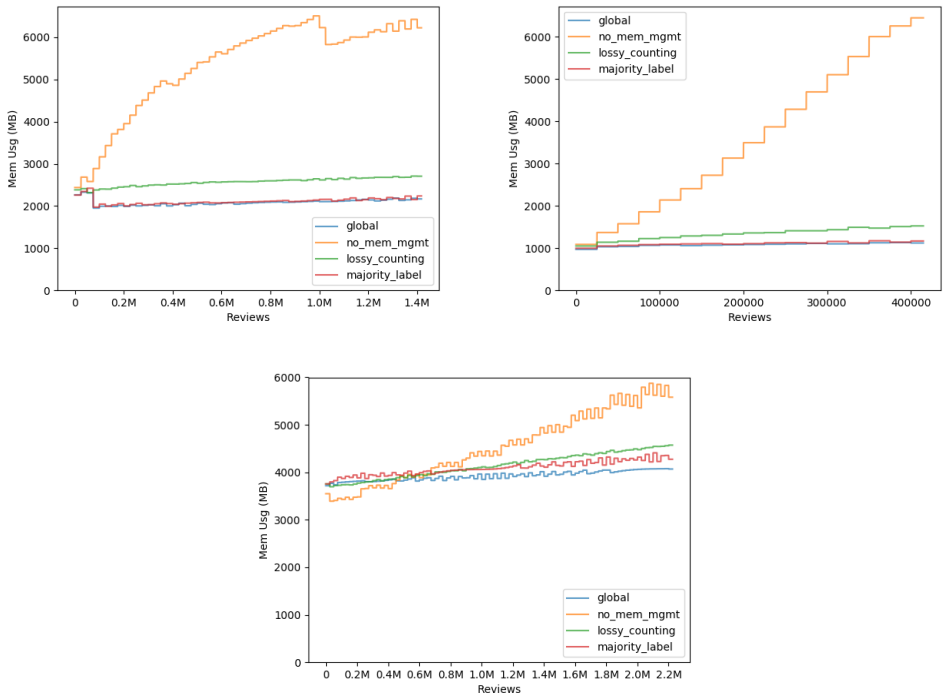

Figure 6.4.: Comparison of primary memory usage for different models: entity-ignorant model (blue), combined model without memory management (orange), combined model with Lossy Counting (green), and combined model using majority-labels (red). **Top Left**: `tools`; **Top Right**: `watches`. **Bottom:** `bars5`. The entity-ignorant model uses the least memory, followed by the combined model with majority-labels. Memory usage for Lossy Counting grows slightly, while the model without memory management shows a sharp increase. Figures reproduced with permission from [10] under CC BY 4.0.

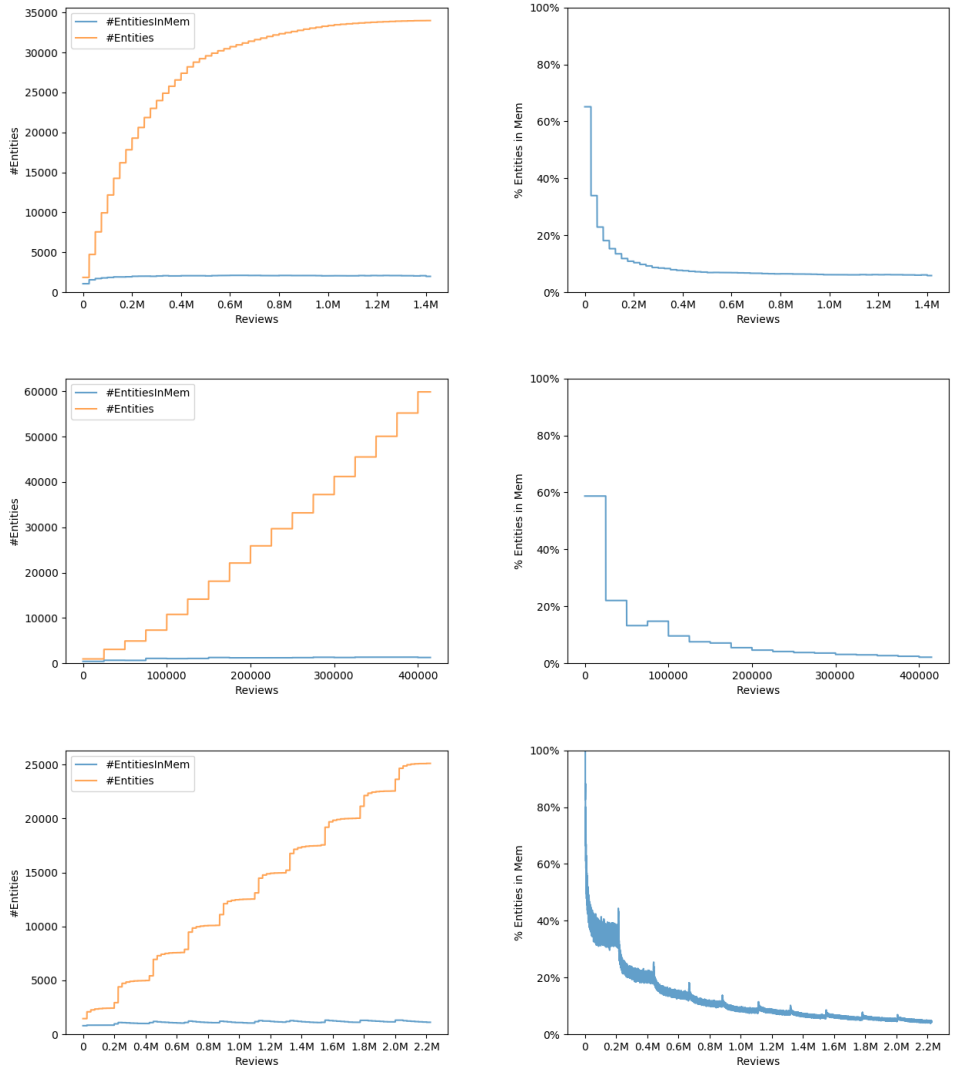

Figure 6.5.: Lossy Counting: comparison between the number of entities observed and those retained in memory, along with the percentage of entities kept over time. **`tools`** (top), **`watches`** (middle), and **`bars5`** (bottom). While the number of seen entities increases rapidly, the count of entities in memory stays steady and significantly lower (**left**). The percentage of entities in memory drops sharply early on, followed by a more gradual decline (**right**). Figure reproduced with permission from [10] under CC BY 4.0.

6.4.3. Discussion

Both memory-reduction approaches effectively reduce memory consumption while preserving strong predictive performance. The results indicate that the majority-label method offers greater advantages in memory savings, with only a slight increase in RMSE when compared to the Lossy Counting method on the **tools** and **barsFull** dataset. Additionally, it outperforms the Lossy Counting approach on the **watches** dataset. This discrepancy in performance could be attributed to the high number of entities with relatively few observations in these datasets, where most of the predictive information is stored in the label distribution rather than word counts. This effect is particularly noticeable on **watches**, where the average entity length is considerably shorter (6.2 in **watches** compared to 41.7 in **tools**).

However, these findings are specific to the datasets used, and further research is needed to determine whether these trends hold across datasets with different entity characteristics.

Although it is theoretically possible to combine the two approaches—storing majority-label models in secondary memory using the Lossy Counting method—the potential memory savings would likely be minimal. This is because the majority-label approach already has a memory footprint close to that of the entity-ignorant model.

6.5. Conclusion

The previous chapter identified the increasing memory demands of entity-centric models as a major challenge for implementing entity-centric learning in data streams, as the full **tools** dataset caused system crashes due to insufficient primary memory.

This chapter introduces two methods aimed at reducing memory requirements. The first method employs Lossy Counting to detect inactive entities, whose models can then be transferred to secondary memory and retrieved only when new observations for those entities are received. The second method replaces entity-centric MNBFs (Multinomial Naive Bayes with Fading) with a simpler majority-label classifier, which significantly lowers memory consumption.

Both approaches effectively reduce memory usage for entity-centric models, maintaining nearly constant memory consumption while still surpassing the performance of the entity-ignorant model. The majority-label approach is the most efficient in terms of memory usage, requiring approximately 2GB, compared to 2.6GB for the Lossy Counting method and 6.5GB for the previous memory-intensive approach on the **tools** dataset. Across all datasets, the majority-label approach closely matches the memory requirements of the entity-ignorant model, indicating that prediction quality can be enhanced with minimal additional memory cost.

On the **tools** and **barsFull** datasets, the majority-label approach results in a slight reduction in predictive performance compared to the entity-centric MNBFs. However, this decline is minimal, and in the **watches** dataset, the majority-label classifier even slightly outperforms the MNBFs. This suggests that combining a complex entity-ignorant model trained on a large amount of data with simple entity-centric models trained on limited data can improve prediction quality.

It is hypothesized that this outcome is due to the nature of the datasets, where many entities have few observations, making it difficult to train MNBFs effectively. Datasets with many short-lived entities are common, which makes this finding significant for future research. This also explains why the performance discrepancy is more pronounced on the **barsFull** dataset, as it contains fewer short-lived entities,

allowing MNBFs to perform better at the entity level.

Overall, both methods present promising solutions for making entity-centric learning feasible in real-world applications where memory constraints are a concern.

The next chapter goes into additional reflections of Part I before it is concluded in chapter 8.

7. Additional Reflections on Entity-Centric Learning

This chapter contains additional reflections on how entity-centric learning can be realized efficiently and how the results presented in this thesis might warrant a new stream evaluation metric.

7.1. Efficient Entity-Model Management using Databases and Deletion

Chapter 6 introduced two methods for managing the growing memory demands of an entity-centric learning approach, where new models have to be created and stored for each entity encountered in a data stream. The first presented option stores complex entity-centric models of inactive entities in secondary memory to retrieve them later in case new instances regarding those entities appear. The second approach replaces complex models with much simpler entity-centric models that have a negligible impact on the required memory.

The analysis of the presented datasets showed that the majority of entities have very short substreams, which means they receive few instances. In such circumstances, it might be more resource-efficient to delete entity-centric models flagged for storage rather than actually moving them to secondary memory for later retrieval, as long as the training data is stored so that they can be reconstructed later. For example, in many scenarios, user data has to be kept for multiple years because of legal requirements, but this does not extend to models of inactive users. Furthermore, storing all available information in databases and data warehouses is very common in many domains, like eCommerce, so retrieving the necessary training data to reconstruct a model should be possible.

This can also be realized as an additional step so that the secondary memory acts like a buffer:

1. Inactive entities are identified with lossy counting.
2. Their models are moved to secondary storage and kept there for a defined time period T so they can quickly be recovered in case new instances arrive
3. After T has passed, delete the models.
4. If later new instances referring to such an entity arrive again, reconstruct the entity-centric model from the stored training data.

A more radical approach would be to delete inactive models for good and to start a new entity-centric model once these entities become active again. Such an approach could be applied where the benefits of storing all potential training data are outweighed by the cost of doing so.

7.2. A New Performance Metric for Data Streams?

In chapter 4, we used κ^+ [13] as one of the evaluation metrics, which compares the performance of a classifier against a model that propagates the last observed label. The assumption was that entities that benefit from entity-centric learning the most would probably show different label distributions; therefore, propagating an entity's last observed label to the next arriving instance of that entity made intuitive sense.

The authors of [13] developed an additional metric for performance evaluation on data streams [88] κ_m which compares a classifier against a simple model that assigns the most observed label to an instance - a majority-label classifier. The results from chapter 6 indicate that, at least in some scenarios, the combination of an entity-ignorant classifier as well as a majority-label classifier leads to improvements outperforming each of the individual models. These improvements were almost for 'free' as the additional computational and memory requirements were negligible. The employed performance metric was $RMSE$ and not *kappa*, which motivates future research that investigates if this is also true for the κ metric in the case of ordinal class labels. It also motivates non-entity-centric experiments where two entity-ignorant models' predictions are combined, one of them being a majority-label classifier, and comparing the performance of the combined approach against each model individually. Such experiments could give rise to a new (entity-centric) performance evaluation metric on data streams, which would evaluate if a model is justified in not taking the for free improvements of combining with a (entity-centric) majority-label classifier.

If $P_{combined}$ is the performance of the combined model and P_{single} is the performance of the original model then $\kappa_{combined}$ could be defined as:

$$\kappa_{combined} = \frac{P_{combined} - P_{single}}{1 - P_{single}} \tag{7.1}$$

Values around 0 would indicate no advantage in combining with a majority-label classifier, negative values would show that combining is actually detrimental and positive values would show that it would beneficial. This metric could be used in conjunction with the metrics proposed in [13] and [88].

8. Entity-Centric Learning on Data Streams: Discussion and Conclusion

In Part I of this thesis, the entity-instance relationship within the data space of data streams is examined, demonstrating its potential to enhance the quality of predictions for opinionated documents.

RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model? Chapter 4 emphasizes that entity-centric models, even without text-awareness, yield superior predictions for a subset of entities in the Amazon review dataset. Further analysis in chapters 5 and 6 show that combining entity-centric and entity-ignorant models results in improved predictive performance compared to using only entity-ignorant models. However, the increasing number of entities presents a significant challenge for memory management, making entity-centric approaches impractical without proper handling.

RQ2: How can the memory footprint of entity-centric models be reduced? To address memory constraints, chapter 6 introduces two strategies. The first uses lossy-counting to monitor entities likely to receive new instances, archiving others in secondary memory for potential retrieval. The second combines the simplicity of entity-centric models from chapter 4 with the predictive power of entity-ignorant models. Both approaches effectively reduce memory consumption while maintaining predictive quality, confirming the viability of entity-centric learning.

Additionally, frameworks are developed to facilitate entity-centric learning, resource management and evaluation schemes that account for the length of entity substreams, i.e., the available training data. These frameworks enable:

- Investigation of the percentage of entities benefiting from entity-centric learning,
- Analysis of the minimum number of instances required per entity to make entity-centric learning advantageous,
- Comparison of ensemble methods, combining entity-centric and entity-ignorant predictions, against a single entity-ignorant model.
- Monitoring and analysis of memory requirements of entity-centric models in relation to the number of encountered entities.

Overall, it is demonstrated that entity-centric learning in data streams can offer cost-effective performance improvements, although open challenges remain. These limitations, open challenges and future work are discussed below.

8.1. Limitations

The presented work has limitations, which we will briefly list here for clarity and revisit in the following section on open questions and future directions.

- In all three studies, entity-centric models are built for each entity, regardless of the entity's length or characteristics. However, the results indicate that entity-centric models are particularly effective for long entities that do not conform to the overall trend captured by the entity-ignorant model.
- Very short entities are often excluded to ensure a fair comparison between models. They represent a substantial portion of the dataset, which leaves room for future work especially focused on the handling of very short entities.
- The datasets and models selected are specifically focused on opinionated documents from particular domains. However, as instance-entity relationships are common across various other domains, our approach has the potential to be adapted and expanded to new areas in future work.
- The ensemble-based approach utilizes the same text-aware model (MNBF) for both entity-centric and entity-ignorant models, which has proven effective in the current setup. Exploring different models at the entity and global level could offer further performance improvements and adaptability.

8.2. Open Questions and Future Work

The open challenges, grouped by the chapter in which they arise, are discussed in this section.

Chapter 4 - Entity-Centric Learning without Features: This chapter shows that while some entities benefit from exclusive entity-centric learning, the majority do not. A key challenge is determining whether identifiable characteristics can predict when entity-centric learning will be effective. For example, do these entities have distinctive label or feature distributions? Are these distributions different from those of the general population? Preliminary experiments suggest that entity-centric learning is most effective for entities with labels that differ from the overall dataset, but this observation was based on limited visual inspection, with follow-up work being inconclusive. Future work will involve quantifying these differences and developing predictive models to identify when entity-centric learning is likely to be beneficial or detrimental.

Chapter 5 - Entity-Centric Ensemble: In this chapter, ensembles combining predictions from both entity-centric and entity-ignorant models are used, with a separate entity-centric model built for each entity. If characteristics can indeed predict the viability of entity-centric approaches, an avenue for future exploration would be to build entity-centric models only for entities where the approach is promising, combining them with entity-ignorant models.

Another challenge arises from the limited training data available for each entity-centric model, which increases susceptibility to outliers. A potential solution is to group similar entities into clusters and build models for these clusters. While

stream clustering introduces its own difficulties, some methods, such as [44], may be applicable.

Future work will also involve exploring alternative base models beyond MNBF and applying these models to datasets from various domains, rather than relying solely on opinionated review data. Additionally, modifications to the evaluation scheme could enable the use of more of the available data, as the current approach requires a fixed amount of training and test data per entity, which limits the usable dataset.

Chapter 6 - Resource Management of Entity-Centric Models: The balance between model complexity, prediction quality, and memory requirements warrants further investigation. The majority-label classifier, used as an entity-centric model in chapter 4, performed best according to κ^+, but when using RMSE, a simple moving-average performed better. Revisiting some experiments with alternative text-unaware, entity-centric models is planned. Furthermore, chapter 7 motivates experiments that favor the deletion of inactive entity-centric models over storage, as well as a general investigation of whether an entity-ignorant model combined with a majority-classifier could serve as a new performance baseline in certain stream mining scenarios.

This concludes the first part of the thesis. The second part shifts the focus from the data space to the feature space, exploring how active feature acquisition and imputation methods can address missing feature values in data streams.

Part II.

Dealing with the Feature Space in Stream Mining

9. Motivation & Background - Active Feature Acquisition on Data Streams

The previous chapters discussed how entity-centric models can improve prediction quality and how these models can be realized efficiently on a data stream. All the aforementioned work assumed that instances arriving on the data stream are feature-complete and therefore easy to work with, but this is not always true. Sensors in a smart home can break, users can skip questions in a personal health app, and doctors do not send all the arriving patients to all the available tests, but only a selected few.

Let us consider an industrial plant where we have multiple sensors that continuously record humidity, pressure, and temperature. The missingness is classified according to [72] into three categories.

Missing Completely At Random (MCAR): Data is considered to be missing completely at random if the mechanism leading to a missing feature value neither depends on observable features nor on unrecorded additional information regarding the instance or the feature value itself. This can be simulated by randomly deleting feature values. An example of MCAR would be if a temperature sensor randomly fails to record data at certain times due to occasional power glitches. The missing readings are not related to the actual temperature, the location of the sensor, or any other external factor. The failure happens randomly, making the missing data unrelated to both observed and unobserved variables.

Missing At Random (MAR): In the case of MAR the missingness of a feature depends on observable data. An example of Missing at Random (MAR) using sensors would be if a humidity sensor is more likely to fail and not record data when the temperature is particularly high. In this case, the missing humidity readings are not random but depend on an observable feature (temperature).

Missing Not At Random (MNAR): MNAR concerns scenarios that neither fall into the MCAR nor MAR category. This entails situations where the missingness of a feature can be explained by the feature value itself, e.g. if a pressure sensor malfunctions under very high pressure. Another form of MNAR is, when the missingness of a feature value is dependent on unrecorded additional information, for instance, if a temperature sensor tends to miss recordings during maintenance periods, but the timing of these maintenance activities is not logged in the dataset.

In a static setting, it is custom to deal with missing data during preprocessing, where missing data is traditionally addressed in three ways [72].

Deletion: The most obvious way to handle missing values is to delete all instances with missing values from our dataset, which is also called complete-case analysis. If only certain features are necessary for a task, then one can also delete instances

that have these features missing while other features are ignored. This is called available-case analysis. Alternatively, it can also make sense to delete certain features that express a high degree of missingness. This method limits the amount of available training data, which can affect how representative our training data is as well as our model performance. Considering available-case analysis it also has the disadvantage that tasks that use different sets of features might not be comparable as they will have sampled from different populations. In the case of deleting whole features, there is also a chance that we ignore predictive features which in turn can negatively impact the performance of models, that will be trained on the preprocessed data. The advantages of deletion are that it is easy and in case of MCAR missingness it would not introduce biases into our dataset as long as the total size after deletion is still big enough to be representative.

Single Imputation: In the case of single imputation, missing values of a feature are imputed in a way that solely relies on other values expressed by that single feature. Prominent examples are the use of statistics like mean and mode or linear interpolation. The latter is useful when we have temporal data with missing values. Another prominent method in the case of temporal data is to carry forward the last observed value. Single imputation has the advantages that we don't throw away any data and that the imputation can be conducted quickly as the used methods and statistics can be computed rather quickly. The main disadvantages are that single imputation will inevitably introduce biases into our data, and that we ignore the other featurest that were available for an instance and might help to give a more unbiased estimate of a missing feature.

Model-Based Imputation: Model-based imputation entails multiple types of methods, which, for the most part, consider multiple features at the same time when imputing a missing value. Often an iterative strategy is employed where firstly all missing values are imputed using single imputation, which then enables the training of a model to predict a target feature using the other features. When training the model we only consider instances where the target feature was not missing. Missing values of the target feature can now be imputed by the aforementioned model. This is usually done one feature at a time and then repeated until the imputed values stabilize. Popular models are regression models, nearest-neighbor approaches, and random forests. Model-based approaches have the disadvantage that the predicted values can be unreliable if the features used for making the prediction are independent of the target feature. They are also usually much slower than single imputation due to the complexity of the predictive model and the iterative nature of the approach. Still, they usually introduce less biases into the dataset than single imputation and also retain the size of the original dataset.

We can see, that each of these methods has certain advantages but also disadvantages like reducing the sample size that is available for training or introducing unwanted biases. In certain scenarios, it is possible to acquire missing features for a cost. For instance, asking a study participant to answer a question that they have missed or running a lab test on a material to figure out a missing property. In such scenarios, one of the most pressing questions is:

For which instances should we acquire which features under budget constraints?

The research field that addresses this question is called Active Feature Acquisition (AFA) which is part of the bigger and more well-known field of Active Learning (AL).

In static AL, we traditionally face the situation that labels and not features are missing for the vast majority of the instances. Settles [74] describes an iterative approach where first a model is trained on the small pool of labeled instances. Afterward, the model is exploited to pick instances from the big pool of unlabelled data for labeling. Next, an oracle labels the chosen data, and the newly labeled instances are added to the pool of labeled data. The process starts again with the updated pools until a stopping criterion is met, see Figure 9.1. The most common stopping criterion is that the budget available for labeling has been expended.

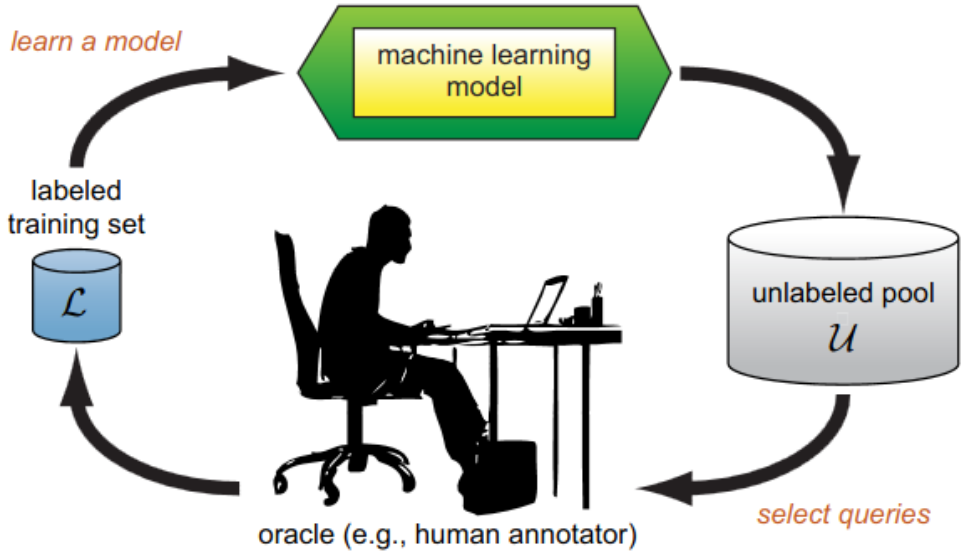

Figure 9.1.: Figure of Pool-Based AL from Settles. Reproduced with permission from [74]

In a static AFA scenario, we usually see a similar loop, where we have a pool of feature-incomplete instances from which we pick instances for feature acquisition and subsequently retrain a predictive model. Static approaches differ on how many features are bought per instance, in [59] the authors suggest that all missing features are purchased for a selected instance, irrespective of how much the missing features contribute to the machine learning task at hand and also irrespective of the features costs. A different approach is presented in [71] where the authors suggest testing every possible feature set combination but realize that this is computationally infeasible, so they rely on random sampling or a confidence-based approach to choose for which instances all possible acquisition scenarios are generated. In contrast to the aforementioned methods [20] presents a feature selection method that iteratively selects features for acquisition and then acquires the selected features for all instances where the respective features are missing. All three approaches share the commonality that they use a static test set to evaluate their newly trained models, which guides the selection process. They also have in common that feature values that have not been acquired either necessitate imputation of the values or to ignore the corresponding feature altogether during the learning phase of the models.

9.1. Challenges for AFA on Data Streams

Retaining a hold-out set for guiding the selection process and being able to consider all instances and acquisition-set options is unrealistic in a data stream. Facing concept drift, a hold-out set would have to change, to still be representative. Furthermore, we often face the scenario in streaming environments where we have a continuous influx of new instances, and decisions have to be made close to the arrival time of an instance. This means we cannot wait with our acquisition decisions until we have seen the whole data stream but we have to find new mechanisms that allow us to make decisions much quicker. This decision making process also has to consider the budget and how budget is modeled in a data stream. In contrast to a static setting the budget cannot be defined as finite because a stream is considered to be endless which would mean that once a finite budget has been expended no more acquisitions could be made at all. This would be especially detrimental in case of concept drift because the drift of a feature with high missingness might be detected late or not at all.

The following sections describe our work in the field of AFA on data streams which addresses these challenges and successively loosens the constraints we have built into our frameworks to make the different aspects of the decision-making process more controllable. Our presented work address the following research questions of this dissertation:

- RQ3: How can Active Feature Acquisition be realized in a data stream setting?
- RQ4: How can varying feature costs be considered during AFA on streams?

RQ3 is explored in Chapters 10, 11, and 12, while RQ4 is addressed in Chapters 11 and 12. Chapter 10 outlines our initial work, where we assume equal feature costs and limit acquisition to one missing feature per instance. Chapter 11 expands on this by considering larger acquisition sets and incorporating varying feature costs. Finally, Chapter 12 demonstrates how combining AFA with strategic imputation can reduce costs while maintaining performance levels similar to an AFA-only approach. Part II ends with additional reflections on stream-based AFA in chapter 13 and a conclusion of the part in chapter 14.

10. Active Feature Acquisition on Data Streams under Feature Drift

This chapter is based on the work presented in [9], where the goal was to build one of the first AFA methods that can be applied to data streams and that addresses some of the challenges mentioned earlier. It addresses:

RQ3: How can Active Feature Acquisition be realized in a data stream setting?

The primary challenge encountered was how to model budget expenditure in a data stream environment, where potentially unlimited instances could arrive. This problem was addressed by drawing inspiration from a study on budgeting for active learning on data streams [41], which introduced an Incremental Percentile Filter (IPF) to manage missing label acquisition.

In this context, two key assumptions are made: first, that all features incur the same cost, and second, that at most one missing feature can be acquired per instance. The first assumption allows classification performance to serve as a proxy for evaluating Active Feature Acquisition (AFA) strategies, as feature cost does not influence the decision. The second assumption simplifies the budgeting process, enabling the use of a relative budgeting approach without the need to account for absolute feature costs.

The following sections will cover related work, introduce the proposed methods, describe the experimental setup, and conclude with a discussion of the results and final remarks.

10.1. Related work

This section is based on the related work section presented in [9]. The concepts of active feature acquisition (AFA) and active feature selection (AFS) often overlap, and many papers do not strictly distinguish between the two. AFA focuses on completing feature-incomplete instances to boost performance, whereas AFS deals with selecting specific features from feature-complete instances to reduce dimensionality without compromising model accuracy. Although these areas differ in their primary objectives, they share common methodologies in identifying the most relevant features.

In the static context, Huang et al. [37] propose a matrix completion-based AFA method. This approach minimizes both classification error and matrix reconstruction error through accelerated proximal gradient descent, allowing the informativeness of unknown feature values to fluctuate across iterations. By dividing the informativeness value by the cost of feature acquisition, the method incorporates basic cost considerations. Tests on six datasets demonstrated consistent performance improvements, but the method is not designed for streaming data.

Saar-Tsechansky et al. [71] present an alternative AFA method called Sampled Expected Utility, which estimates the utility of feature acquisition based on expected acquisition values and their impact on model performance. To compute this utility, the

method assesses both the likelihood of a feature value and the resulting performance gain from adding that feature. Although highly effective in static settings, the method's complexity limits its application in stream environments.

Melville et al. [59] offer another AFA method designed for partially completing training data in a pool-based setting. By identifying misclassified, incomplete instances and acquiring their missing features, the method iteratively builds a model that achieves approximately 17

DesJardins et al. [20] develop a confidence-based approach to AFA, which involves training successive models on increasingly larger feature sets. Starting with zero-cost features, the method acquires additional features for uncertain instances until no more features can be added or all uncertain instances are resolved. This method, like the others, is designed for static use cases.

In the context of streaming data, Yuan et al. [85] introduce a batch-based AFS method. Their work ranks features based on how well they separate class labels, using metrics such as Average Euclidean Distance. While their goal is to reduce dimensionality to enhance classifier performance, this method is adapted here to acquire missing features that would otherwise need to be imputed.

To manage feature acquisition budgets in stream environments, the approach of Kottke et al. [42] is employed. Their method uses an Incremental Percentile Filter (IPF) that maintains a sorted list of usefulness values for label acquisition decisions. When a new label's usefulness score exceeds the top percentile, the label is acquired. This dynamic approach allows budget management to adapt to concept drift, ensuring acquisition decisions remain efficient over time. In the presented method, feature scores are presented to the IPF instead of label scores.

10.2. Methods

This section outlines the methods developed to implement budgeting for Active Feature Acquisition (AFA) on a data stream. It covers how feature importance and the associated feature cost was modeled as feature merit and how this merit was used to assess the quality of an instance, guiding acquisition decisions.

10.2.1. Budgeting Acquisitions on a Stream with an IPF

The Incremental Percentile Filter (IPF) was originally designed for budgeting based on a relative allocation, such as acquiring 50% of the missing labels. To function, the IPF requires a score that estimates the utility of acquiring a missing label. The IPF uses this score to determine whether a label should be purchased. It operates by maintaining a sliding window of fixed length, where scores are stored in an ordered manner. As new scores arrive, the oldest score in the window is removed.

Using the relative budget—in this case, 50%—and the window length, a rank threshold is computed by multiplying the two values. This threshold indicates the rank a score must reach to trigger the acquisition. For instance, with a window length of 6 and a budget of 0.5 (50%), any score that ranks in the top 3 ($6 * 0.5 = 3$) will prompt an acquisition. Figure 10.1 provides a visual representation of how the IPF operates.

Unless the distribution of the scores changes over time, this setup leads to a budget expenditure close to the user-defined limit.

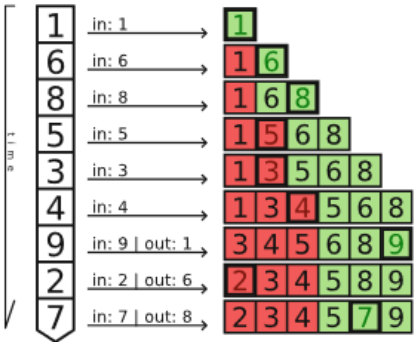

Figure 10.1.: Incremental Percentile Filter with a window length of 6, a relative budget of 0.5 (50%), and the oldest score being dropped if the maximum window length is reached. Figure reproduced with permission from the author from [41].

10.2.2. Budgeting Acquisitions on a Stream with an SBM

An alternative budgeting mechanism, the **Simple Budget Manager** (SBM), was proposed as a comparison to the IPF. The SBM operates by monitoring the ratio of triggered acquisitions to instances with missing features, checking if this ratio falls below a user-defined threshold. If the current ratio is below the threshold, the SBM triggers an acquisition; otherwise, it holds off until the ratio decreases below the threshold again.

For example, with a threshold of 0.1, if one feature has been acquired out of 11 instances with missing features, the ratio remains below the threshold ($1/11 < 0.1$), prompting the SBM to trigger a feature purchase for the next instance. However, once 2 acquisitions have been made out of 12 instances, the ratio exceeds the threshold ($2/12 > 0.1$). As a result, no further acquisitions are triggered until 21 instances with missing features are encountered, at which point the ratio falls below the threshold again ($2/21 < 0.1$).

With both the IPF and SBM available, and under the assumption of equal feature costs and acquiring a maximum of one feature per instance, the AFA problem was simplified to one resembling traditional Active Learning (AL). This allowed the problem to be modeled in relative terms rather than absolute costs. For instance, in a stream containing 1000 instances per day, with a feature cost of 5€ and a daily budget of 100€, only 20 features can be acquired per day ($100/5 = 20$), which corresponds to acquiring features for 2% of the instances ($20/1000 = 0.02$). The IPF threshold, in this case, would be set at 0.02.

The next step was to develop a score that could be used by the IPF to guide the acquisition of valuable features for instances that would benefit the most. The requirements for this scoring system are twofold: it must address both the features themselves and the characteristics of the instance.

10.2.3. Modelling Feature Importance on a Stream

The presented approach employs metrics originally developed for subset selection on data streams [85]. In subset selection, the aim is to disregard irrelevant features and train a classifier using a smaller subset of features, which typically reduces

computational complexity and mitigates the risk of overfitting [46]. To achieve this, it is crucial to assess the contribution of each feature to the classification task.

The authors of [85] propose several metrics that can be efficiently updated on data streams and assume feature independence. These include average Euclidean distance (AED), symmetric uncertainty (SU), and information gain (IG). Among these, AED is highlighted due to its simplicity, ease of implementation, and comparable performance to the other metrics in the experiments. Detailed explanations of the other metrics can be found in Appendix A.2.1.

AED calculates the mean value $MV(F_{ic})$ of a feature F_i for each possible class c among the L classes and then computes the Euclidean distance between the class-conditional means to determine how far apart they are. This process is described in Equations 10.1 and 10.2.

$$MV(F_{ic}) = \frac{1}{|F_{ic}|} \sum_{n=0}^{|F_{ic}|-1} F_{ic}^n \tag{10.1}$$

$$AED_{num}(F_i) = \sqrt{\sum_{0 \leq c < k < L} (MV(F_{ic}) - MV(F_{ik}))^2} \tag{10.2}$$

In the case of categorical features, the metric compares how many instances of a particular class share that specific feature value compared to all the other classes. Here F_{icv} denotes the set of instances that belong to class c, where the feature F_i has value v and V_i denotes all the categorical values that F_i can take. The categorical AED is then calculated as in equation 10.3.

$$AED_{cat}(F_i) = \sum_{0 \leq c < k < L} \left[\frac{1}{|V_i|} \sum_{v \in V_i} \left(\frac{|F_{icv}|}{|F_{ic}|} - \frac{|F_{ikv}|}{|F_{ik}|} \right) \right] \tag{10.3}$$

Merit of a feature is defined to be the feature's importance $g(F_i)$ (e.g., $AED(F_i)$) divided by the feature's cost C_i, see equation 10.4. The merits of all features are kept in a vector based on the current window W and the costs of the features C, equation 10.5

$$merit(F_i) = \frac{g(F_i)}{C_i} \tag{10.4}$$

$$merits(W, C) = (merit(F_0, C_0), \ldots, merit(F_{|F|-1}, C_{|C|-1}))^T \tag{10.5}$$

This definition of merit allows for experiments with varying feature costs later on and to replace the feature importance metric according to the use case.

Feature Importance Windows: For the purpose of calculating feature importance, only true feature values are considered, with imputations being disregarded. Since a windowing approach is employed to mitigate potential concept drift, several alternatives exist for computing streaming feature importance. One method, referred to as the single window (SW), maintains a single window for all features, while the second method, referred to as multiple windows (MW), retains distinct windows for each feature and class combination. Let w denote the size of the window. In the SW approach, the most recent w instances are stored, while MW retains the most recent w available feature values for each feature-class combination. For instance, in a dataset with three features and two classes, MW would require $3 \times 2 = 6$ windows, whereas SW would only use a single window for feature importance computation.

The primary benefit of SW is that the merit estimates for each feature are derived from identical instances and span the same time period. However, a drawback is that the window may not contain w instances for every feature, as the absence of a feature in the stream reduces the number of values retained within the window. Conversely, MW ensures that up to w feature values are kept per class for each feature, but the time span covered by the windows may vary, depending on the distribution of missing data and class balance within the stream.

Both windowing methods were employed in the AED metric experiments, detailed in section 10.5. Ultimately, the SW option was selected to ensure that the merit of each feature is based on the same temporal range and derived from the same set of instances.

With a merit function suitable for streaming data, it is now possible to assign a score to an instance using its available features as well as the missing feature with the highest merit.

10.2.4. Modelling Instance Quality

We assume a scenario where we must decide immediately whether we want to acquire missing features for each incoming instance. This decision is guided by estimating the value of acquiring the most informative missing feature for the current instance.

To perform this estimation, the previously introduced merit function is applied. The quality of an instance x is calculated by first summing the merits of all known features $f \in x.known$, then adding the merit of the highest-ranked missing feature. This combined value is then normalized by the total number of features, as shown in equations 10.6 and 10.7. The normalization ensures that instances with fewer missing features are not disproportionately favored when deciding whether to acquire additional features.

$$best\_f(x) = argmax_{f \notin x.known} merit(f) \tag{10.6}$$

$$quality(x, merits(W, C)) = \frac{\sum_{f \in x.known} merit(f) + best\_f(x)}{|x.known| + 1} \tag{10.7}$$

For each incoming instance, the quality score can now be calculated and plugged into the IPF to decide whether to trigger an acquisition, see algorithm 10.1.

Algorithm 10.1 AFA-Stream

Require: stream X, budgetmanager bm, costs C
Ensure: window W initialized
  for all instance x in X **do**
    $m \leftarrow merits(W, C)$ // described in 10.5
    $q \leftarrow quality(x, m)$ // described in 10.7
    if $bm(q)$ **then**
      acquire feature with highest *merit* // described in 10.4
    end if
    update W
  end for

10.3. Evaluation Scheme and Datasets

Random Sampling is a widely used baseline in Active Learning research [74], and in certain cases, it can be surprisingly challenging to outperform. Consequently, a Random Acquisition (RA) baseline was implemented, employing the Simple Budget Manager (SBM) discussed in section 10.2.1. This baseline randomly selects a missing feature for acquisition whenever the ratio of acquisitions to instances with missing features falls below a predefined user threshold.

To assess the effectiveness of the approach, the classification performance was compared across nine datasets using a prequential evaluation approach. Cohen's Kappa was chosen as the performance metric due to its suitability for handling class imbalance, as outlined in section 2.2. It was used in favor of κ^+ as temporal aspects, and the performance of the classifier are not of interest, but the performance difference given an AFA strategy.

The method was tested on six static datasets and three stream-based datasets, listed in Table 10.1. The static datasets, **abalone, adult, magic, nursery, occupancy**, and **pendigits**, are all accessible through the UCI machine learning repository[1] [40]. The stream datasets consist of **electricity**, **sea**, and **gen**. The **SEA** dataset is synthetic and commonly used in stream mining experiments, comprising four distinct concepts, each spanning 15,000 instances. The **GEN** dataset is another synthetic stream dataset, developed for this work, which has 10 concepts, each lasting for 500 instances. During each concept, a categorical feature is made to resemble the label, enabling an assessment of how the feature importance metric responds to concept changes.

In order to facilitate reproducibility and support future experimentation with alternative Active Feature Acquisition strategies and feature importance metrics, an evaluation framework was established, which will be detailed in the following section.

Table 10.1.: All datasets that were used for the experiments. Table reproduced from [9].

Dataset	Instances	Labels	Features	Type	Purpose
sea	60,000	2	0 cat. 3 num.	synth. stream	determine if two specific features greater than threshold
electricity	45,312	2	1 cat. 7 num.	stream	determine if the market price of electricity rises or drops
adult	32,561	2	4 cat. 8 num.	static	determine if yearly income of individual is above $50,000
occupancy	20,560	2	1 cat. 7 num.	static	determine whether an office room is occupied
magic	19,020	2	0 cat. 10 num.	synth. static	determine if signal is gamma ray based on Cherenkov radiation
nursery	12,960	5	8 cat. 0 num.	static	determine the rank of child application for nursery school
pendigits	10,992	10	0 cat. 16 num.	static	determine digit written on a pad
gen	5,000	2	3 cat. 0 num.	synth. stream	find the current feature describing the label
abalone	4,177	3	0 cat. 8 num.	static	determine sex of abalones

10.4. Evaluation Framework

The evaluation framework[2] evaluates various AFA strategies by measuring their impact on the performance of a stream classifier applied to the same data stream. The

[1] `https://archive.ics.uci.edu/`, visited 29th of May 2024
[2] `https://github.com/Buettner-Maik/afa-stream`

core assumption is that an effective AFA strategy will acquire missing features that contribute to significant improvements in classification performance. Performance is assessed using Cohen's Kappa, chosen for its robustness in handling class imbalance.

Simulation of Missingness: To ensure consistency across runs, a feature-complete dataset X is used. Missing values are introduced according to the Missing Completely at Random (MCAR) mechanism, whereby a user-defined threshold is applied to randomly delete a specified percentage of selected feature values based on a uniform distribution. The missing values are then imputed using the mean for numerical features and the most frequent value for categorical features. Running a stream classifier on the imputed dataset without any feature acquisitions establishes a lower performance bound, while the feature-complete dataset provides the upper bound.

The specific choice of imputation method is arbitrary; however, the better the imputation, the closer the lower bound may be to the upper bound. For this reason, basic imputation methods were selected. While mean and most frequent value imputation are not the most accurate methods, they simplify the visualization of results. Additionally, the framework includes the random baseline discussed earlier, and for any AFA strategy to be considered successful, it must perform better than both the lower bound and the random baseline.

10.5. Experimental Setup

The evaluation framework was realized by extending an existing framework for AL on data streams[3] which was designed for streams with an evolving feature space.

Stream of Batches: The original framework processes incoming streams in batches, and this setup was maintained throughout the experiments. However, as the developed methods are also compatible with conventional data streams, reimplementation is currently underway using the widely adopted Python online machine learning library, River[4].

For the experiments, each dataset is divided into batches of 50 instances, with the first batch used to initialize the classifier. To ensure that all labels are represented during initialization, one instance from each label is randomly selected and added to the first batch. For static datasets, the order of instances is randomized for each run to ensure robust performance estimates. In contrast, for stream datasets, the chronological order is maintained to accurately reflect concept drift and preserve the original temporal sequence of the data.

Parameters for Missingness and Budget: In the experiments, Missing Completely at Random (MCAR) missingness is simulated across seven levels, specifically with 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, and 87.5% of the feature values being missing. Additionally, eight different budget scenarios are applied, allowing for the acquisition of one missing feature per instance in 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, and 100% of the instances.

Classifier and Number of Runs: A stochastic gradient descent support vector machine (SGD SVM) from the scikit-learn library was employed with a limit of 100

[3] `https://github.com/elrasp/osm`

[4] `https://riverml.xyz/latest/`, visited 7th of June 2024

iterations to achieve a tolerance of 0.001 using log-loss. This choice was driven by the requirements of the framework, though it is not crucial for the experiments, as selecting a different classifier would mainly impact the performance bounds but not the ranking of the Active Feature Acquisition (AFA) strategies.

To account for the inherent randomness in the experiments, each static dataset was permuted 10 times for every combination of parameters (Budget Mechanism, Budget, Degree of Missingness, Window Mode, and Feature Importance Metric). For the streaming datasets, only one run was performed, preserving the original sequence of instances. Initial experiments were conducted using the AED metric, but it was observed that there was no significant difference between the single window (SW) and multiple windows (MW) approaches. Given that MW was more complex to implement and manage, the SU and IG feature importance metrics were implemented solely using the SW approach.

This resulted in a total of 5180 runs for static datasets and 518 runs for stream-based datasets.

10.6. Results

The experiments had the following aims:

1. Find out if the stream-based AFA methods outperform a random baseline.
2. Investigate if AED is a suitable feature importance metric.
3. Investigate what effect the two different budgeting mechanisms have on performance.

With respect to the first point, it can be stated with confidence that the proposed AFA methods outperform the random acquisition (RA) baseline on 7 out of the 9 datasets. The only exceptions are the **adult** and **abalone** datasets, as well as the **gen** dataset at very low missingness levels, as shown in Figures 10.2 and 10.3. These figures illustrate model performance across different budgets with 75% of the feature values missing. It is evident that $SWAED$, regardless of the budgeting mechanism, consistently surpasses the RA baseline, with this effect becoming more pronounced as the proportion of missing features increases. The specific results for the **magic** dataset are detailed in Table 10.2, while the results for the remaining datasets are available in the Appendix, sections A.1 to A.8.

Table 10.2.: Mean kappa values over 10 runs on the **magic** dataset using an SGD classifier. $SWAED + IPF$ always outperforms $RA + SBM$. Table reproduced from [9].

	mean kappa over 10 runs on data set magic											
missingness	0.25(kappa $\in [0.319, 0.409]$)				0.5(kappa $\in [0.229, 0.409]$)				0.75(kappa $\in [0.131, 0.41]$)			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.344	0.357	0.375	**0.395**	**0.27**	**0.303**	0.333	0.367	**0.185**	0.226	**0.276**	0.321
MWAED+SBM	0.337	0.357	0.375	0.394	0.26	0.292	0.326	0.364	0.177	0.224	0.274	0.321
SWAED+IPF	**0.347**	**0.358**	**0.377**	0.391	0.265	0.302	**0.339**	0.366	**0.185**	**0.23**	0.27	**0.323**
SWAED+SBM	0.34	0.357	0.376	**0.395**	0.261	0.296	0.333	**0.368**	0.174	0.223	0.274	0.317
SWIG+IPF	0.341	0.353	0.372	0.391	0.259	0.29	0.32	0.358	0.173	0.205	0.256	0.301
SWIG+SBM	0.34	0.357	0.375	0.392	0.259	0.289	0.326	0.361	0.165	0.209	0.256	0.299
SWSU+IPF	0.336	0.353	0.37	0.391	0.256	0.287	0.32	0.359	0.171	0.205	0.251	0.299
SWSU+SBM	0.337	0.354	0.376	0.393	0.257	0.29	0.325	0.361	0.169	0.212	0.254	0.304
RA+SBM	0.326	0.334	0.34	0.355	0.232	0.248	0.26	0.266	0.142	0.151	0.159	0.171

The only exceptions are the **abalone** and **adult** datasets. In the case of abalone, the most likely reason is that we created an almost unsolvable classification task from a dataset, which is usually used for regression. A strong indicator is the very low upper bound, which reaches, at most, a Kappa value of 0.252 when all features are available. This means there is very little to learn in the first place, which is supported by the fact that at 25% missingness, the lower performance bound, where all missing features are imputed, lies at 0.226, which is just below the upper bound of 0.238, which means that 25% of the features missing made almost no difference to the classifier. On **adult** the issue seems to be that the artificial restriction to only be able to acquire one feature per instance, is too limiting. This assessment is supported by the huge gap between all strategies and the upper bound, which can be seen in Figure 10.3. Eventhough, the proposed methods were not clearly favorable on the two aforementioned scenarios, they still performed similarly to the random baseline. Summarizing one can say that the proposed methods outperform the random baseline in the vast majority of cases and only occasionally perform similar to the random baseline which is mostly due to the restriction of being only able to purchase a single feature, as well as using a dataset for testing which seems not suitable to the task.

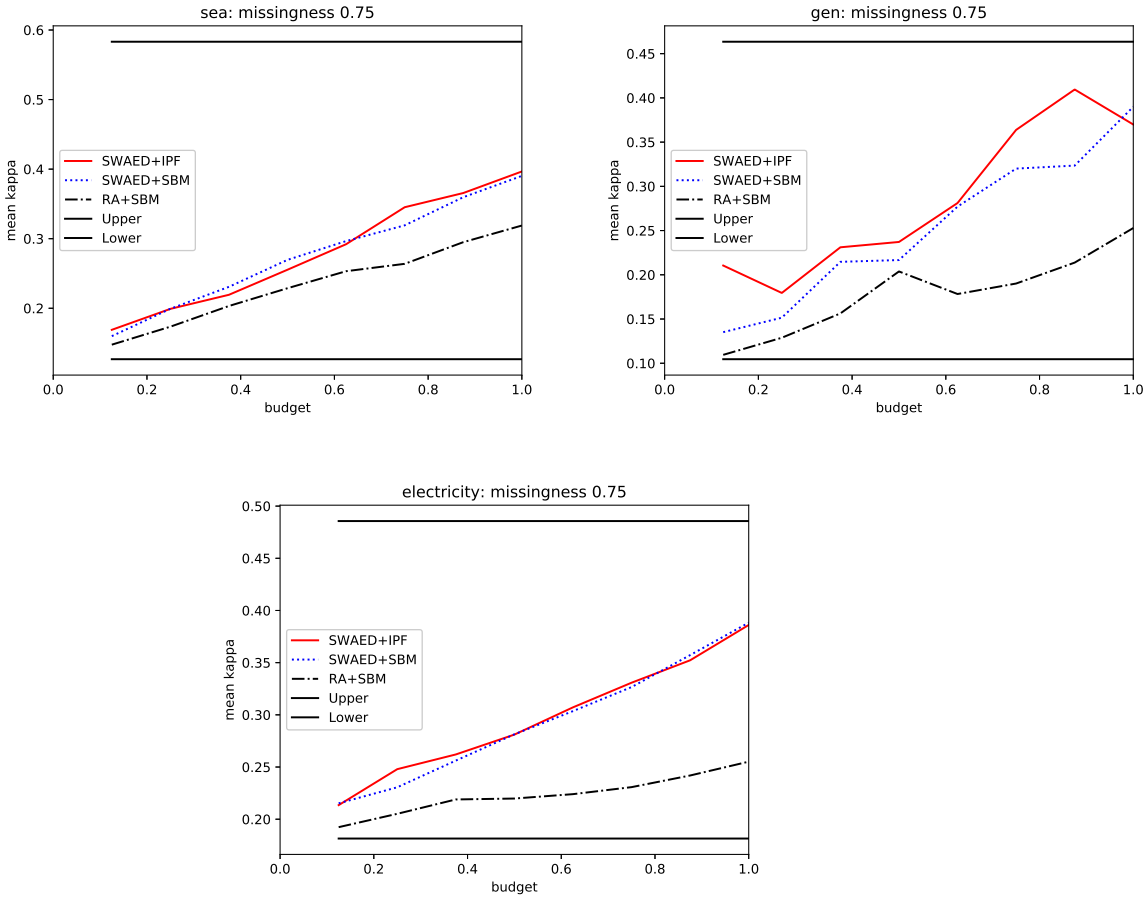

Figure 10.2.: Mean kappa performance comparison over ten runs of *Single Window Average Euclidean Distance (SWAED)* configurations on three stream datasets, with a fixed feature missingness rate of 0.75. Figure reproduced with permission from [9] under CC BY 4.0.

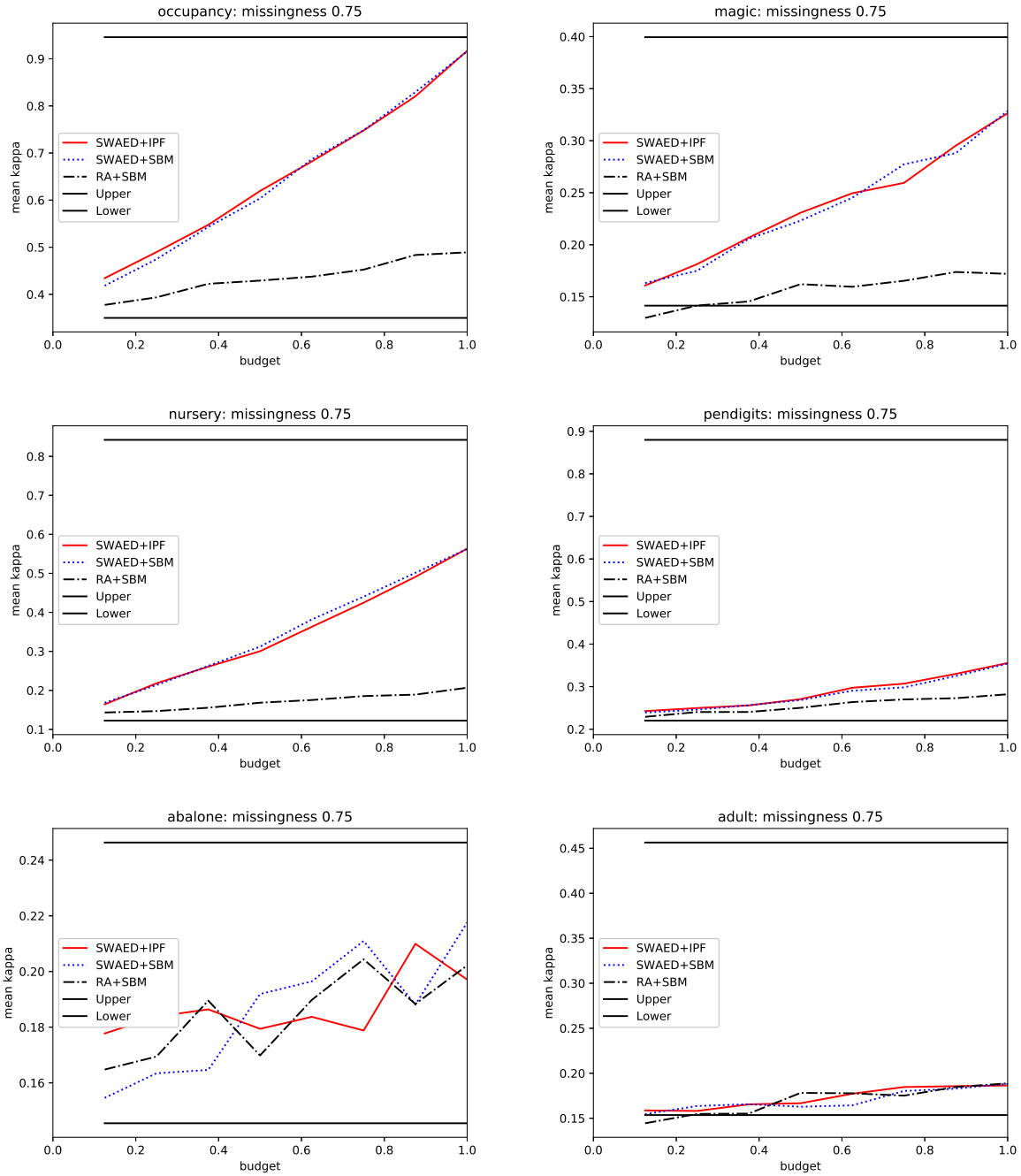

Figure 10.3.: Mean kappa performance comparison across ten runs of *Single Window Average Euclidean Distance (SWAED)* configurations on six static datasets, with a fixed feature missingness rate of 0.75. Figure reproduced with permission from [9] under CC BY 4.0.

Investigating AED as a Feature Importance Metric: All three feature importance metrics, Average Euclidean Distance (AED), Information Gain (IG) and Symmetric Uncertainty(SU), usually outperform the baseline. Still, it was observed that IG and SU performed better on all datasets containing categorical features. Unfortunately, SU and IG require a discretization step [27], which noticeably slows down the processing of the stream. Therefore AED is the preferred option, which performs well and is fast, but a more detailed investigation of which feature importance metric fits which data and if there are heuristics that can tell us which metric to use on a given dataset are still open for future work. One obvious drawback of using the proposed metrics to suggest acquisition candidates is, that they consider each feature independently, ignoring correlation and other inter-feature dependencies. This aspect is highlighted more in chapter 11 where a synthetic dataset is introduced, where the label is decided by the configuration of multiple features, and in chapter 12 where a synthetic dataset is introduced that contains multiple strong correlations among the features. An alternative approach using decision trees to generate is currently in the works and being prepared for publication, see section 13.1.

Effects of Budgeting Mechanism: The experiments did not reveal a substantial difference between the Simple Budget Manager (SBM) or Incremental Percentile Filter (IPF), which can be visually confirmed in the Figures 10.2 and 10.3. In most cases the two lines match each other, or are very close even when one dominates the other. This wasn't a big surprise as both methods show the same behavior at the thresholds 0 and 1, as they never acquire features in the former and always a acquire the best missing feature in the latter case. So the only expected difference would be the shape of the curve. A slight advantage of the IPF was anticipated, as it can contextualise its current decision within the window of past scores, which in principle should lead to better decisions than just buying when you can. While there seems to be a slight advantage in using the IPF, the experiments did not underscore this expectation sufficiently. Therefore, a more in depth investigation into the behavior and performance impacts of budgeting mechanisms, is part of the open future work.

AED, Window Length and Concept Drift The principal method to deal with potential concept drift in this paper is to use a windowing approach when calculating the merits of our features and training the classifier. Windows of static size have the disadvantage though, that they can miss change our react late if they are too big and might overreact to short fluctuations and outliers if they are to small. An example of a scenario where the window length is too long can be seen in Figure 10.4, where our metrics only reacted to a new concept after half of the respective instances were seen. Potential solutions that can address this issue in the future are the use of change detection algorithms and adaptive windowing [11].

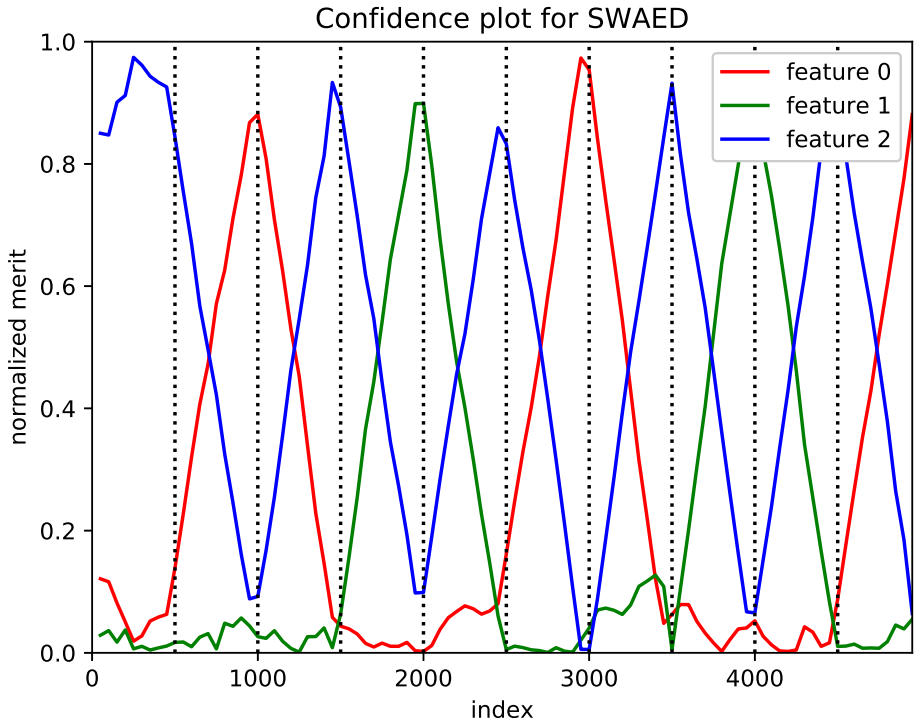

Figure 10.4.: The merit of each feature is plotted over time on the **gen** dataset, with concepts lasting 500 instances, indicated by vertical lines. In each concept phase, one feature aligns with the label. The AED calculation uses the same window size of 500, meaning it takes half a window to detect the current concept. Figure reproduced with permission from [9] under CC BY 4.0.

10.7. Conclusion

Missing feature values in a data stream can degrade prediction performance. They are usually treated with imputation, which relies on educated guessing of the missing values. An alternative approach is active feature acquisition, where true feature values can be acquired for a cost.

This work represents one of the initial efforts to investigate how Active Feature Acquisition (AFA) can be applied to data streams, where budgeting must account for an endless flow of incoming instances. As part of this investigation, a framework for conducting AFA experiments on data streams is provided, though it is subject to the limitation of allowing only one missing feature to be acquired per instance. Additionally, all features were assumed to have equal costs, as the focus was on evaluating the effectiveness of the selected feature importance metrics rather than being influenced by varying feature costs, which can differ across datasets.

To evaluate how well an AFA strategy identifies the missing features that most influence classification outcomes, the performance of a stream classifier is measured using Cohen's Kappa. The framework incorporates three feature importance metrics (AED, IG, and SU), two stream budgeting mechanisms (IPF and SBM), and a random acquisition baseline for comparison. It also calculates and displays the lower and upper performance bounds: the lower bound reflects the classifier's performance when no missing features are acquired and all missing values are imputed, while the upper bound represents the classifier's performance in the absence of any missing features.

The results successfully demonstrate that utilizing streaming feature importance heuristics in combination with a stream-appropriate budgeting mechanism generally leads to superior acquisition decisions compared to a random baseline on 7 out of the 9 datasets. The `abalone` and `adult` datasets presented exceptions where the proposed AFA methods performed similar to the random baseline. In the case of `abalone`, the learning task was considered intractable, while for `adult`, the limitation of acquiring only one feature per instance likely proved too restrictive. This hypothesis is supported by subsequent experiments, as shown in Table A.9.

In terms of the two budgeting methods, no definitive conclusion was reached regarding their relative effectiveness; however, IPF appeared to perform marginally better. Consequently, SBM was discontinued in future experiments, except when used for the random baseline.

One of the key questions was whether AED would serve as a suitable metric for estimating feature importance, particularly given its applicability to both numerical and categorical data without requiring costly discretization. The results indicate that AED is indeed effective, consistently outperforming the random baseline. Nevertheless, SU and IG tend to deliver superior results on datasets with categorical features. Despite this slight advantage, the additional cost of discretization was deemed unnecessary, and thus AED was selected for use in future experiments.

In the next chapter this work is extended by introducing cost-aware AFA on data streams with the potential for larger acquisition sets, which addresses the two most important limitations of this chapter.

11. Cost-Aware AFA

This chapter covers the work published in [15] which addresses:

RQ3: How can Active Feature Acquisition be realized in a data stream setting?
RQ4: How can varying feature costs be considered during AFA on streams?

The previous chapter focused on the initial realization of AFA on a data stream, addressing **RQ3**. This follow-up work focuses on **RQ4** while also extending **RQ3** by allowing for the acquisition of more than one feature per instance and adjusting the scoring function for the IPF accordingly. Furthermore, statistical tests are introduced to validate the results better.

One of the biggest drawbacks of the approach presented in chapter 10 is, that it was tried to model AFA in a way similar to AL. This resulted in the constraints that we can only acquire one missing feature per instance, akin to purchasing a label for an instance. The second limitation was that all features are assumed to cost the same, which made it possible to model the problem in terms of relative budget, enabling the use of a preexisting budgeting mechanisms for AL on data streams.

The work presented in this chapter had the main goals to enable varying feature costs and bigger acquisition sets. This means a budgeting mechanism that works in absolute terms and enables the acquisition of multiple features per instance is needed, which also makes changes to the budgeting mechanism and the scoring of instances necessary. The main contributions are:

- Providing an updated research framework for AFA on data streams
- Changing the budgeting model to deal with multiple feature acquisitions per instance
- Modelling budget in absolute terms and enabling varying feature costs
- Providing a mechanism to deal with overspending by means of a penalty to the IPF threshold

The major changes to the framework described in the previous chapter can be seen in Figure 11.1. They encompass multiple strategies for feature set selection (fss), changing the score handed to the IPF from estimated quality to estimated quality gain, acquiring a set of features instead of the single best missing feature and lastly a necessary update to the IPF threshold in order to deal with changing acquisition set sizes and their associated costs. The sections 11.1, 11.4 and 11.5 are based on the respective counterparts in [15].

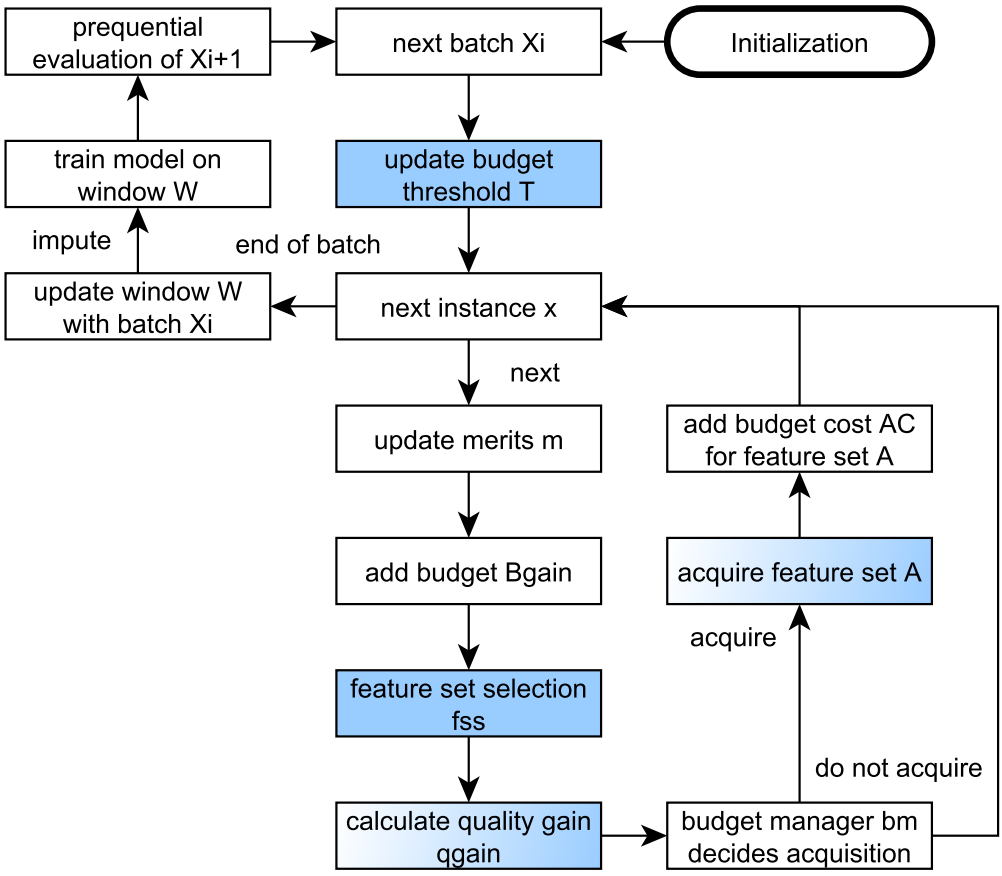

Figure 11.1.: The framework figure shows fully colored blue nodes for new elements, while shaded blue nodes represent elements modified from chapter 10. Reproduced with permission from [15] ©2022 IEEE.

11.1. Related Work

Two more recent works that address AFA and extend the related work presented in the previous chapter, use reinforcement learning.

In [48] the authors model feature acquisition as a Markov decision process (MDP). In this framework, the state space consists of different sets of known features, along with a terminal state representing the end of the acquisition process. Feature acquisition is viewed as state transitions, with actions corresponding to acquiring additional features. Rewards are assigned based on information gain and instance uncertainty, making this method adaptable to stream environments.

Building on this, [49] introduces hierarchical decision chains for high-dimensional feature spaces, clustering actions and incorporating out-of-distribution detection. This refinement enhances the method's accuracy and flexibility, though the authors do not address budget management for feature acquisition, which is the central focus of this chapter.

11.2. Methods for Acquiring Sets of Features

In order to make the AFA framework of chapter 10 more realistic to real-life situations, it was necessary to lift the artificial constraint of only being able to acquire a single feature per instance. Therefore, three types of strategies which accomplish this goal were designed. In case of all three strategies the user defines a k which sets the maximum number of features that can be acquired per instance.

k-Best Strategy: All missing features of an instance x will be ranked according to their *merit* and then the top k missing features will be put into the acquisition set for x.

k-GlobalBest Strategy: All features are ranked according to their merit, irrespective of the instance under consideration, and only features within the global top k ranks are eligible for acquisition. This approach imposes a stricter limitation compared to the k-Best strategy, as budget can only be spent on the global top k features, regardless of which features are actually missing in a given instance x. Consequently, if x has multiple missing features but none of them are ranked within the global top k based on their merit, no acquisition is allowed. In contrast, with the k-Best strategy, up to k features could still be acquired, even if they do not belong to the global top k ranked features.

k-MaxMean: First the quality of an instance x is calculated without any acquisitions, formula 11.1

$$quality(x) = \sum_{f \in known(x)} merit(f)/|known(x)| \tag{11.1}$$

Then iteratively the feature with the highest merit is added as long as the $quality(x)$ improves. This strategy often ends up selecting only one or two features for the acquisition.

We can see the differences in behavior of these strategies in Figure 11.2, for instance how $4 - GlobalBest$ is more restrictive than $4 - Best$ on instance 1, as it only considers features $f2$, $f1$,$f3$ and$f5$ as these have the highest merits (15, 11, 9, 6) and of those 4 globally best features instance 1 has only$f2$ and$f3$ missing so its other missing features are not considered by $4 - GlobalBest$ but would be considered under $4 - Best$. One can also see how $4 - MaxMean$ would only purchase 2 features as well, as instance 1 starts with a quality of 6 based on its available attributes ($quality(instance1) = (11 + 6 + 1)/3$), goes up to 8.25 if we add the best missing feature $f2$, goes to 8.4 when adding $f3$ and would have dropped to 7.66 if we had added the next best feature $f7$.

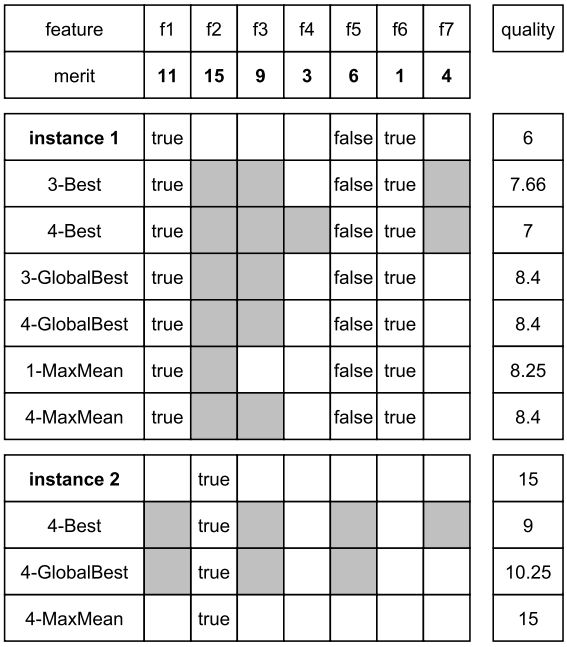

Figure 11.2.: Examples of $k - Best$, $k - GlobalBest$, and $k - MaxMean$ on two instances from [15]. Missing features are indicated by white squares. The merit of each feature is displayed at the top, with selected features for each fss-strategy shown in grey and the resulting instance quality at the end of each row. Reproduced with permission from [15] ©2022 IEEE.

The figure shows how different fss-strategies lead to very different acquisition sets, which in turn will lead to very different strains on our available budget, so let us consider next how budgeting is being realized in the framework and which steps were taken to facilitate the full usage of the available budget.

11.3. Methods for Dealing with Feature Cost and Absolute Budgeting

In the proposed framework, the budget is distributed incrementally for each incoming instance, rather than making the entire budget available at once. With every new instance, a budget increment B_{gain} is received, and the total budget received ($B_{received}$) and spent (B_{spent}) are tracked accordingly; see Figure 11.1.

In a real-world application, this would require estimating the number of instances to be processed within a specific time frame, and dividing the overall allocated budget by this estimated number of instances.

This design was chosen to facilitate an even distribution of the budget across the stream. Without this approach, especially in high-missingness scenarios, a significant portion of the budget could be spent on consecutive instances, which could lead to missing important events such as concept drift.

The objective of the budget mechanism is to ensure that all available budget is utilized without exceeding it, so the ratio of spent budget to received budget $B_{used} = \frac{B_{spent}}{B_{received}}$ should approach 1. To achieve this, the IPF allows for more acquisitions when there is surplus budget and restricts acquisitions when overspending occurs, limiting further spending until the necessary budget has accumulated.

To enable this behavior, the threshold used by the IPF is dynamically adjusted according to the current budget usage.

11.3.1. Adapting the IPF-Threshold with a Penalty:

When transitioning from relative feature costs to absolute feature costs, it became necessary to estimate an initial value for the threshold T used by the IPF. As a reminder, T represents the percentage of decisions for which the IPF should decide positively. Let A denote the acquisition set, and A_C the associated cost of acquiring A. The average acquisition cost, $\overline{A_C}$, is then defined. The initial value of the threshold is calculated as the fraction of the budget gained per instance over the average acquisition cost:

$$T_{pre} = \frac{B_{gain}}{\overline{A_C}} \tag{11.2}$$

Since the true average acquisition costs are typically unknown, two different approaches were implemented to estimate the threshold. The first approach assumes the worst-case scenario, using the maximum possible acquisition costs, while the second approach tracks recent acquisition costs to provide an estimate of the average. Let C represent the set of feature costs, where each C_f denotes the cost of feature f. The threshold in the worst-case scenario is determined by using the maximum cost from the set C:

$$T_{pre} = \frac{B_{gain}}{\sum_{f \in F} C_f} \tag{11.3}$$

Alternatively, the costs of past acquisitions can be tracked and then used to estimate $\overline{A_C}$ using the mean of the recorded costs $\hat{A}_C$.

$$T_{pre} = \frac{B_{gain}}{\hat{A}_C} \tag{11.4}$$

As a batch-based framework is used to process the streams, the threshold is only updated after the first batch has been processed; see Figure 11.1.

The advantage of assuming the maximum cost is that it generally ensures the budget is not exceeded, but it often leads to underutilization of the available budget. In contrast, the second method, which tracks recent acquisition costs, tends to overspend and requires the collection of examples to estimate $\hat{A}_C$. Both behaviors were found to be unsatisfactory, as the aim is to avoid both overspending and leaving budget unutilized.

To address this,the threshold is dynamically recalculated based on the actual budget usage B_{used}:

$$T_{basic} = \frac{T_{pre}}{B_{used}} \tag{11.5}$$

Second, a penalty term is added, which triggers in case of overspending, as initial experiments showed that adjusting with B_{used} was still not enough to deal with overspending.

$$P = \lfloor p_c \cdot (B_{used} - 1) \rfloor + 2 \tag{11.6}$$

$$T = \begin{cases} T_{basic} & if B_{used} \leq 1 \\ \frac{T_{basic}}{P} & otherwise \end{cases} \tag{11.7}$$

A comparison of all three methods can be seen in Figure 11.3 where it can be observed that the red line denoting the maximum cost approach often stays under the target value of 1, which means the available budget was underutilized. The green line denotes the tracking-based approach with $\hat{A}_C$, which often stays over the target value of 1, meaning it spends more budget than was available. The blue line denotes the final version, including a penalty for overspending, which mostly stays at the target value of $B_{used} = 1$.

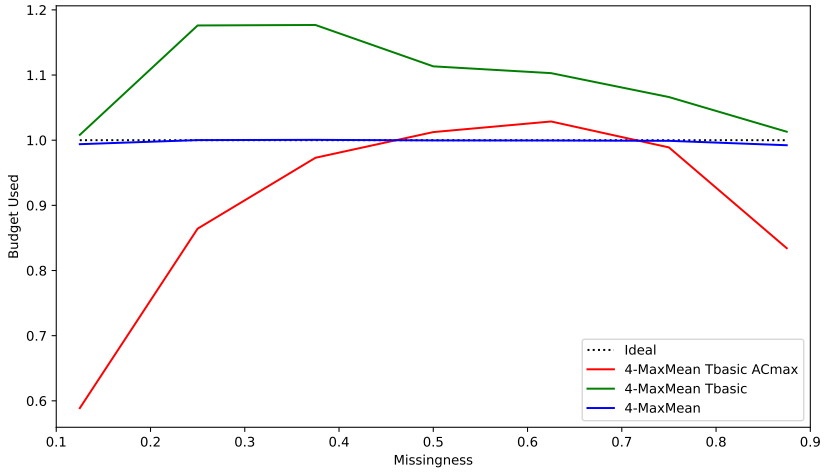

Figure 11.3.: The figure shows the budget usage on the *pendigits* dataset. Assuming maximum costs (red line) results in underspending, tracking actual costs (green line) leads to overspending, while introducing a penalty term achieves near-optimal budget utilization. Reproduced with permission from [15] ©2022 IEEE.

Overspending can still be an issue, especially in scenarios with high missingness, low budget, AFA strategies that select many features, and high feature costs. High missingness means the methods often want to trigger acquisitions, which strains the budget, and high costs amplify this issue. A lot of features missing means that the strategies might want to buy more missing features, which, in the case of high costs and AFA strategies that allow for bigger acquisition sets, translates into high average costs per instance for which acquisitions are triggered. In certain scenarios, even a few positive acquisition decisions can lead to overexpenditure. No matter how the IPF threshold is adjusted, if an instance arrives with a higher score than currently

included in our IPF, it will move to the front of the IPF, which will always trigger an acquisition; see Figure 11.4.

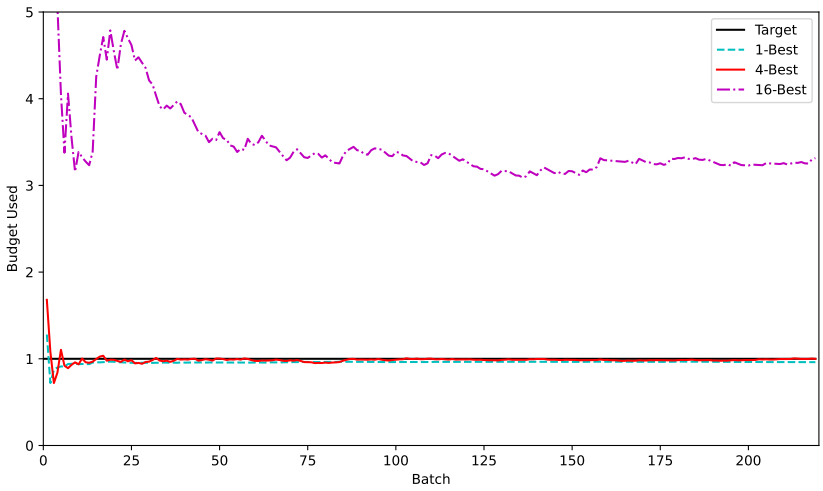

Figure 11.4.: The figure illustrates how the $16-Best$ strategy (pink line) overspends due to high acquisition costs, even with fewer triggers. In contrast, $4-Best$ and $1-Best$ remain near the target value of 1. Reproduced with permission from [15] ©2022 IEEE.

During early experiments, the insight was gained that using the instance quality as the score that is being handed to the IPF has certain disadvantages, which will be discussed next.

11.3.2. Replacing Quality Score with Quality Gain:

In chapter 10 the quality of an instance was calculated after the potential acquisition of the best missing feature. The quality score was handed to the IPF, which then decided whether to trigger the acquisition. The quality was defined as follows:

$$quality(x) = \sum_{f \in known(x)} merit(f)/|known(x)| \tag{11.8}$$

Using the quality directly wastes information about an instance's initial quality, which can negatively affect budgeting. For example, let us consider two instances x_1 and x_2, where x_1 has the feature with the highest merit f_{m_1}, but lacks the feature with the second highest merit f_{m_2}, x_2 on the other hand lacks the feature with the highest merit f_{m_1} but has the feature with the second highest merit available f_{m_2}. If we assume that both instances share all the other features, then this means that $quality(x_1) > quality(x_2)$. The use of the $1-Best$ strategy, would consider f_{m_2} for x_1 and f_{m_1} for x_2. The resulting quality estimates after acquisition would be the same, which makes these instances look the same to our budget manager $quality(x_1 \cup f_{m_2}) = quality(x_2 \cup f_{m_1})$. This is the case even though x_1, according to the merits, had a better quality to begin with and could also lead to a waste of budget. In an extreme scenario where the decision must be made to acquire either f_{m_2} for x_1 or f_{m_1} for x_2, the preference would be to allocate the budget to x_2 because this would provide greater information gain according to our metrics.

As a result, we now calculate the quality of an instance after adding a feature set A to x ($quality(x \cup A)$) and compare it to the quality before the acquisition

($quality(x)$). The resulting gain in quality is then passed to the budget manager to inform future acquisition decisions.

$$quality_{gain}(x, A) = quality(x \cup A) - quality(x) \tag{11.9}$$

Using $quality_{gain}$ can have the theoretical disadvantage that budget might be left unused. This would be the case for instance 2 in Figure 11.2, which has all features missing except the feature with the highest merit. The way $quality(x)$ is defined, does not allow for any improvements as adding any feature with a lower merit will reduce the instances quality and therefore lead to a negative $quality_{gain}$, which makes acquisitions very unlikely. The choice which score to use and hand to the budget manager influences the acquisition behavior independent of which AFA strategy and merit metrics are being used. Therefore, the optimal choice depends on the specific scenario of an application.

Algorithm 11.1 formally describes the framework using the terminology depicted in Table 11.1.

Table 11.1.: Terminology for the AFA algorithm which was taken from [15]

Symbols	**Meaning**
X	the data stream
W	the window
X_i	the i-th batch of the stream
x	an instance
f	a feature
C	the set of feature costs
C_f	the cost of feature f
A	a feature set
A_C	the accumulated feature costs of feature set A
bm	the budget manager
fss	the active feature set selection strategy
B_{gain}	the budget added for every instance
T	the budget threshold for the budget manager
$B_{received}$	the total budget received so far
B_{spent}	the total budget spent so far
B_{used}	the fraction of budget used $B_{received}/B_{spent}$ so far

Algorithm 11.1 AFA-Stream as in [15]

Require: stream X, budgetmanager bm, active feature set selection fss, feature costs C, budget gain B_{gain}
Ensure: window W initialized

for all batch X_i in X **do**
  $B_{received} \leftarrow B_{received} + B_{gain}$
  if i > 0 **then**
    update threshold T // described in 11.3.1
  end if
  for all instance x in X_i **do**
    $m \leftarrow merits(W, C)$
    $A \leftarrow fss(x, m)$ // described in 11.2
    $q_{gain} \leftarrow quality(x \cup A, m) - quality(x, m)$
    if $bm(q_{gain})$ **then**
      acquire feature set A // described in 11.3
      $B_{spent} \leftarrow B_{spent} + A_C$
    end if
    update W
  end for
end for

Table 11.2.: Description of the acquisition strategies presented in [15]

Strategy	Description
1-Best	Buys at most the one locally best missing feature according to the merits using the IPF
2-Best	Buys at most the two locally best missing features according to the merits using the IPF
3-Best	Buys at most the three locally best missing features according to the merits using the IPF
4-Best	Buys at most the four locally best missing features according to the merits using the IPF
100-Best	Buys at most the hundred locally best missing features according to the merits using the IPF
4-GlobalBest	Buys at most 4 missing features if these features have the highest global merits using the IPF
4-MaxMean	Buys up to four features to maximize the instance's mean merit using the IPF
4-Random SBM	Random baseline that acquires at most 4 random missing features as long as budget is available by SBM

11.4. Experimental Setup and Evaluation

In the experiments 8 different AFA-methods were evaluated, see Table 11.2, on 15 different data sets, see Table 11.3. The data sets **adult**, **occupancy**, **magic**, **nursery**, **pendigits**, **abalone**, **electricity** and **sea** were also used in the previous chapter and are described in Section 10.3.

Similarly to the **sea** dataset, the **evenodd** datasets were generated to test how well AFA-methods perform when labels depend on multiple variables, but without introducing concept drift. This dependency is determined using an even/odd function: if the sum of the dependent features is odd, the label is 1; otherwise, it is 0. For instance, in the **evenoddf9d5** dataset, there are 9 features, of which 5 are the dependent features that determine the label, while the remaining 4 are noise.

The **evenodd** datasets replace the **gen** dataset from chapter 10.

Each dataset is tested with three different feature cost distributions. The first, *equal*, replicates chapter 10 by assigning the same cost to all features. The second, *increasing*, assigns feature costs based on their position, where the first feature costs 1 unit, the second 2 units, and so on. The third, *decreasing*, reverses the order, with the last feature costing 1 unit, the second to last costing 2 units, etc.

Additionally, we evaluate these cost settings under budgets (B_{gain}) of 1, 2, and 3 units. Each configuration is tested under seven levels of missingness, assuming Missing Completely At Random (MCAR), where random entries are deleted uniformly across the datasets. This results in a total of $3 \cdot 7 \cdot 3 \cdot 15 = 945$ experiments for each AFA method, repeated 10 times for static datasets and once for stream datasets.

For classification, a Support Vector Machine (SVM) is trained via stochastic gradient descent for most datasets, except for the **evenodd** datasets, where a Decision Tree (DT) is used, based on an optimized CART algorithm with a maximum depth of 5. The κ metric is used to evaluate the performance of each method, and statistical significance is determined through a Friedman-Test with a Nemenyi post-hoc test as recommended by [18]. All classifiers are evaluated in a prequential manner [28].

The framework extends the basic active feature acquisition framework of the previous chapter, which itself is an extension of the active stream mining framework from [73], and is implemented in Python. The classification algorithms are built using the *scikit-learn* library [65].

Table 11.3.: Descriptions of the datasets taken from [15]

Dataset	Instances	Labels	Cat. Features	Num. Features	Purpose
sea	60000	2	0	3	determine if two specific features exceed a threshold
electricity	45312	2	1	7	determine if electricity prices increase beyond a threshold
adult	32561	2	4	8	determine if annual income exceeds \$50,000
occupancy	20560	2	1	7	determine whether an office room is occupied
magic	19020	2	0	10	determine if signal is a gamma ray based on Cherenkov radiation
nursery	12960	5	8	0	determine rank of a child's nursery school application
pendigits	10992	10	0	16	determine digit written on a pad
evenoddf$\mathbf{x}$d$\mathbf{y}$	10000	2	0	$\mathbf{x}$	out of a set of $\mathbf{x}$ features, determine the outcome of an $\mathbf{y}$-wide even/odd function $(\mathbf{x},\mathbf{y}) \in \{(3,3),(5,3),(7,3),(9,2),(9,3),(9,4),(9,5)\}$
abalone	4177	3	0	8	determine sex of abalones

11.5. Results and Discussion

The performance of different acquisition strategies is first analyzed on the 8 regular datasets and 7 **evenodd** datasets separately, followed by a comparison aimed at understanding the significance of the quality and merit functions. The evaluation then continues with an analysis of the budget behavior of the IPF, concluding with a discussion on how the thresholds for the IPF were determined.

11.5.1. Performance Analysis on Regular Datasets

Across the regular datasets, all proposed AFA methods outperform the random baseline in the majority of scenarios. The best-performing strategy, however, varies based on the dataset, cost distribution, budget, and level of missingness. In low-budget settings, strategies like 2-Best, 1-Best, and 4-MaxMean dominate, but these strategies tend to leave a lot of budget unused in high-budget scenarios, as seen in Figure 11.5 versus Figure 11.6. In contrast, with higher budgets, strategies such as 4-Best and 4-GlobalBest catch up or surpass 2-Best, 1-Best, and 4-MaxMean, with 4-GlobalBest generally outperforming 4-Best.

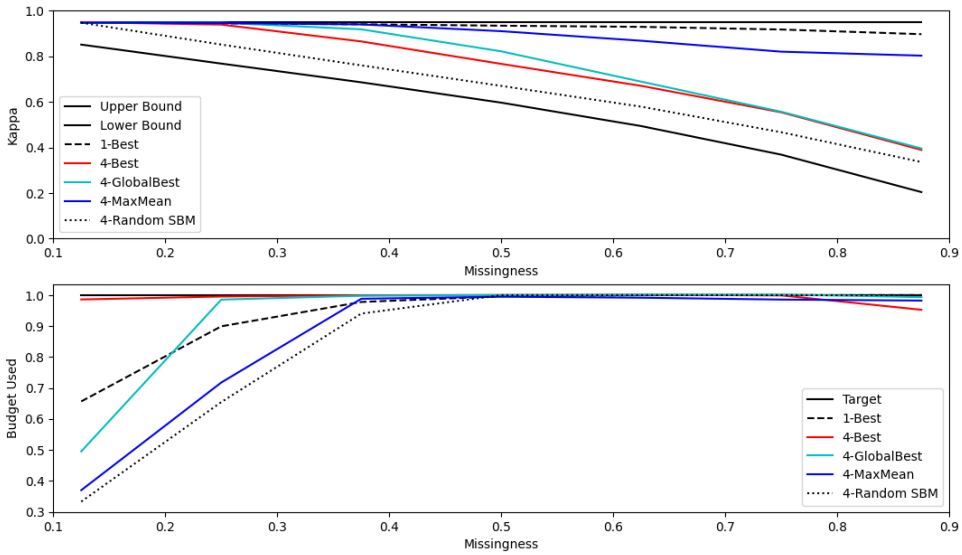

Figure 11.5.: In low-budget settings with a few highly predictive features, strategies that acquire fewer features per instance (1-Best, 4-MaxMean) excel by focusing on the best features for more instances. They also maintain better performance with increasing missingness, as they avoid spending on irrelevant features. This effect becomes more prominent as the missingness increases (x-Axis). Reproduced with permission from [15] ©2022 IEEE.

Interestingly, despite its simplicity, the 1-Best strategy ranks second overall in average performance across all datasets. This is attributed to the fact that in some cases, a single highly predictive feature has a disproportionately large impact on the classification outcome compared to larger feature sets.

The statistical analysis begins with a Friedman-Test, confirming that the performance of different AFA strategies significantly differs ($p = 2.24e - 145$). A Nemenyi post-hoc test for pairwise comparison reveals that, on non- **evenodd** datasets, 2-Best ranks highest, followed by 1-Best and 4-MaxMean. The 2-Best method performs

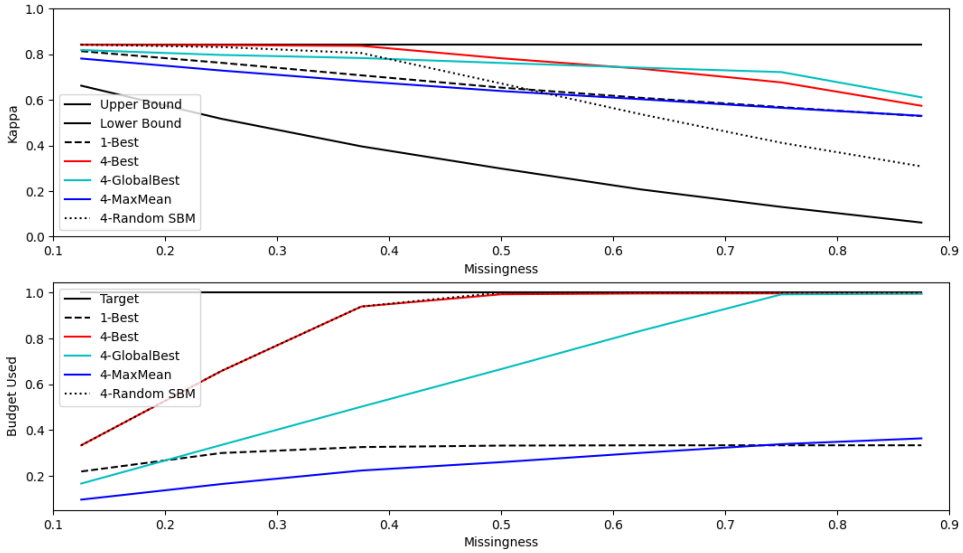

Figure 11.6.: Strategies with limited feature acquisitions per instance (1-Best, 4-MaxMean) fail to achieve optimal results in high-budget scenarios, despite having budget available. This effect becomes more prominent as the missingness increases (x-Axis). Reproduced with permission from [15] ©2022 IEEE.

significantly better than most other strategies, with the exception of 1-Best, suggesting that $k = 2$ is an optimal value given the combinations of dataset, budget, missingness, and cost distributions that were tested. Lower-ranked strategies, such as 4-Best, 100-Best, and 4-Random+SBM, perform worse, with the results being statistically significant, as illustrated in Fig. 11.7.

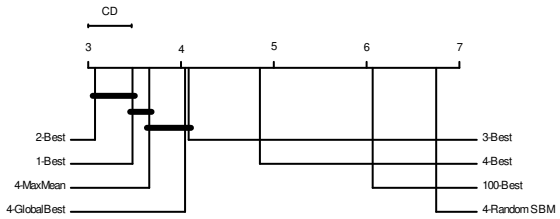

Figure 11.7.: Critical distance plot across all datasets except **`evenodd`** (CD=0.47). Reproduced with permission from [15] ©2022 IEEE.

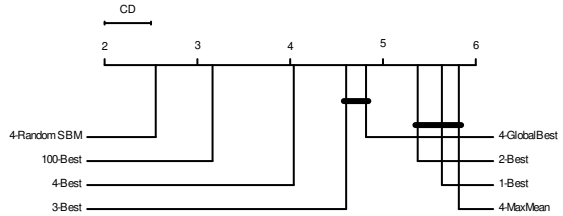

Figure 11.8.: Critical distance plot for **evenodd** datasets (CD=0.50). Reproduced with permission from [15] ©2022 IEEE.

11.5.2. Performance Analysis on Evenodd Datasets

To better understand the behavior of the proposed AFA strategies and eliminate the influence of single highly predictive features, the **evenodd** datasets were created — designed as a worst-case scenario—where the label depends on knowing all relevant feature values. Since AED relies on individual features for predicting label distributions, it struggles to prioritize features in these datasets, treating features more randomly. In this case, methods that acquire a greater number of dependent features tend to perform best.

As expected, the order of top-performing AFA strategies on the **evenodd** datasets differs from that on the regular datasets. Here, methods like 100-Best and 4-Best outperform 1-Best, 2-Best, and 4-MaxMean, which were dominant on other datasets, as shown in Figs. 11.7 and 11.8.

Statistical analysis of the **evenodd** datasets, conducted separately from the regular datasets, yields a Friedman-Test result of $p = 4.10e - 99$.

The **evenodd** experiments also highlight differences between using the IPF and a simple budget model. Notably, the random baseline AFA method ranks highest across these experiments, primarily due to its straightforward budget strategy, which acquires up to four features whenever budget permits, showing a bias toward acquiring smaller, cheaper feature sets that complete instances. In contrast, the quality gain provided by IPF prioritizes acquisition sets that favor very incomplete instances missing several key features.

Thus, on the **evenodd** datasets, the random baseline with a simple budget strategy performs significantly better than the other approaches. Among the k-Best strategies, a positive correlation emerges between the size of the acquisition set and performance, as shown in Fig. 11.8. The 4-MaxMean strategy performs the worst, largely due to its reluctance to complete instances and inefficient use of available budget. Meanwhile, 4-GlobalBest performs comparably to the more flexible 3-Best strategy.

11.5.3. Impact of Quality and Merit Functions

The experiments on the **evenodd** datasets reveal that single-feature-based merit functions, like AED, struggle to guide effective feature acquisition when the label is dependent on multiple features. Additionally, the current quality function sums up the merits of available features without considering whether those features are highly correlated. These limitations explain why the random baseline outperforms all our devised methods on the **evenodd** datasets. To address this, the aforementioned tree-based acquisition strategy is currently being developed, see section 13.1.

Table 11.4.: Percentage of runs where each strategy exceeded budget and average percentage of overspending on the 8 regular datasets. Table reproduced from [15].

Strategy	**% Runs**	**Avg. % over Budget**
1-Best	0.2	0.01
2-Best	0.4	0.02
3-Best	1.79	0.03
4-Best	3.17	0.06
100-Best	32.74	12.95
4-GlobalBest	23.41	0.22
4-MaxMean	7.34	0.16
4-Random SBM	0	0

11.5.4. Budget Usage

In analyzing budget usage, the focus is placed on overspending, which is particularly significant in real-world applications. The frequency with which each strategy exceeded its budget and the average amount by which it was exceeded were recorded, as shown in Table 11.4. This analysis covers only the 8 regular datasets. The baseline 4-Random strategy, which employs a simple budget manager (SBM), ensures that it never exceeds its budget, as it only acquires features when sufficient budget is available.

The methods using the IPF are more informative, as their behavior is influenced by the cost and quality of missing features as well as the degree of missingness. Most IPF-based methods stay within budget more than 90% of the time, with the exceptions being 100-Best and 4-GlobalBest. The 100-Best method, as expected, frequently overspends since it aims to acquire all features for an instance, which can quickly deplete the budget when expensive features are missing. The 4-GlobalBest method selects up to 4 of the globally best features by merit, which can also lead to overspending when the selected features are costly.

For all methods except 100-Best, the average overspend is less than 1%, whereas 100-Best exceeds the budget by an average of 12.95% in overspending scenarios. Given this, it can be concluded that 1-Best is a solid baseline for future experiments due to its strong performance and minimal budget overspend among IPF-based methods. Additionally the k-MaxMean strategy is recommended ($k = 4$ in our experiment) for its adaptability in adjusting feature set size dynamically and its performance, which rivals that of 1-Best.

However, one drawback of both 1-Best and k-MaxMean is their frugality in high-budget scenarios, where they leave large portions of their budget unused. This is evident in Figure 11.6, where both strategies underperform due to insufficient investment, even being outperformed by the random baseline up to certain levels of missingness. On the other hand, spending fewer features per instance can be advantageous when only a few features are highly predictive and the budget is low, as seen in Figure 11.5. For the `occupancy` dataset, only a few features are required for high classification performance, and thus 1-Best and 4-MaxMean show a slower decline in performance as missingness increases, avoiding wasteful spending on unnecessary features. In contrast, other strategies suffer a steady decline in performance as they acquire incomplete instances.

Dynamic strategies that adjust the number of features k purchased per instance according to the available budget present interesting avenues for future work, with the aim of optimizing budget usage and potentially enhancing classifier performance.

11.5.5. Threshold for the Incremental Percentile Filter

In the initial experiments, a conservative threshold T_{basic} based on a worst-case approach $A_{C_{max}}$ was compared with a threshold T_{basic}, which is based on the estimated average cost $\hat{A}_C$. Both methods initially applied basic threshold adjustments. The estimated average cost version often overspent its budget, while the conservative method stayed closer to the ideal budget usage. However, the conservative method sometimes underutilized the available budget, especially in cases of low missingness, which prompted the addition of a penalty term.

Figure 11.3 illustrates how both basic versions deviate from optimal budget usage, and how the penalty term improves adherence to budget limits. The penalty term ($p_c = 16$) was tuned using one dataset and it was found that this value generalized well to most other datasets. Future work should focus on developing a heuristic for setting the penalty term automatically, though it is recommended to start with this experimentally determined value.

11.6. Conclusion

This chapter introduces novel methods for active feature acquisition in data streams, considering real feature costs. Furthermore, several acquisition strategies are proposed, all of which can acquire multiple missing features per instance. This addresses the main limitations of the initial work presented in chapter 10.

The strategies presented either select the top k features with the highest merit for each instance or, in the case of the k-GlobalBest strategy, acquire up to k features from the globally top k features within the current stream window. Additionally, a dynamic strategy, k-MaxMean ($k = 4$), is proposed. This strategy adjusts the number of features purchased per instance to optimize the average instance quality, while limiting the acquisition to a maximum of k features.

To make these strategies practical, the Incremental Percentile Filter (IPF) has been adapted to work with absolute budgets, as it was originally designed for relative budgets. A method is also introduced for configuring and dynamically adjusting the IPF threshold using a penalty term to ensure optimal budget usage over time. The strategies are then compared against a random baseline, which acquires up to 4 random features. Experiments are conducted across 12 datasets, using three feature cost configurations (equal, decreasing, increasing) and seven levels of missingness.

The results show that the proposed strategies consistently outperform the random baseline, with most strategies staying within the defined budget in over 90% of cases. However, strategies utilizing the IPF occasionally exceed the budget in scenarios with high missingness and a very limited budget, an issue that requires further exploration.

In summary, this chapter offers a robust framework for active feature acquisition in data streams, considering varying feature costs and absolute budget constraints. Multiple acquisition strategies are proposed, and recommendations for their use in different contexts are provided. These promising initial results suggest further investigation into more advanced acquisition strategies and budget management techniques is warranted.

In this chapter, as well as the previous one, a feature importance metric was utilized that evaluates each feature independently. The experiments conducted on the **evenodd** datasets revealed that this type of metric encounters difficulties when guiding the acquisition process in cases where the label depends on multiple features.

Another limitation of such metrics is their tendency to lead to the acquisition of correlated features, especially if these features receive a high feature importance score. For instance, consider an extreme scenario where a duplicate of the most important feature is introduced into the dataset. The proposed AFA method would likely allocate significant budget to acquiring this duplicate, even if the original feature is already available. This, of course, would not result in better class separation but would instead waste the available budget.

The following chapter will address this inefficiency and explore strategies for reducing costs by selectively using imputation in certain situations.

12. Reducing Costs with Strategic Imputation

The previous chapter highlighted how certain feature importance metrics could misguide the acquisition process in the presence of correlated features, leading to unnecessary costs. This chapter examines the hybrid approach for handling missing feature values in data streams, as described in [16], using a two-stage process aimed at optimizing budget usage.

This chapter addresses the following research questions:

RQ3: How can Active Feature Acquisition be realized in a data stream setting?
RQ4: How can varying feature costs be considered during AFA on streams?

In the first stage, a subset of missing features from an instance is selected for acquisition based on their estimated merit. Feature merit measures the contribution of a feature to class separation [85, 32].

In the second stage, each feature in the selected acquisition set is evaluated to determine its predictability using the available features of the instance. If the prediction appears promising, imputation may be chosen over acquiring the missing feature to conserve the budget.

The primary contributions of this approach include a method that optimizes budget allocation by selecting imputation when feasible as an alternative to Active Feature Acquisition (AFA), as well as an imputation technique that monitors the predictive relationships between features.

The sections in this chapter are based on their counterparts in [16].

12.1. Related Work

In addition to the related work presented in the chapters 10 and 11, this chapter addresses related work concerning imputation.

Imputation refers to the process of replacing missing values with estimates based on statistical methods, information-theoretic approaches, or model-driven techniques [52, 1]. These techniques range from basic methods, such as substituting missing numerical features with the mean or using the nearest neighbors for imputation [39], to more advanced methods, including multiple imputation by chained equations [5] or deep learning-based imputation [33].

Imputation in data streams presents additional complexities compared to static learning, as new data is constantly arriving and must be integrated. Additionally, feature distributions may shift due to concept drift [87]. One approach to handle this continuous data flow is to use incremental models that update in real-time, avoiding the need for full retraining [66]. Another strategy involves focusing only on a sliding window of recent data points [23], which helps manage data growth and addresses concept drift by discarding outdated information. In this chapter, an imputation

method similar to [66] is proposed, using individual linear regression models for pairs of features, selecting the most suitable model for imputation. A windowed approach is also incorporated, as suggested in [23].

12.2. Methods

Algorithm 12.1 Simplified pseudo-code of the hybrid framework published in [16].

Require: Initial data X_{init}, A data stream consisting of batches X, a sliding window of batches W, a classifier C, a budget manager BM, an imputation model I, a feature importance metric afa, the cost of features C, a feature set selection method fss, an initial budget threshold T_{init}, the budget added for each instance B_{gain}

Ensure: $B_{spent} \leftarrow 0$, $B_{given} \leftarrow 0$, $B_{saved} \leftarrow 0$

1: add X_{init} to W
2: train initial model C_0 on W
3: **for** X_i in X, $i \geq 1$ **do**
4: update I with X_i
5: adjust budget threshold T
6: **for** x in X_i **do**
7: update merits using afa, W and x
8: $B_{given} \leftarrow B_{given} + B_{gain}$
9: get acquisition set A of x using fss
10: calculate quality gain of $x \cup A$
11: determine acquisition decision of BM according to quality gain of $x \cup A$
12: determine confidence decision of I given x
13: **if** (I) BM wants to acquire and I is confident **then**
14: $B_{spent} \leftarrow B_{spent} + A_C$
15: $B_{saved} \leftarrow B_{saved} + A_C$
16: **else if** (IV) BM wants to acquire and I is not confident **then**
17: $B_{spent} \leftarrow B_{spent} + A_C$
18: acquire A for x
19: **end if**
20: **end for**
21: impute remaining missing values of X_i using I
22: evaluate X_i using C_i
23: add imputed X_i to W
24: train new model C_{i+1} on W
25: **end for**

At the core of this chapter is an imputation model called the Feature Pair Imputer (FPI), which tracks the ability of each feature to impute other features. This allows the model to assess how accurately each missing feature can be predicted based on the available features. Algorithm 12.1 outlines the updated framework, and in line 12, it can be seen that, in addition to $quality_{gain}$, the confidence in the imputation model's ability to predict the missing features of an instance x is calculated. If there is high confidence in the imputation quality, the features are treated as if they were purchased, but the budget is saved and added to B_{saved}, as shown in lines 13 and 15. It is still added to the spent budget (line 14) to prevent the saved budget from being used on later instances, as the IPF threshold would otherwise adjust to allow

for more acquisitions.

Furthermore, the mean and most-frequent-value imputation methods are replaced with the FPI method, as indicated in line 21. The following section provides a detailed explanation of how the FPI functions and how it helps conserve the budget.

12.2.1. Feature Pair Imputer (FPI)

The Feature Pair Imputer (FPI) is designed to work efficiently in data streams by providing a fast mechanism to evaluate imputation quality. It maintains separate sliding windows W_{F_i,F_j} (denoted as $W_{i,j}$ for simplicity) with a size of w_{FPI} for each feature pair (i, j), as well as for each individual feature (i, i), resulting in a total of $\sum_{i=1}^{|F|} i$ windows. These sliding windows store feature value pairs and train two imputation models, $M_{i,j}$ and $M_{j,i}$, for each pair. These models are also evaluated using the data stored in the sliding windows.

The imputation model and error calculation method depend on the types of input and output features, as summarized in Table 12.1. For self-imputation, a simple mean imputer is used for numeric features, while a mode imputer is used for categorical features, both using the relevant error metrics shown in Table 12.1.

Feature In	Feature Out	Imputation	Error
numeric	numeric	linear regression model	$\frac{RMSE(M_{i,j})}{max(W_{i,j})-min(W_{i,j})}$
categorical	numeric	mean of the posterior distribution of output values given the input value	$\frac{RMSE(M_{i,j})}{max(W_{i,j})-min(W_{i,j})}$
categorical	categorical	mode of the posterior distribution of output values given the input value	$1 - Jaccard(W_{i,j}[:, j], M_{i,j})$
numeric	categorical	1-d nearest neighbour mapping	$1 - Jaccard(W_{i,j}[:, j], M_{i,j})$

Table 12.1.: Methods used for feature pair imputations and their error calculation methods. Table reproduced from [16].

The imputation errors are stored in an error matrix $E \in [0, 1]^{|F|\times|F|}$, where rows represent input features and columns represent output features. Therefore, $E_{i,j}$ corresponds to the imputation error of model $M_{i,j}$.

Given the error matrix E and a vector mask $known(x) = \{1, 0\}^{|F|}$ representing the known features of instance x, where $known(x)_i = 1$ if feature i is known, the contribution of imputation models to imputing the missing features of x can be estimated by creating a weight matrix based on the errors. The known features of x are referred to as K_x. The weight matrix, $weight$, is computed using the reciprocal of the error values as follows:

$$weight_{i,j} = \begin{cases} (E_{i,j} + \epsilon)^{-1} & if\ \exists a \exists b E_{a,b} = 0 \\ E_{i,j}^{-1} & otherwise \end{cases} \tag{12.1}$$

This ensures that imputation models with lower errors contribute more to the imputation process. A small constant $\epsilon = 0.001$ is added to all error values if any $E_{a,b} = 0$, to avoid division by zero.

Using the weight matrix $weight$, the known feature mask $known(x)$, and the predicted values from the models, the missing values for x can be imputed. A specific weight matrix Z for instance x is computed as:

$$Z_{i,j} = weight_{i,j} \cdot known(x)_j + weight_{i,j} \cdot I_{i,j} \tag{12.2}$$

where I is the identity matrix of size $|F|$. The left side of the equation is applied when $i \neq j$, and the right side is used when $i = j$.

Let X' be the matrix of imputed values from the models M, given the known feature values of x. Depending on the importance strategy, the imputed value for the missing feature j in x, denoted as $\hat{f}_j$, can be computed using one of the following methods:

Weighted Approach

$$\hat{f}_j = \begin{cases} \frac{\sum_{i=1}^{|F|} Z_{i,j} \cdot X'_{i,j}}{\sum_{i=1}^{|F|} Z_{i,j}} & if\ isNumerical(F_j) \\ X'_{m,j}, m = argmax_{1 \leq i \leq |F|} Z_{i,j} & otherwise \end{cases} \tag{12.3}$$

Choose Best Approach

$$\hat{f}_j = X'_{m,j}, m = argmax_{1 \leq i \leq |F|} Z_{i,j} \tag{12.4}$$

In this chapter, the focus is on the *choose best approach*, where missing values are imputed using the model with the lowest error.

12.2.2. Feature Pair Imputer Threshold Skip (FPITS)

The FPI framework quickly calculates a confidence value for imputing the missing features of an instance based on the known features. Before performing any calculations, the system distinguishes between three cases depending on the number of known features for an instance x:

$$FPI_{conf}(x) = \begin{cases} 0 & \text{if } known(x) = 0 \\ 1 & \text{if } known(x) = |F| \\ 1 - \max(Miss_{min}(x)) & \text{otherwise} \end{cases} \tag{12.5}$$

For instances with at least one missing feature and one known feature, the FPI confidence FPI_{conf} is calculated using the highest reconstruction error among the missing features, representing the potential error in imputation. Given the set of known features K_x for an instance x and the current error matrix E, the FPI computes the ease of imputing x's missing values.

For each missing feature j in x, the corresponding error vector V_j is extracted from E, which contains all the errors $E_{i,j}$, where $i \in \{K_x \cup j\}$. These represent the errors of models that could be used for imputation. According to the "choose best approach" described in equation 12.4, the smallest error for each missing feature is selected. These minimum errors are stored in a set $Miss_{min}(x)$:

$$Miss_{min}(x) = \bigcup_{j \notin K_x}^{F} \min(V_j)$$

The maximum of these errors reflects the worst-case error for imputing x. Therefore, the FPI confidence is calculated as:

$$FPI_{conf}(x) = 1 - \max(Miss_{min}(x))$$

A value closer to 1 indicates a high confidence in the FPI's ability to accurately impute the missing features for x.

This confidence value is integrated into the decision-making process alongside the framework's budget manager. A new decision-making method, named Feature Pair Imputer Threshold Skip (FPITS), further refines the acquisition process. FPITS imposes an additional constraint: it allows the acquisition of a feature set A for instance x only if both the budget manager's IPF decides to acquire A and the FPI confidence threshold deems x difficult to impute (line 12).

FPITS is designed to reduce budget consumption by avoiding unnecessary acquisitions in cases where a feature set is considered highly informative but the FPI is confident that the missing features can be imputed accurately. This additional constraint ensures more efficient budget usage in scenarios where imputation is highly reliable, for instance, in the case of highly correlated features.

12.3. Experiments

To evaluate the effectiveness of the proposed method, multiple experimental runs were conducted using the proposed framework[1]. The following sections provide an overview of the datasets used, outline the framework's hyperparameters, and present the experimental results and discussion.

12.3.1. Datasets

Dataset	Instances	Labels	Cat. features	Num. features
`electricity`	45312	2	1	7
`adult`	32561	2	4	8
`magic`	19020	2	0	10
`cfpdss`	13000	2	5	5
`nursery`	12960	5	8	0
`pendigits`	10992	10	0	16

Table 12.2.: Datasets used in our experiments. Table reproduced from [16].

The majority of the datasets are the same as described in section 10.5. They including four static datasets (`adult`, `magic`, `nursery`, and `pendigits`) and two data streams (`cfpdss` and `electricity`). Both `magic` and `pendigits` consist solely of numerical features, while `nursery` contains only categorical features. The datasets `adult`, `cfpdss`, and `electricity` include a mix of categorical and numerical features. The method was tested on static datasets to validate the imputation model's performance in the absence of concept drift.

The differences between datasets are important to note, as mixed and categorical features tend to have a negative impact on the performance of FPI, which requires careful evaluation. All datasets except `cfpdss` are available through the UCI repository [40].

The `cfpdss` dataset is a synthetic dataset generated for this work, designed with various feature-to-feature correlations and shifts in feature and label generation every thousand instances to simulate concept drift, including incremental, gradual, and sudden changes. These drifts are synchronized with specific types of feature

[1] `https://github.com/Buettner-Maik/caafa-stream`

correlations, meaning that only features involved in a particular correlation are affected by concept changes. This setup allows the FPI and its imputation models to be evaluated effectively.

The correlations in the data include linear relationships between numerical features with slight noise, numeric-to-bicategorical correlations based on threshold values, and linear combinations of features influencing another feature's value, also with added noise. The label function is determined by three numerical features, each linearly correlated with another feature, while one categorical feature is also correlated with other features. This setup allows for accurate imputation using the FPI's linear regressions and other models, as missing features that strongly predict the label, can often be imputed using a known correlated feature. This dataset replaces the previously used `gen` and `evenoddfxdy` datasets from earlier works, and more details are available in the online repository.

12.3.2. Experiment Parameters

Incomplete data streams were generated according to the type of dataset being used. For static datasets, the data was shuffled, and an initial batch consisting of 50 instances plus one example for each label was created. For stream datasets, the order of instances was maintained, with the first 50 instances forming the initial batch. The remaining data was split into batches of 50 instances each. Based on the missingness parameter m, features of the instances were randomly infused with missing values in a completely at random manner. This process was repeated 10 times across 7 levels of missingness, resulting in 70 distinct permutations of data streams for each dataset.

The classification model used in the experiments was a Support Vector Machine (SVM) with default parameters from the sklearn library. An initial classifier was trained on the first batch (line 2), and a new classifier was trained after each acquisition and imputation step (line 24); see Algorithm 12.1. As in the other chapters a prequential evaluation method was used [29]. For acquisitions, the single-window average Euclidean distance function was used to calculate feature merit [85], and a budget manager in the form of an incremental percentile filter with a window size of $w_{IPF} = 50$ was employed, along with a 4-Best feature set selection strategy; see section 11.2. The cost of each feature was uniformly set to 1.

The FPI used the *choose best approach* for imputation with a feature pair window size of $w_{FPI} = 25$. Thresholds for the FPITS method were selected from the values 0, 0.1, 0.2, ..., 0.9, and 1.0. A threshold of 0.0 corresponds to a method without FPITS, so AFA-only, while a threshold of 1.0 means no feature values are acquired, so every missing value is imputed. The IPF used by the FPI had a window size of $w_{IPF\_FPITS} = 100$, meaning the window was fully updated every two batches.

To contextualize the performance results, two reference methods are used: a lower bound, representing minimal performance when no budget is spent, and an upper bound, representing maximum performance when all feature values are acquired.

Data set	p-value	FPI mean rank	SI mean rank
`adult`	< 0.001	1.896	**1.104**
`cfpdss`	< 0.001	**1.246**	1.754
`electricity`	0.717	1.529	1.471
`magic`	0.371	1.504	1.496
`nursery`	< 0.001	1.739	**1.261**
`pendigits`	< 0.001	**1.046**	1.954

Table 12.3.: Comparison of Feature Pair Imputer vs. Simple Imputer using Wilcoxon test results for each dataset, with 280 paired sample points per dataset (7 missingness levels, 4 budgets, and 10 iterations). Winning strategies are highlighted in bold. Table reproduced from [16].

12.4. Results

The performance of the FPI was evaluated in comparison to the simple imputers discussed in chapters 10 and 11. This evaluation was followed by a detailed analysis of the FPI's behavior under different conditions. Subsequently, the challenges of working with tight budget constraints were addressed, and finally, an exploration of the potential budget savings achievable with the FPI was conducted, while maintaining similar classification performance.

12.4.1. FPI Performance

To assess the effectiveness of the FPI and its imputation models, comparisons were made against the Simple Imputer (SI), which relies on mean and mode imputation methods. These comparisons were carried out across several datasets, incorporating seven missingness levels ($m = \{0.125, 0.25, \ldots, 0.875\}$), four budget settings ($B_{gain} = \{0, 0.5, 1, 2\}$), and 10 iterations per configuration, leading to a total of 280 comparisons for each dataset. A Wilcoxon test ($\alpha = 0.05$) was performed to determine whether there were statistically significant differences between the two methods. Additionally, method rankings were used to identify which approach performed better on each dataset. The results, along with the corresponding p-values, are presented in Table 12.3.

The FPI exhibited superior performance over SI on the `pendigits` and `cfpdss` datasets, which is expected for the `cfpdss` dataset due to its strong linear correlations between features. The better results on the `pendigits` dataset suggest a notable linear dependence between features, and the static nature of the dataset further contributed to FPI's advantage. However, when missingness levels increased, the FPI took more time to adjust to new patterns in the data.

In contrast, SI outperformed FPI on the `adult` and `nursery` datasets. The results from the `nursery` dataset highlight a limitation of FPI in accurately predicting categorical features when only one other categorical feature is available. The largest gap in performance was observed on the `adult` dataset.

For the `electricity` and `magic` datasets, no significant difference was observed, with both methods showing comparable performance.

12.4.2. FPITS Behavior

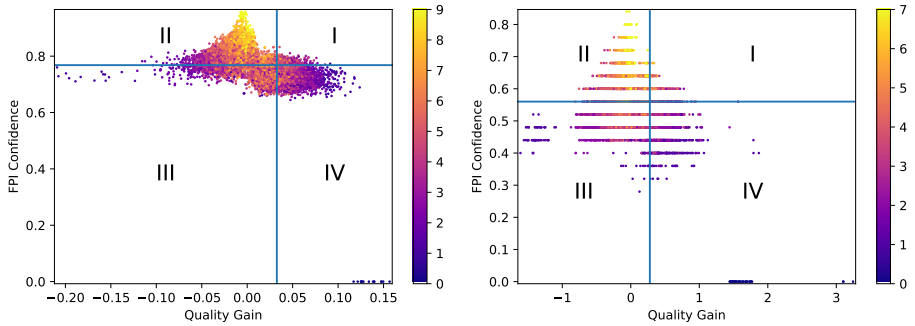

Figure 12.1.: Scatter plots of the decision variables seen by both decision makers. The left plot shows data set *magic* and the right data set *nursery* each on a single missingness $m = 50\%$ run and the number of known features of an instance encoded in the color. Figure reproduced with permission from [16] ©Springer Nature 2023.

To better understand the operation of the FPITS method, consider Figure 12.1, which shows the quality gain values of the budget manager for the selected acquisition set A and the FPI's confidence values FPI_{conf} for its corresponding instance x on single runs for the **magic** and **nursery** datasets. Each point in this figure represents a single instance, and its position indicates whether the instance's features should be acquired or imputed.

The vertical and horizontal lines represent the mean decision thresholds for the budget manager and FPI, respectively, dividing the graphs into four quadrants. Instances in quadrants Q_I and Q_{IV} (which meet the threshold on the x-axis) are candidates for feature acquisition, while instances in quadrants Q_{II} and Q_{III} will only be imputed. Whether an instance in Q_I will be imputed or selected for acquisition (Q_{IV}) is determined by the FPI based on its threshold.

The ratio of values in Q_I to the total in Q_I and Q_{IV} indicates how much budget is conserved.

It is worth noting that during the execution, the thresholds shift according to recent stream history. In the right-hand plot for **nursery**, which contains only categorical features, the nature of the categorical imputation models results in well-defined ratios when calculating FPI_{conf}. The combined distribution of quality gain and FPI_{conf} values determines how well the user-defined FPI threshold translates into budget conservation. As the FPI's confidence increases with the number of known features in an instance, FPI_{conf} positively correlates with the number of known features. Meanwhile, the averaging nature of the quality function causes changes in quality to become less pronounced as more features are known. As a result, the quality gain values for the acquisition sets tend to cluster around the mean on the x-axis. This relationship is visualized through the color bar, which represents the number of known feature values in an instance. Most instances with many known features (lighter colors) appear in Q_{II}, meaning they are not selected for acquisition and do not require FPITS intervention.

FPITS in budget-constrained scenarios

In scenarios with increasingly constrained budgets (i.e., high levels of missingness and low budgets), it is expected that the proposed method will utilize a greater portion of the available budget. This occurs because the distributions of FPI_{conf} and quality gain values shift towards quadrant IV (Q_{IV}), and as more features become missing, more instances are mapped to Q_{IV}. Since the ratio of skipped acquisitions is represented by $|Q_I|/(|Q_I \cup Q_{IV}|)$, this shift towards quadrant IV results in fewer skipped acquisitions.

This behavior is intentional, as higher levels of missing data typically necessitate the acquisition of more features to compensate for the lack of information. Figure 12.2 illustrates this effect by showing the relationship between B_{spent} and the missing feature probability m for the **pendigits** and **nursery** datasets.

However, an adverse effect is observed when missingness levels become extremely high, and thresholds are set at elevated levels. In such cases, the number of instances with all features missing increases, leading to a rise in instances where $FPI_{conf} = 0$. When the FPITS' IPF window becomes saturated with these zero values, i.e., when more than $w_{IPF\_FPITS} \cdot thr$ values in the window are equal to zero, the IPF will compare the FPI_{conf} of incoming instances to a threshold of 0. As any incoming value will be greater than or equal to zero, the IPF will skip acquisitions.

In extreme cases, this situation can persist indefinitely, causing the FPITS to continuously skip acquisitions, which leads to a sharp decline in budget expenditure. This is evident in Figure 12.2, where the lowest line on the left and the three lowest lines on the right indicate this behavior at higher thresholds.

The expected minimum critical threshold, at which the IPF will begin evaluating incoming instances with a comparison value of 0 for a given missingness probability $m \in [0, 1]$, can be computed as follows:

$$thr_{critical} \leq 1 - m^{|F|} \tag{12.6}$$

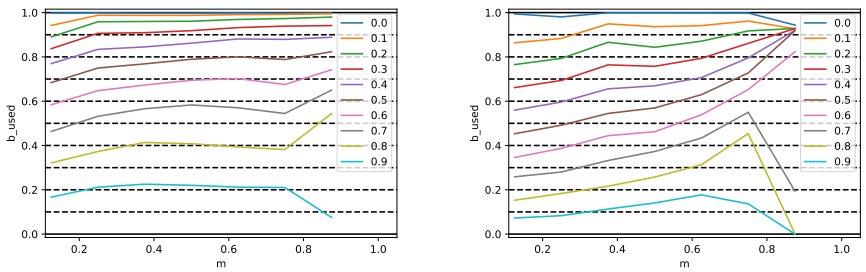

Figure 12.2.: Budget spent by various FPITS methods with *thr* values ranging from 0.0 (top) to 0.9 (bottom) on the **pendigits** dataset (left) and the **nursery** dataset (right) at $B_{gain} = 1$. As missingness increases, budget expenditure rises, except when IPF becomes saturated with zeros, causing a sharp drop in spending (lowest line on the left and three lowest lines on the right). Figure reproduced with permission from [16] ©Springer Nature 2023.

12.4.3. Budget Comparison at Similar Performance

To determine whether the proposed method can conserve budget while maintaining comparable performance, a comparison was conducted using the Friedman test, followed by post-hoc Nemenyi tests. The prequential classification accuracy was compared across batches and iterations for 11 FPITS methods, with thresholds ranging from 0.0 to 1.0 with increments of 0.1, and their performance was ranked accordingly. An FPITS method with $thr = 0$ is equivalent to the previous SWAED method using FPI as the imputation model, as acquisitions are always triggered and no candidate set is imputed. Conversely, an FPITS method with $thr = 1.0$ represents the lower bound method, where every acquisition candidate set is imputed. The comparison was based on ten iterations, and the number of batches (i.e., 10 iterations times $(13000 - 50)/50 = 259$ batches), with the Friedman test applied to assess significant differences in performance among the methods. When significant differences were detected, the post-hoc Nemenyi test was used to further analyze the rankings by calculating the mean ranks of the methods and the critical distance at a confidence level of $\alpha = 0.05$. For two methods to exhibit significantly different performance, their mean ranks must be separated by a distance greater than the critical distance. Intuitively, it would be expected that methods spending the highest budget would achieve the best performance, resulting in the lowest ranks. As the threshold increases, less budget is spent, leading to a deterioration in relative rank (i.e., higher ranks). If the Nemenyi test shows that methods with higher thresholds perform similarly to the method with $thr = 0$, the corresponding budget expenditures can be compared to identify the method that conserved more budget while still making effective acquisition decisions. All Friedman tests returned p-values below $2.2 \cdot 10^{-16}$, validating the results of the post-hoc Nemenyi test, as shown in Figure 12.3. The plots, generated using the autorank package [35], illustrate the critical distance for the relevant results. A summary of these findings is provided in Table 12.4. In one scenario, the method with $thr = 0.4$ achieved comparable performance to an AFA-only method on the **adult** dataset, but required only 69% of the original budget per batch to do so. Across various degrees of missingness ($m = \{0.25, 0.5, 0.75\}$ and $B_{gain} = \{0.5, 1.0, 1.5, 2.0\}$), budget savings ranged from 1% on the **nursery** dataset to as much as 27% on the **adult** dataset, all while maintaining similar classification performance.

	adult	**cfpdss**	**electricity**	**magic**	**nursery**	**pendigits**
cd	0.187	0.297	0.159	0.245	0.297	0.323
$rank(thr = 0)$	5.714	5.408	5.246	5.434	3.305	4.927
thr_{sim}	0.4	0.3	0.1	0.5	0.0	0.3
$rank(thr_{sim})$	5.891	5.655	5.322	5.678	3.305	5.175
B_{saved}	31%	23%	4%	15%	0%	8%

Table 12.4.: Overview of the relevant information of cd plots in Figure 12.3 with $m = 0.5$ and $B_{gain} = 1$. The rank of the method without imputation ($rank(thr = 0)$) is compared to the method that saves the most budget while still performing similarly ($rank(thr_{sim})$). The amount of budget saved is depicted in the last row. Table reproduced from [16].

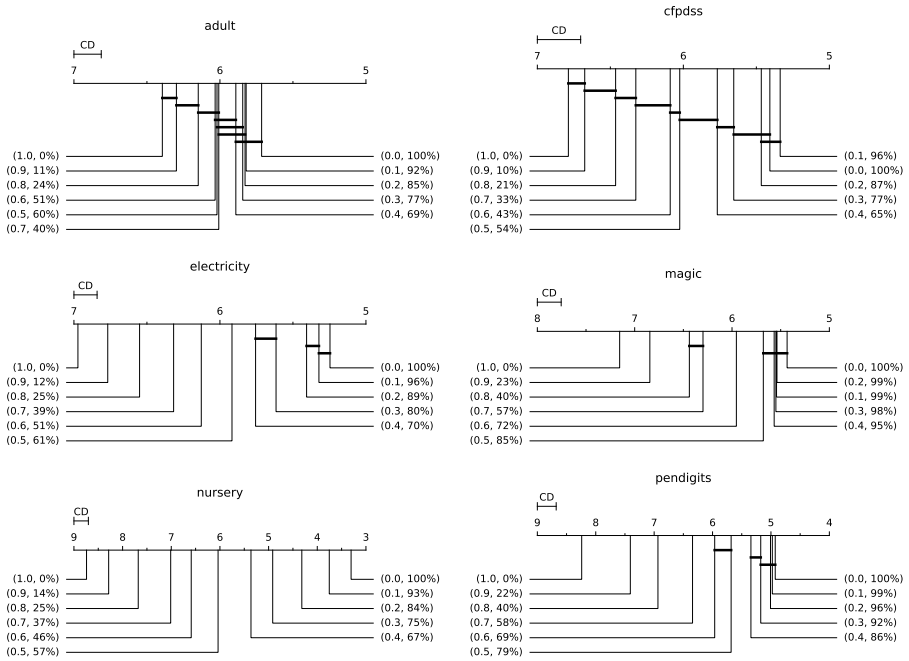

Figure 12.3.: CD-plots for six datasets with $m = 0.5$ and $B_{gain} = 1$. Tuples indicate the threshold and percentage of budget spent relative to the method without FPITS. The original method without imputation (threshold = 0.0, budget = 100%) appears on the right with the lowest rank (except for *cfpdss*), while the imputation-only method (threshold = 1.0, budget = 0%) is on the left with the highest rank, indicating worse performance. Models with similar performance are linked by vertical bars. For example, on *electricity*, the model at threshold = 0.1 performs comparably to the original method, spending 96% of the budget and saving 4%; see Table 12.4. Figure reproduced with permission from [16] ©Springer Nature 2023.

12.5. Conclusion

The previous chapters highlighted the issue of budget wastage in datasets with highly correlated features. In this chapter, a hybrid approach was introduced, combining AFA for data streams with an intelligent imputation mechanism designed to handle feature-incomplete instances. This mechanism helps avoid expensive feature acquisitions when there is a strong likelihood that the missing features can be accurately imputed using the available data.

Additionally, a novel, lightweight imputation model was proposed, capable of self-evaluation even in very fast streams. Its plug-and-play nature allows it to flexibly adapt to various simple feature-to-feature correlations.

Experiments were conducted across six different datasets, each with varying budget constraints, levels of missingness, and targeted budget savings. The results demonstrated that in terms of performance, the approach was comparable to methods relying exclusively on AFA, while saving on average between 1% and 27% of the budget, depending on the dataset.

Several limitations of the feature pair imputer were identified during the experiments, which are intended to be addressed in future work. This includes incorporating

more advanced pairwise models, such as stream-applicable Gaussian and deep regression models, that are not limited to linear correlations. Additionally, improved models are needed for handling categorical features, as the current methods were often surpassed by simple mode imputation.

A significant challenge remains in identifying a unified error metric capable of evaluating imputation losses for both numeric and categorical features or in mapping these errors into a common space. One possible approach involves discretizing numerical features into buckets, so that the categorical error metric can universally applied or rescaling numeric and categorical errors to make them more comparable.

The next chapter will present additional reflections on AFA on data streams before moving to the conclusion of part II.

13. Additional Reflections on Active Feature Acquisition

This chapter contains additional reflections with regards to handling feature correlations and providing cost-sensitive performance bounds.

13.1. Generation of Tree-Based Acquisition Sets

One of the discussed shortcomings of using independent feature importance metrics like Average Euclidean Distance (AED) is that it can lead to the acquisition of superfluous features in case they are correlated. As the total number of possible feature combinations is 2^n, it becomes obvious that it is infeasible to calculate all feature **set** importance scores, especially in high dimensional settings.

It would, however, be feasible to co-train multiple stream-based decision trees [32] or streaming forests [31]. The branches of a decision tree inherently avoid feature correlations and give concrete feature expressions, making them ideal candidates for generating potential acquisition sets.

The solutions that are currently being investigated and prepared for publication use the branches of the decision trees to generate acquisition set candidates. For each branch, it extracts which features were used on the path from the root to the leaf node while making sure that the feature conditions on the branch match the known features of an instance with missing features. If a branch is eligible according to the feature expressions of our instance, then score the branch candidate C_b according to some metric, e.g.

$$\text{Score}(C_b) = \frac{\text{Leaf Purity}(C_b)}{\text{TotalAcquisitionCost}(C_b)}$$

The score of the best candidate branch can then be presented to the IPF to decide for or against acquisition; see Algorithm 13.1.

Such an approach will focus on the most important features according to the decision tree, avoid acquiring correlated features and gives control over the maximum acquisition set size as the allowed depth of a decision tree can be specified by the users. Early results from ongoing work where the feature with the highest score (according to AED) was duplicated indicate that it successfully avoids acquiring correlated features, but that the scoring of the branches requires further refinement.

Algorithm 13.1 Tree-Based Feature Acquisition

Require: Data stream S with instances x, features F, missing features M, known features K, branches B of the streaming trees

Ensure: Candidate acquisition set with maximum score for each instance

1: **for** each instance x in data stream S **do**
2: **for** each branch b in B **do**
3: Generate candidate acquisition set C_b from branch b, consisting of features $f_b \in b$
4: **if** any feature values in K contradict feature values in C_b **then**
5: Remove candidate C_b
6: **end if**
7: **end for**
8: **for** each retained candidate C_b **do**
9: Calculate the total cost of missing features:

$$\text{TotalCost}(C_b) = \sum_{m \in M} \text{Cost}(m)$$

10: Score each retained candidate C_b, using:

$$\text{Score}(C_b) = \frac{\text{Leaf Purity}(C_b)}{\text{TotalCost}(C_b)}$$

11: **end for**
12: Select the candidate acquisition set $C_{\max}$ with the maximum score
13: Pass the score of $C_{\max}$ to the Incremental Partial Forest (IPF) to decide whether to acquire the features in $C_{\max}$
14: Process next instance in data stream S
15: **end for**

13.2. Realizing a Cost-Sensitive Performance Bound using Genetic Programming

The proposed frameworks show an upper and lower performance bound where either all missing features are acquired, or all missing features are imputed. While it gives an effect range for possible AFA improvements, it is not satisfactory in assessing how effective an AFA strategy was given the budget it spent on acquisitions. Calculating the globally optimal acquisition at a given budget is too computationally expensive, so a locally optimal solution is being proposed.

This solution treats the stream as one static object and relies on a genetic algorithm that creates random acquisition candidates under budget constraints and then refines them iteratively using mutation and crossovers. To imagine the stream as a static object, consider Figure 13.1, which shows the whole stream with missing features at the top and two acquisition candidates that each acquire 3 features below.

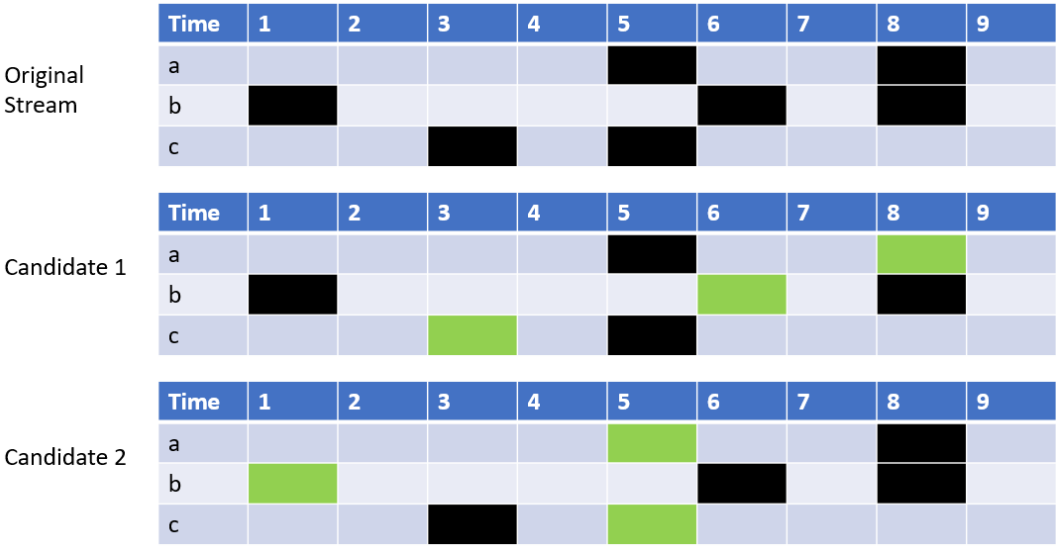

Figure 13.1.: **Top:** Whole stream with 9 timepoints and missing features colored in black. **Middle and Bottom:** two acquisition candidates which each acquire 3 features.

Here is a brief description of each component of the algorithm.

Fitness Function: The fitness function in this case is the cumulative prequential classification performance across the stream given a prediction Model M, which was employed on a solution candidate C_i.

Budget Balancer BB: As mutation and crossover lead to new acquisition candidates, it has to be made sure that each resulting candidate still respects the budget constraints as much as possible. This is achieved by iteratively removing features flagged for acquisition until the acquisition costs do not exceed the given budget constraint. In case of unspent budget, it iteratively picks features randomly for acquisition, as long as adding their feature costs will not exceed the given constraint. This guarantees that a candidate stays equal or slightly below the given budget constraint.

Mutation: This Is realized by randomly selecting a percentage feature flagged for acquisition (green) and features flagged for imputation (black) and swapping their flags. Afterward, the BB is applied to the resulting candidates to ensure budget conformity.

Crossover: Is realized by picking two candidates and cutting them vertically at randomly chosen timepoint and then swapping their end parts. This will almost always require the BB to ensure budget constraints are respected.

Initial results show that this approach does increase the classification performance (fitness) and delivers locally optimal solutions; see Figure 13.2.

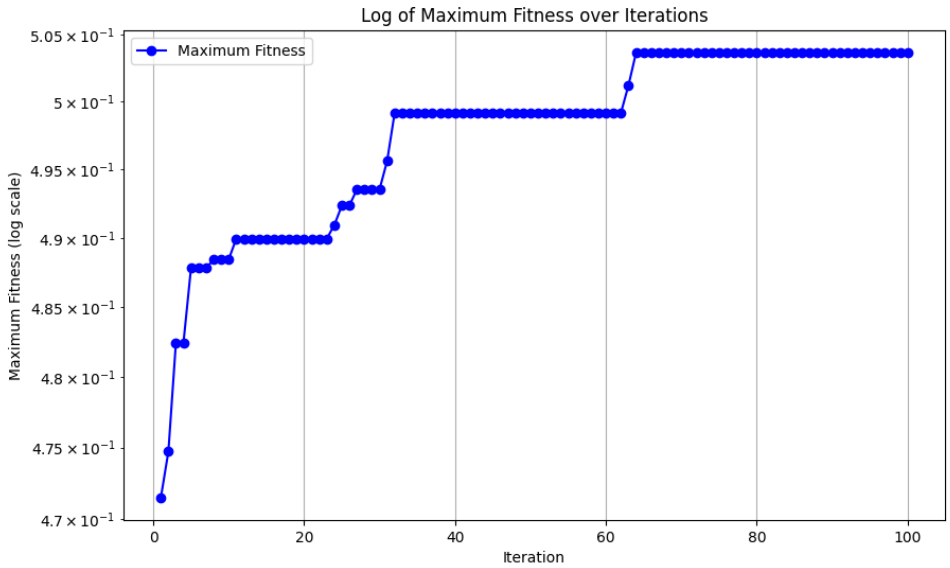

Figure 13.2.: Example of GA on the `cfpdss` with 50% of the features missing, showing how the classification performance (fitness) of a Hoeffding-Tree increases as the acquisition candidates evolve over time.

14. Active Feature Acquisition on Data Streams: Discussion and Conclusion

Part II of this thesis explores strategies to address missing data in the feature space of data streams, focusing on Active Feature Acquisition (AFA) and imputation techniques. The contributions to each research question in this part are detailed below, followed by a discussion on open challenges and future directions.

RQ3: How can Active Feature Acquisition be realized in a data stream setting? One of the first methods to implement AFA in data streams is presented in chapter 10, where a feature importance metric is combined with a budgeting mechanism to enable stream-based AFA. In that chapter, the assumption is that feature costs are equal, and the acquisition of missing features is limited to one per instance. These constraints are relaxed in chapter 11, where multiple acquisition strategies capable of acquiring up to k features are introduced. Experimental results demonstrate that these methods outperform a random acquisition baseline across most datasets.

RQ4: How can varying feature costs be considered during AFA on streams? To incorporate varying feature costs, the budgeting mechanism from chapter 10 is adapted by making the internal decision threshold dynamic. Chapter 11 presents two ways to set the initial threshold, one based on the average cost and another on the maximum cost, which assumes all features of an instance are missing. The threshold is then adjusted dynamically to account for budget overspending or underspending. Experiments with different feature costs confirm that the proposed methods outperform random baselines in all cases, except for datasets with interdependent features, where performance was expected to decline due to the feature importance scoring, which considers each feature independently. The work is extended in chapter 12 by combining AFA with strategic imputation through the feature-pair-imputer (FPI), which uses pairwise imputation models to predict missing features, saving acquisition costs when appropriate. Results indicate that this hybrid approach can reduce budget consumption while maintaining comparable performance to an AFA-only approach.

Furthermore, frameworks for conducting AFA experiments and generating datasets are provided to facilitate the testing of AFA strategies under specific conditions. These frameworks allow for:

- Developing and testing new stream-based AFA and imputation methods under varying budgets and degrees of missing data.
- Comparing AFA strategies against random baselines, lower bounds where all missing values are imputed, upper bounds with no missing features, and against each other, including statistical significance tests.
- Generating datasets to assess performance on concept drift, data with interdependent features and data with highly correlated features.

14.1. Limitations

The presented work has limitations, which we will briefly list here for clarity and revisit in the following section on open questions and future directions.

- The proposed feature importance metrics consider each feature independently, which can lead to problems when certain sets of features are needed to predict a label or when features are highly correlated so that acquiring one of them would be sufficient.
- Using the IPF as a budget manager can lead to under and overspending in certain extreme situations, which should be addressed in the future.
- The FPI showed a subpar performance when applied on categorical features, emphasizing the need for different pair-wise imputers when categorical features are involved.
- While most experiments are conducted on static datasets and synthetic data streams, these provide a strong foundation for testing. Expanding to more diverse, real-world streaming datasets offers potential for further validation.

14.2. Open Questions and Future Work

Challenges in this section are not separated by chapter, as they mostly apply across all three chapters: 10, 11, and 12.

All three AFA chapters rely on merit functions that do not account for inter-feature dependencies, leading to suboptimal performance when interdependent features influence the label of an instance (see Section 11.5.2). An ongoing effort is the development of an AFA method that generates acquisition set candidates using decision trees, ensuring that highly predictive features are not acquired if a correlated feature has already been selected, see section 13.1.

Another challenge is the absence of a performance bound that represents the optimal acquisition strategy given a specific budget. Current work addresses this by applying genetic programming to generate a locally optimal acquisition baseline based on the available budget, see section 13.2. Both of these extensions are being prepared for publication.

Further research will evaluate the performance of the proposed methods on diverse datasets, using different classifiers and feature importance functions beyond AED. Additionally, efforts will be made to improve the comparability of imputation loss across categorical and numerical features by developing error functions with similar value ranges and scales. Another aspect concerning imputation is the improvement of the FPI concerning the imputation quality of categorical attributes.

15. Overall Conclusion

This thesis approaches the study of data streams from two distinct perspectives. The first, termed the horizontal view, focuses on the data space and examines how the entity-instance relationship within a dataset can be utilized to assign data to entity-centric models. The goal of this approach is to capture entity-specific characteristics that might be overlooked by an entity-ignorant model, thereby enhancing the quality of predictions.

The second perspective, referred to as the vertical view, concerns the feature space of the data stream. It specifically investigates how missing features can be acquired under budget constraints, with the objective of improving the performance of a prediction model.

Both parts of the thesis share a common aim: to enhance prediction performance while maintaining efficiency. For entity-centric learning, this involves managing memory usage, and for active feature acquisition, it relates to optimizing budget expenditure. Each part addresses two research questions, with one question evaluating the effectiveness of the proposed approach and the other assessing its efficiency.

There is an inevitable overlap with the concluding chapters of each individual part (8, 14), although those chapters provide a more detailed discussion of limitations and future directions. In contrast, this chapter focuses on offering a brief summary and a high-level perspective.

The findings of each part are now summarized and connected to the research questions outlined at the beginning of the thesis. Following this, more ambitious directions for future work are proposed, extending beyond what has been outlined in chapters 8 and 14.

Entity-Centric Learning on Data Streams:

RQ1: To what extent can entity-centric models improve performance compared to an entity-ignorant model?
RQ2: How can the memory footprint of entity-centric models be reduced?

RQ1 has been addressed in chapters 4, 5, and 6, where it has been demonstrated how simple entity-centric models, such as a majority classifier, can enhance predictions for specific entities. Additionally, it was shown that combining entity-centric models with an entity-ignorant model in an ensemble improves performance across the entire data stream.

Chapter 5 emphasizes the growing memory demands of entity-centric models, as each encountered entity necessitates the creation of a new model, potentially leading to system crashes.

This challenge is tackled in chapter 6, which addresses RQ2 and introduces two approaches to managing these memory demands effectively. The first method reallocates models of inactive entities to secondary memory, thus freeing up primary memory. The second approach reduces memory usage by creating lightweight, label-only models for each entity, minimizing the burden on primary memory.

Active Feature Acquisition on Data Streams:

RQ3: How can Active Feature Acquisition be realized in a data stream setting?
RQ4: How can varying feature costs be considered during AFA on streams?

RQ3 is addressed in chapters 10, 11, and 12, where the merit of a feature is defined by its ability to separate classes and its associated cost. It is demonstrated that an acquisition process based on feature merits often outperforms or delivers comparable performance to a random acquisition baseline. Chapter 12 further addresses the limitation of using a feature importance metric that evaluates features independently, by introducing a feature-pair imputer (FPI) that considers feature correlations when a selected feature can be imputed effectively from available features.

RQ4 is discussed in chapters 11 and 12. Chapter 11 introduces AFA methods capable of handling varying feature costs, while chapter 12 presents a budget-saving mechanism while maintaining a certain level of prediction performance.

15.1. Future Work

Chapters 8 and 14 outline open challenges as well as future work for each part of the thesis, including the use of more datasets, metrics, and new classifiers. The most straightforward extensions of the presented work are discussed in the chapters 7 and 13 containing additional reflections.

This section provides more ambitious future research directions.

Entity-Centric Learning Deep learning methods, especially Large Language Models (LLMs), have gained widespread popularity due to their superior performance across many domains. However, they are still not fully optimized for use in data streams. Current concept drift adaptation techniques are often costly and batch-based [83], presenting opportunities for improvement. Given their success in opinion mining, it would be particularly interesting to explore the use of LLMs as base models for both the entity-ignorant and entity-centric components. Although LLMs typically require vast amounts of training data, once initialized, they can be applied to various tasks [17], potentially reducing the impact of concept drift on textual data. For instance, while opinions on a product may shift over time—leading to changes in the vocabulary encountered in reviews — the underlying meaning of words is unlikely to change in the short term. This suggests that an LLM scoring reviews based on text could maintain robust performance over time.

To initialize an LLM for entity-specific tasks, few-shot learning could be employed, using the short substreams of training instances for each entity as examples. This approach would allow the use of a single model for the entire stream, rather than one model per entity, with entity-specific adjustments made via prompts.

Another area of interest is clustering entities and building dedicated models for each cluster. Cluster-based models could either serve as a third component in the ensemble alongside entity-ignorant and entity-centric models or replace them, depending on their performance. This would require the adaptation of substream clustering techniques [68] to suit the specific requirements of this approach.

Active Feature Acquisition The most pressing future work involves completing the tree-based acquisition set approach, which aims to reduce the acquisition of superfluous features and better handle scenarios where a combination of multiple features determines the label of an instance. This approach will be compared against existing methods, as well as a new performance bound that provides a locally optimal acquisition strategy given the same budget. This performance bound, realized through genetic programming, will be submitted for publication along with the tree-based approach in the near future, see chapter 13.

More ambitious plans include scenarios where feature costs fluctuate over time. For instance, laboratory costs for feature acquisition may vary depending on the availability of certain services, necessitating strategic selection based on real-time cost. This closely relates to improvements in the budget manager, which can still under- or overspend in extreme cases depending on the chosen AFA strategy, see section 12.4.2. One potential improvement could involve a hybrid approach using both the IPF and the Simple Budget Manager (SBM), where the IPF is applied in most situations, but switches to the SBM when an over- or underspending threshold is reached. Another avenue for future work is combining the merit and imputation error of a missing feature into a single score presented to the budget manager, instead of the current two-stage approach (chapter 12) where merit and imputation error are considered separately. Additionally, the interaction between entity-centric learning and AFA can be investigated, particularly in scenarios where specific features are highly predictive for certain entities.

This thesis demonstrates how entity-centric learning and AFA can be effectively implemented on data streams, often leading to performance gains. As highlighted here and in chapters 8 and 14, there is still a wealth of exciting future work to be explored in both areas.

Part III.

Appendix

A. Appendix

A.1. Appendix: Entity-Centric Learning

This part of the Appendix covers additional material concerning Part I of the thesis.

A.1.1. Error Analysis of Simple Entity-Centric Models in Chapter 4

Figure A.1 shows the distribution of errors for all predictors on both datasets for entities that have different numbers of reviews. The large skew in the dataset towards 5-star ratings is particularly apparent on inspection of GlobalPrior. Even though the algorithm always predicts 5 stars, most ratings are correct. It can also be seen that only MNB classifiers predict the exact label more often than GlobalPrior. In addition, SMA also makes the least number of large prediction errors (four-off). The high percentage of one-off errors is caused by the sensitivity towards outliers which is shared by the regression predictor. The sensitivity of regression methods to outlier ratings is further stressed by the fact that it yields the least number of exact label predictions over all algorithms.

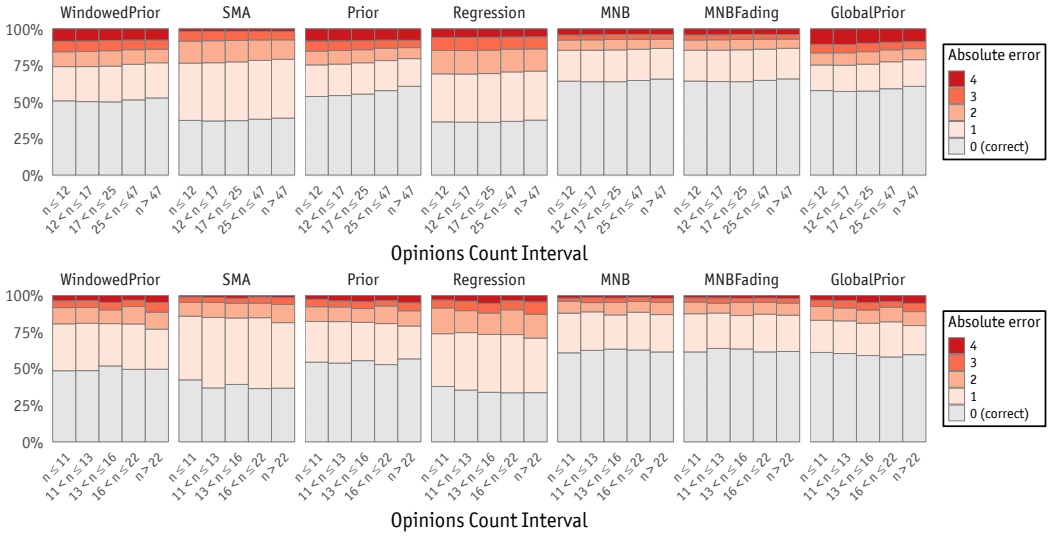

Figure A.1.: Distribution of absolute errors for selected predictors with $w = 5$ for Tools (top) and Watches (bottom). Predictions are stratified by a product's review count into (nearly) equally sized groups.

A.1.2. Additional Results for Entity-Centric Ensembles on Watches Dataset Chapter 5

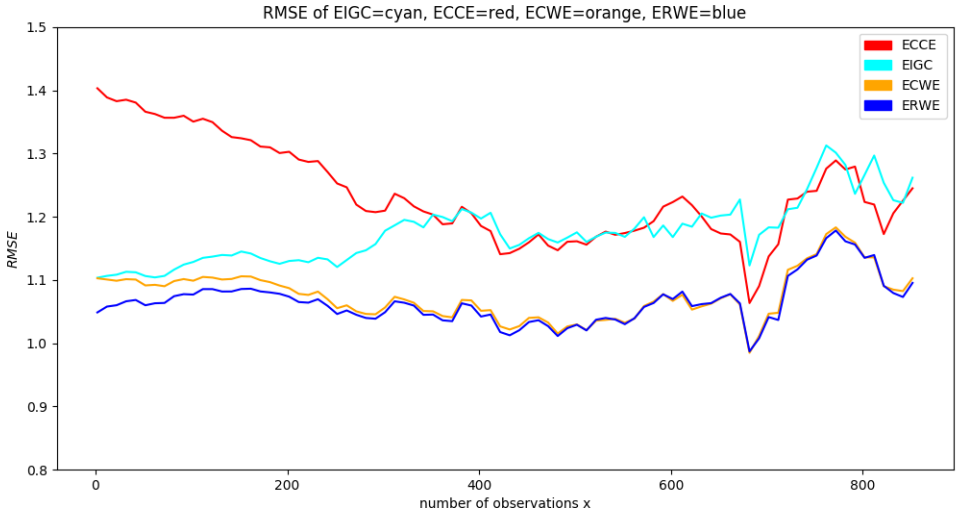

Figure A.2.: Dataset **watches**: Entity-centric $RMSE$; lower values indicate better performance. $ECCE$ starts to show better performance than $EIGC$ when around $x = 400$ instances are available for training per entity. Figure reproduced with permission from [8] ©2019 ACM.

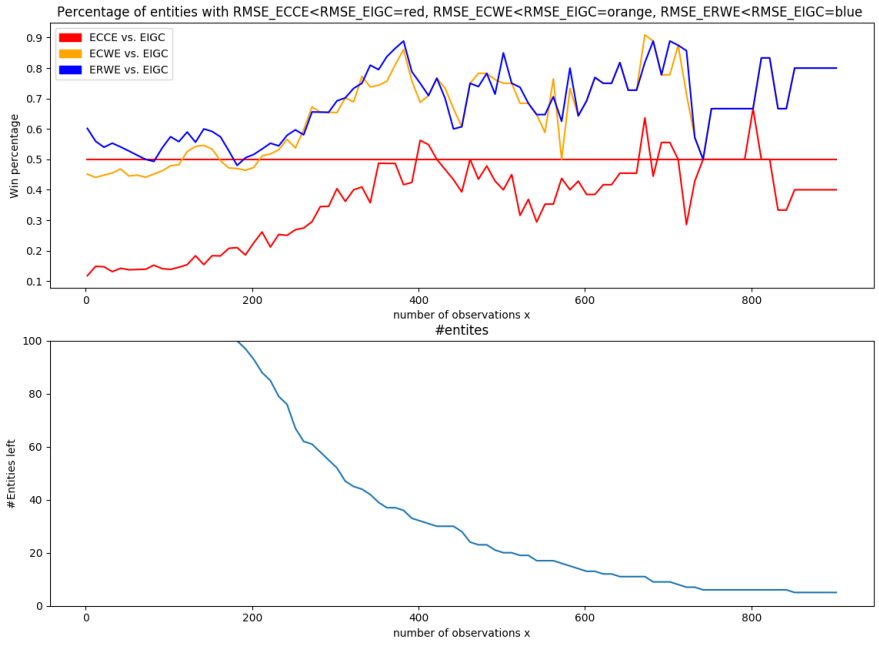

Figure A.3.: Dataset **`watches`** - **Top:** Proportion of entities at threshold x where the $RMSE$ of entity-centric models outperforms that of $EIGC$: $ECCE$ (red), $ECWE$ (orange), and $ERWE$ (blue). Higher values correspond to better results.
Bottom: The count of entities remaining at threshold x. Figure reproduced with permission from [8] ©2019 ACM.

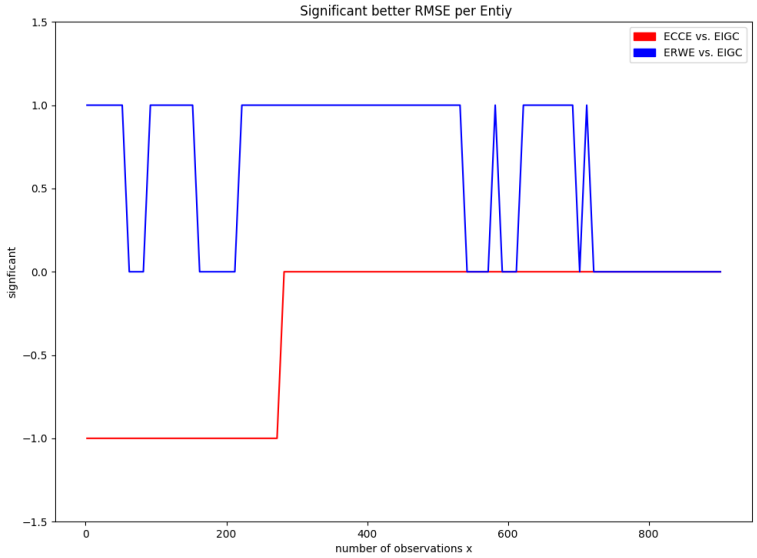

Figure A.4.: Significance testing using the Wilcoxon signed-rank test ($p = 0.025$) comparing the $RMSE$ of $EIGC$ with $ECCE$ and $ERWE$. A value of 1 means the entity-centric ensemble outperforms $EIGC$, while -1 means $EIGC$ performs better. $ERWE$ demonstrates superior results for most x values and is never significantly worse. Figure reproduced with permission from [8] ©2019 ACM.

A.2. Appendix: Active Feature Acquisition

This part of the Appendix covers additional material concerning Part II of the thesis.

A.2.1. Additional Feature Importance Metrics of Chapter 10

Yuan et al. define Information Gain and Symmetric Uncertainty as follows:

"The formula used to calculate Information Gain (IG) for an attribute X_i and a class attribute Y is defined as follows:

$$IG(Y|X_i) = H(Y) - H(Y|X_i) \tag{4}$$

where $H(X_i)$ is the entropy for an attribute X_i with N distinct values, given by the following formula:

$$H(X_i) = \sum_{j=0}^{N} -P(X_i = x_j) \log P(X_i = x_j) \tag{5}$$

and $H(Y|X_i)$ is the conditional entropy, given by:

$$H(Y|X_i) = \sum_{j=0}^{N} p(x_j) H(Y|X_i = x_j) \tag{6}$$

Symmetric Uncertainty (SU) can be seen as a normalized version of Information Gain and ... the formula for Symmetric Uncertainty for an attribute X with the class attribute Y is defined as follows:

$$SU(X, Y) = \frac{IG(X|Y)}{H(X) + H(Y)} \tag{7}$$

" [85]

A.2.2. Extensive Result Tables from Chapter 10

Table A.1.: Mean kappa values over 1 run on the `electricity` dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 1 run on data set electricity											
missingness	0.25(kappa ∈ [0.395, 0.477])				0.5(kappa ∈ [0.308, 0.478])				0.75(kappa ∈ [0.181, 0.486])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.404	0.418	0.431	0.437	0.325	0.335	0.347	0.37	0.212	0.23	0.253	0.263
MWAED+SBM	0.406	0.415	0.416	0.432	0.331	0.336	0.355	0.361	0.204	0.22	0.249	0.265
SWAED+IPF	0.41	0.422	0.435	0.459	0.33	0.362	0.375	0.424	0.248	0.281	0.331	0.386
SWAED+SBM	0.414	0.424	0.441	0.456	0.331	0.359	0.379	0.427	0.231	0.281	0.327	0.388
SWIG+IPF	0.419	0.431	0.452	0.463	**0.34**	0.351	0.385	**0.445**	0.244	0.272	0.334	0.403
SWIG+SBM	0.414	0.426	0.448	**0.472**	0.331	0.352	0.391	0.438	0.233	0.265	0.33	0.394
SWSU+IPF	**0.427**	**0.436**	0.439	0.471	0.336	0.361	0.398	0.44	**0.249**	**0.3**	**0.349**	0.417
SWSU+SBM	0.41	0.424	**0.454**	0.467	0.332	**0.366**	**0.405**	0.442	0.237	0.29	0.347	**0.425**
RA+SBM	0.404	0.416	0.43	0.443	0.317	0.322	0.333	0.352	0.205	0.22	0.231	0.255

Table A.2.: Mean kappa values over 10 runs on the **nursery** data set using an SGD classifier. Table reproduced from [9].

	mean kappa over 10 runs on data set nursery											
missingness	0.25(kappa ∈ [0.516, 0.84])				0.5(kappa ∈ [0.3, 0.843])				0.75(kappa ∈ [0.129, 0.842])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.577	0.632	0.69	0.75	0.385	0.458	0.549	0.65	0.225	0.317	0.43	0.566
MWAED+SBM	0.572	0.634	0.691	0.751	0.377	0.462	0.557	0.649	0.219	0.324	0.437	0.567
SWAED+IPF	0.586	0.645	0.704	0.761	0.385	0.467	0.558	0.654	0.222	0.313	0.424	0.568
SWAED+SBM	0.577	0.638	0.697	0.762	0.378	0.465	0.561	0.653	0.215	0.32	0.441	0.568
SWIG+IPF	**0.599**	0.661	**0.735**	**0.779**	**0.401**	**0.494**	0.594	0.683	0.232	0.335	0.458	0.591
SWIG+SBM	0.583	0.646	0.71	0.778	0.381	0.477	0.58	0.68	0.225	0.333	0.454	**0.592**
SWSU+IPF	0.598	**0.666**	0.734	0.777	**0.401**	**0.494**	**0.598**	**0.684**	**0.233**	**0.336**	**0.46**	0.589
SWSU+SBM	0.578	0.646	0.711	**0.779**	0.386	0.475	0.58	0.682	0.222	0.331	0.454	0.589
RA+SBM	0.549	0.584	0.614	0.646	0.321	0.346	0.371	0.398	0.147	0.169	0.189	0.209

Table A.3.: Mean kappa values over 1 run on the **sea** dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 1 run on data set sea											
missingness	0.25(kappa ∈ [0.428, 0.593])				0.5(kappa ∈ [0.275, 0.588])				0.75(kappa ∈ [0.127, 0.583])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.469	0.493	**0.529**	0.562	0.339	0.402	0.442	0.49	0.189	0.254	0.34	0.389
MWAED+SBM	0.458	0.49	0.528	0.565	0.331	0.384	0.434	0.495	0.193	0.261	0.327	0.394
SWAED+IPF	0.468	**0.5**	0.526	0.564	**0.343**	0.399	**0.447**	0.497	**0.199**	0.255	**0.345**	**0.397**
SWAED+SBM	0.456	0.494	0.526	0.561	0.327	0.387	0.444	0.499	**0.199**	**0.27**	0.319	0.39
SWIG+IPF	0.467	0.496	0.523	0.561	0.336	**0.412**	0.441	0.499	0.19	0.263	0.34	0.393
SWIG+SBM	0.448	0.493	0.528	0.565	0.33	0.385	0.446	**0.502**	0.197	0.264	0.33	0.395
SWSU+IPF	**0.476**	0.499	0.518	**0.567**	0.341	0.408	0.444	0.496	0.195	0.255	0.335	0.394
SWSU+SBM	0.46	0.493	0.527	0.559	0.334	0.384	0.446	**0.502**	0.197	0.264	0.332	0.396
RA+SBM	0.453	0.483	0.518	0.544	0.314	0.359	0.399	0.442	0.174	0.229	0.264	0.319

Table A.4.: Mean kappa values over 1 run on the **gen** dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 1 run on data set gen											
missingness	0.25(kappa ∈ [0.376, 0.503])				0.5(kappa ∈ [0.225, 0.508])				0.75(kappa ∈ [0.105, 0.463])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.403	0.415	0.438	0.471	0.27	0.339	0.379	0.437	0.169	0.223	0.303	0.355
MWAED+SBM	0.391	0.418	0.449	0.482	0.248	0.29	0.352	0.442	0.12	0.197	0.288	0.37
SWAED+IPF	0.406	0.424	**0.459**	0.436	0.289	0.35	0.395	0.465	0.179	0.237	**0.364**	0.37
SWAED+SBM	0.41	0.405	0.434	0.444	0.266	0.32	0.404	0.431	0.152	0.217	0.32	0.39
SWIG+IPF	0.404	0.437	0.437	0.459	0.312	0.353	**0.415**	0.438	**0.226**	0.275	0.319	**0.415**
SWIG+SBM	0.41	0.421	0.442	0.478	0.259	0.298	**0.415**	0.45	0.156	0.209	0.32	0.401
SWSU+IPF	**0.437**	**0.445**	0.433	**0.509**	**0.321**	**0.354**	0.413	**0.478**	0.185	**0.291**	0.305	0.392
SWSU+SBM	0.395	0.4	0.452	0.467	0.307	0.324	0.39	0.43	0.174	0.23	0.296	0.414
RA+SBM	0.365	0.436	0.444	0.476	0.265	0.3	0.298	0.371	0.129	0.204	0.19	0.253

Table A.5.: Mean kappa values over 10 runs on the **adult** dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 10 runs on data set adult											
missingness	0.25(kappa ∈ [0.371, 0.445])				0.5(kappa ∈ [0.283, 0.443])				0.75(kappa ∈ [0.156, 0.445])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.379	0.38	0.385	0.401	0.285	0.294	0.304	0.32	0.16	0.172	0.186	0.204
MWAED+SBM	0.378	0.382	0.389	0.397	0.289	0.299	0.307	0.319	0.163	0.173	0.185	0.202
SWAED+IPF	0.376	0.381	0.388	0.399	0.286	0.288	0.299	0.312	0.164	0.172	0.185	0.199
SWAED+SBM	0.376	0.384	0.39	0.398	0.289	0.296	0.302	0.312	0.163	0.174	0.183	0.198
SWIG+IPF	**0.385**	0.388	0.396	0.407	0.29	0.302	0.318	0.337	**0.175**	0.193	**0.216**	**0.242**
SWIG+SBM	0.38	**0.389**	**0.397**	**0.408**	**0.293**	**0.307**	**0.321**	**0.339**	0.17	**0.194**	0.213	**0.242**
SWSU+IPF	0.379	0.385	0.392	0.407	0.29	0.299	0.309	0.335	0.171	0.188	0.208	0.232
SWSU+SBM	0.379	0.387	0.395	0.403	0.29	0.304	0.316	0.334	0.17	0.187	0.207	0.234
RA+SBM	0.375	0.386	0.392	0.396	0.291	0.295	0.304	0.314	0.165	0.177	0.187	0.204

Table A.6.: Mean kappa values over 10 runs on the **occupancy** dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 10 runs on data set occupancy											
missingness	0.25(kappa ∈ [0.771, 0.949])				0.5(kappa ∈ [0.595, 0.95])				0.75(kappa ∈ [0.364, 0.948])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.803	0.837	0.874	0.93	0.645	0.701	0.781	0.887	0.484	0.58	0.695	0.849
MWAED+SBM	0.802	0.842	0.882	0.932	0.656	0.728	0.801	0.888	0.481	0.593	0.711	0.848
SWAED+IPF	0.809	0.848	0.894	0.945	0.666	0.733	0.822	0.934	0.504	0.624	0.75	0.918
SWAED+SBM	0.804	0.846	0.896	0.944	0.666	0.738	0.827	0.933	0.486	0.613	0.751	0.916
SWIG+IPF	0.806	0.846	0.883	**0.946**	0.667	0.735	0.814	0.937	0.51	0.613	0.752	0.931
SWIG+SBM	0.805	0.846	0.893	0.944	0.666	0.74	0.827	0.936	0.492	0.623	0.757	0.932
SWSU+IPF	**0.818**	**0.854**	**0.898**	0.945	**0.685**	**0.759**	**0.841**	**0.94**	**0.525**	**0.651**	**0.777**	0.93
SWSU+SBM	0.804	0.846	0.895	0.945	0.669	0.743	0.831	**0.94**	0.49	0.621	0.756	**0.933**
RA+SBM	0.787	0.805	0.821	0.843	0.62	0.642	0.662	0.685	0.397	0.432	0.458	0.497

Table A.7.: Mean kappa values over 10 runs on **pendigits** dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 10 runs on data set pendigits											
missingness	0.25(kappa ∈ [0.654, 0.881])				0.5(kappa ∈ [0.46, 0.879])				0.75(kappa ∈ [0.22, 0.879])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.664	0.677	0.693	**0.727**	**0.481**	0.498	0.522	**0.566**	**0.247**	**0.273**	0.306	0.354
MWAED+SBM	**0.666**	**0.683**	**0.698**	0.725	**0.481**	**0.503**	**0.53**	0.564	0.241	0.271	**0.307**	**0.355**
SWAED+IPF	0.664	0.676	0.692	**0.727**	0.48	0.497	0.522	0.563	0.246	0.267	0.3	0.345
SWAED+SBM	**0.666**	0.68	**0.698**	**0.727**	0.478	0.502	0.528	0.565	0.24	0.267	0.298	0.345
SWIG+IPF	0.665	0.675	0.688	0.713	0.474	0.484	0.508	0.546	0.238	0.253	0.275	0.309
SWIG+SBM	0.663	0.674	0.692	0.715	0.472	0.49	0.514	0.546	0.234	0.251	0.278	0.305
SWSU+IPF	0.664	0.674	0.692	0.723	0.474	0.49	0.511	0.559	0.237	0.259	0.285	0.328
SWSU+SBM	0.665	0.679	0.695	0.721	0.476	0.495	0.521	0.561	0.235	0.257	0.288	0.328
RA+SBM	0.663	0.672	0.683	0.694	0.471	0.488	0.499	0.515	0.231	0.247	0.264	0.284

Table A.8.: Mean kappa values over 10 runs on the **abalone** dataset using an SGD classifier. Table reproduced from [9].

	mean kappa over 10 runs on data set abalone											
missingness	0.25(kappa ∈ [0.226, 0.238])				0.5(kappa ∈ [0.206, 0.249])				0.75(kappa ∈ [0.163, 0.252])			
budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.226	0.22	0.221	0.233	0.211	**0.216**	0.211	0.218	**0.186**	0.19	0.193	0.193
MWAED+SBM	0.224	**0.232**	0.229	0.223	0.204	0.208	0.215	0.22	0.166	0.179	0.186	0.198
SWAED+IPF	0.23	0.229	0.225	0.234	0.21	0.211	0.213	0.214	0.185	0.188	0.193	0.197
SWAED+SBM	0.223	0.229	0.227	0.229	0.205	0.207	0.215	0.214	0.166	0.181	0.191	0.197
SWIG+IPF	**0.233**	0.231	0.229	**0.241**	0.211	0.212	0.211	0.217	**0.186**	0.186	0.189	0.2
SWIG+SBM	0.222	0.228	0.232	0.234	0.212	0.211	0.215	**0.222**	0.178	0.179	0.199	0.202
SWSU+IPF	0.228	0.219	**0.236**	0.237	0.21	0.212	**0.217**	0.219	0.176	**0.194**	0.192	**0.21**
SWSU+SBM	0.229	0.222	0.233	0.236	0.206	0.21	0.211	0.214	0.165	0.184	**0.201**	0.205
RA+SBM	0.227	0.228	0.232	0.222	**0.213**	0.208	0.216	0.211	0.17	0.169	0.185	0.2

A.2.3. New Experiment on Adult Dataset Chapter 10

The low performance on the `adult` dataset was thought to be a result of the restriction of just being able to buy one single feature per entity and the restricted available budget. While the experiments presented in Table A.9 were conducted with a different version of the code and are not directly comparable, it shows that for equal feature costs the 2-Best strategy with a budget of 2 achieves a higher Kappa value of 0.26 compared to the 1-Best method with budget 1 which only achieves a Kappa of 0.18 while being the closest to the original method presented in chapter 10. This supports our assessment that allowing for more features to be bought and a higher budget would improve the performance.

missingness	75% (kappa $\in [0.15, 0.45]$)								
distribution	*equal*			*increasing*			*decreasing*		
budget	0.5	1.0	2.0	0.5	1.0	2.0	0.5	1.0	2.0
1-Best IG	**0.2**	**0.24**	0.24	0.16	0.18	0.22	0.15	0.16	0.17
1-Best SU	0.19	0.22	0.22	0.16	0.18	0.22	0.15	0.16	**0.18**
1-Best	0.18	0.18	0.2	0.16	**0.2**	0.22	0.15	0.17	0.18
2-Best	0.19	0.22	**0.26**	0.16	0.16	0.18	0.15	0.16	0.17
3-Best	0.16	0.19	0.24	0.16	0.17	0.17	0.15	0.17	0.18
4-Best	0.16	0.19	0.23	0.14	0.16	0.17	0.15	0.16	0.17
100-Best	0.16	0.18	0.22	0.15	0.15	0.15	0.15	0.15	0.16
1-Global Best	0.17	0.17	0.17	**0.17**	0.18	**0.23**	**0.16**	**0.18**	0.18
2-Global Best	0.19	0.21	0.25	0.16	0.16	0.19	0.16	0.16	0.17
4-Global Best	0.17	0.19	0.23	0.16	0.16	0.17	0.16	0.15	0.17
4-Quality Gain	0.18	0.19	0.25	0.16	0.16	0.17	0.16	0.16	0.17
100-SSBQG	0.18	0.2	0.25	0.15	0.16	0.17	0.16	0.16	0.17
1-Best FPI	0.16	0.18	0.19	0.15	0.17	0.21	0.15	0.15	0.16
1-Best IMPTS 2%	0.16	-	-	0.15	-	-	0.14	-	-
1-Best IMPTS 6%	0.16	-	-	0.16	-	-	0.15	-	-
1-Best IMPTS 10%	0.16	-	-	0.15	-	-	0.14	-	-

Table A.9.: Mean kappa values over 10 runs on *adult* data set with *missingness* = 75%.

Bibliography

[1] Adhikari, Deepak et al. "A Comprehensive Survey on Imputation of Missing Data in Internet of Things". In: *ACM Comput. Surv.* 55.7 (2022). ISSN: 0360-0300. DOI: 10.1145/3533381. URL: https://doi.org/10.1145/3533381.

[2] Agrahari, Supriya and Singh, Anil Kumar. "Concept drift detection in data stream mining: A literature review". In: *Journal of King Saud University-Computer and Information Sciences* 34.10 (2022), pp. 9523–9540.

[3] Anagnostopoulos, Christoforos et al. "Online Linear and Quadratic Discriminant Analysis with Adaptive Forgetting for Streaming Classification". In: *Statistical Analysis and Data Mining* (2012). DOI: 10.1002/sam.10151.

[4] Asudani, Deepak Suresh, Nagwani, Naresh Kumar, and Singh, Pradeep. "Impact of word embedding models on text analytics in deep learning environment: a review". In: *Artificial intelligence review* 56.9 (2023), pp. 10345–10425.

[5] Azur, Melissa J et al. "Multiple imputation by chained equations: what is it and how does it work?" In: *International journal of methods in psychiatric research* 20.1 (2011), pp. 40–49.

[6] Bahri, Maroua et al. "Data stream analysis: Foundations, major tasks and tools". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.3 (2021), e1405.

[7] Beyer, Christian et al. "Predicting Polarities of Entity-Centered Documents without Reading their Contents". In: *Proceedings of the Symposium on Applied Computing*. ACM. 2018. DOI: https://doi.org/10.1145/3167132.317287.

[8] Beyer, Christian et al. "Exploiting Entity Information for Stream Classification over a Stream of Reviews". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. ACM, 2019, pp. 564–573. DOI: https://doi.org/10.1145/3297280.3297333. URL: https://doi.org/10.1145/3297280.3297333.

[9] Beyer, Christian et al. "Active feature acquisition on data streams under feature drift". In: *Annals of Telecommunications* 75 (2020), pp. 597–611. DOI: https://doi.org/10.1007/s12243-020-00775-2.

[10] Beyer, Christian et al. "Resource management for model learning at entity level". In: *Annals of Telecommunications* 75.9-10 (2020), pp. 549–561. DOI: 10.1007/s12243-020-00800-4. URL: https://doi.org/10.1007%2Fs12243-020-00800-4.

[11] Bifet, Albert and Gavalda, Ricard. "Adaptive learning from evolving data streams". In: *Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8*. Springer. 2009, pp. 249–260.

[12] Bifet, Albert, Holmes, Geoffrey, and Pfahringer, Bernhard. "MOA-TweetReader: real-time analysis in Twitter streaming data". In: *Proc. of the 14th Int'l. Conf. on Discovery science*. DS'11. Espoo, Finland: Springer-Verlag, 2011, pp. 46–60. ISBN: 978-3-642-24476-6.

[13] Bifet, Albert et al. "Pitfalls in benchmarking data stream classification and how to avoid them". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 465–479.

[14] Breiman, L et al. "Classification and Regression Trees". In: (1984).

[15] Büttner, Maik, Beyer, Christian, and Spiliopoulou, Myra. "Reducing Missingness in a Stream through Cost-Aware Active Feature Acquisition". In: *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. 2022 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA). 2022. DOI: 10.1109/DSAA54385.2022.10032414.

[16] Büttner, Maik, Beyer, Christian, and Spiliopoulou, Myra. "Joining Imputation and Active Feature Acquisition for Cost Saving on Data Streams with Missing Features". In: *International Conference on Discovery Science*. Springer. 2023, pp. 308–322. DOI: doi.org/10.1007/978-3-031-45275-8_21.

[17] Chen, Qijie et al. "An extensive benchmark study on biomedical text generation and mining with ChatGPT". In: *Bioinformatics* 39.9 (2023), btad557.

[18] Demšar, Janez. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[19] Deng, Lingjia and Wiebe, Janyce. "Joint Prediction for Entity/Event-Level Sentiment Analysis using Probabilistic Soft Logic Models". In: *2015 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal: Association for Computational Linguistics, 2015.

[20] desJardins, Marie, MacGlashan, James, and Wagstaff, Kiri L. "Confidence-Based Feature Acquisition to Minimize Training and Test Costs". In: *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 514–524.

[21] Ding, Chaoyue, Zhao, Jing, and Sun, Shiliang. "Concept drift adaptation for time series anomaly detection via transformer". In: *Neural Processing Letters* 55.3 (2023), pp. 2081–2101.

[22] Domingos, Pedro and Hulten, Geoff. "Mining high-speed data streams". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2000, pp. 71–80.

[23] Dong, Wenlu et al. "An exploration of online missing value imputation in non-stationary data stream". In: *SN Computer Science* 2 (2021), pp. 1–11.

[24] Erdogan, Birsen Eygi, Özögür-Akyüz, Süreyya, and Atas, Pınar Karadayi. "A novel approach for panel data: An ensemble of weighted functional margin SVM models". In: *Information Sciences* (2019). ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.02.045. URL: http://www.sciencedirect.com/science/article/pii/S0020025519301549.

[25] Fafalios, Pavlos et al. "Multi-aspect Entity-centric Analysis of Big Social Media Archives". In: *International Conference on Theory and Practice of Digital Libraries*. Springer. 2017, pp. 261–273.

[26] Fudholi, Dhomas Hatta et al. "BERT-based tourism Named Entity Recognition: making use of social media for travel recommendations". In: *PeerJ Computer Science* 9 (2023), e1731.

[27] Gama, Joao and Pinto, Carlos. "Discretization from data streams: applications to histograms and data mining". In: *Proceedings of the 2006 ACM symposium on Applied computing*. 2006, pp. 662–667.

[28] Gama, Joao, Sebastiao, Raquel, and Rodrigues, Pedro Pereira. "Issues in evaluation of stream learning algorithms". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 329–338.

[29] Gama, Joao, Sebastiao, Raquel, and Rodrigues, Pedro Pereira. "On evaluating stream learning algorithms". In: *Machine learning* 90 (2013), pp. 317–346.

[30] Gama, Joao et al. "A Survey on Concept Drift Adaptation". In: *ACM Comput. Surv.* 46.4 (2014). ISSN: 0360-0300. DOI: 10.1145/2523813.

[31] Gomes, Heitor M et al. "Adaptive random forests for evolving data stream classification". In: *Machine Learning* 106 (2017), pp. 1469–1495.

[32] Gomes, Heitor Murilo et al. "Feature scoring using tree-based ensembles for evolving data streams". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 761–769.

[33] Hallaji, Ehsan, Razavi-Far, Roozbeh, and Saif, Mehrdad. "DLIN: Deep ladder imputation network". In: *IEEE Transactions on Cybernetics* 52.9 (2021), pp. 8629–8641.

[34] He, Ruining and McAuley, Julian. "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering". In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pp. 507–517.

[35] Herbold, Steffen. "Autorank: A Python package for automated ranking of classifiers". In: *Journal of Open Source Software* 5.48 (2020), p. 2173.

[36] Hong, Jong-yi, Suh, Eui-ho, and Kim, Sung-Jin. "Context-aware systems: A literature review and classification". In: *Expert Systems with applications* 36.4 (2009), pp. 8509–8522.

[37] Huang, Sheng-Jun et al. *Active Feature Acquisition with Supervised Matrix Completion*. 2018. DOI: 10.48550/ARXIV.1802.05380. URL: https://arxiv.org/abs/1802.05380.

[38] Jakob, Niklas and Gurevych, Iryna. "Extracting Opinion Targets in a Single- and Cross-Domain Setting with Conditional Random Fields". In: *2010 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*. MIT, Massachusetts: Association for Computational Linguistics, 2010, pp. 1035–1045.

[39] Keerin, Phimmarin and Boongoen, Tossapon. "Improved knn imputation for missing values in gene expression data". In: *Computers, Materials and Continua* 70.2 (2021), pp. 4009–4025.

[40] Kelly, Markelle, Longjohn, Rachel, and Nottingham, Kolby. "The UCI machine learning repository". In: *URL https://archive. ics. uci. edu* (2023).

[41] Kottke, D. "Budget Optimization for Active Learning in Data Streams". PhD thesis. Master's thesis, Otto von Guericke University Magdeburg, Germany (10 2014).

[42] Kottke, Daniel, Krempl, Georg, and Spiliopoulou, Myra. "Probabilistic active learning in datastreams". In: *International Symposium on Intelligent Data Analysis*. Springer. 2015, pp. 145–157.

[43] Krawczyk, Bartosz, Woźniak, Michał, and Cyganek, Bogusław. "Clustering-based ensembles for one-class classification". In: *Information Sciences* 264 (2014). Serious Games, pp. 182–195. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2013.12.019. URL: https://www.sciencedirect.com/science/article/pii/S0020025513008694.

[44] Krempl, Georg, Siddiqui, Zaigham Faraz, and Spiliopoulou, Myra. "Online clustering of high-dimensional trajectories under concept drift". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II 22*. Springer. 2011, pp. 261–276.

[45] Kumar, Ashish and Singh, Ajmer. "Stream mining a review: tool and techniques". In: *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*. Vol. 2. IEEE. 2017, pp. 27–32.

[46] Kumar, Vipin and Minz, Sonajharia. "Feature selection". In: *SmartCR* 4.3 (2014), pp. 211–229.

[47] Li, Jing et al. "A survey on deep learning for named entity recognition". In: *IEEE transactions on knowledge and data engineering* 34.1 (2020), pp. 50–70.

[48] Li, Yang and Oliva, Junier. "Active feature acquisition with generative surrogate models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6450–6459.

[49] Li, Yang et al. "Towards Robust Active Feature Acquisition". In: *CoRR* abs/2107.04163 (2021). arXiv: 2107.04163. URL: https://arxiv.org/abs/2107.04163.

[50] Liao, J. and Dai, B. "An Ensemble Learning Approach for Concept Drift". In: *2014 International Conference on Information Science Applications (ICISA)*. 2014, pp. 1–4.

[51] Libera, Caio et al. "'right to be forgotten': analyzing the impact of forgetting data using k-NN algorithm in data stream learning". In: *International Conference on Electronic Government*. Springer. 2022, pp. 530–542.

[52] Lin, Wei-Chao and Tsai, Chih-Fong. "Missing value imputation: a review and analysis of the literature (2006–2017)". In: *Artificial Intelligence Review* 53 (2020), pp. 1487–1509.

[53] Lipka, Nedim, Stein, Benno, and Anderka, Maik. "Cluster-based one-class ensemble for classification problems in information retrieval". In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 2012, pp. 1041–1042.

[54] Liu, Zitao and Hauskrecht, Milos. "Learning adaptive forecasting models from irregularly sampled multivariate clinical data". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[55] Lu, Haibing and Huang, Shengsheng. "Clustering panel data". In: *SIAM International Workshop on Data Mining held in conjunction with the 2011 SIAM International Conference on Data Mining*. 2011, pp. 1–10.

[56] Manku, Gurmeet Singh and Motwani, Rajeev. "Approximate frequency counts over data streams". In: *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier. 2002, pp. 346–357.

[57] Matuszyk, Pawel and Spiliopoulou, Myra. "Selective Forgetting for Incremental Matrix Factorization in Recommender Systems". In: *Int. Conf. on Discovery Science (DS'14)*. Vol. 8777. LNCS. Springer International Publishing, 2014, pp. 204–215. DOI: 10.1007/978-3-319-11812-3_18.

[58] Melidis, Damianos P., Spiliopoulou, Myra, and Ntoutsi, Eirini. "Learning under Feature Drifs in Textual Streams". In: *Proceedings of the 2018 ACM on Conference on Information and Knowledge Management*. to appear. ACM. 2018.

[59] Melville, Prem et al. "Active feature-value acquisition for classifier induction". In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE. 2004, pp. 483–486.

[60] Min, Bonan et al. "Recent advances in natural language processing via large pre-trained language models: A survey". In: *ACM Computing Surveys* 56.2 (2023), pp. 1–40.

[61] Oku, Kenta et al. "Context-aware SVM for context-dependent information recommendation". In: *Proceedings of the 7th international Conference on Mobile Data Management*. IEEE Computer Society. 2006, p. 109.

[62] Pang, Bo and Lee, Lillian. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales". In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2005, pp. 115–124.

[63] Pappas, Nikolaos and Popescu-Belis, Andrei. "Explicit Document Modeling through Weighted Multiple-Instance Learning". In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 591–626.

[64] Parada-Cabaleiro, Emilia et al. "Song lyrics have become simpler and more repetitive over the last five decades". In: *Scientific Reports* 14.1 (2024), p. 5531.

[65] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[66] Peng, Tao, Sellami, Sana, and Boucelma, Omar. "Iot data imputation with incremental multiple linear regression". In: *Open Journal of Internet Of Things (OJIOT)* 5.1 (2019), pp. 69–79.

[67] Qiu, Guang et al. "Opinion word expansion and target extraction through double propagation". In: *Computational Linguistics* 37.1 (2011), pp. 9–27.

[68] Bi-Ru Dai et al. "Adaptive Clustering for Multiple Evolving Streams". In: *IEEE Transactions on Knowledge and Data Engineering* 18.9 (2006), pp. 1166–1180.

[69] Rudovic, Ognjen et al. "Meta-weighted gaussian process experts for personalized forecasting of AD cognitive changes". In: *Machine Learning for Healthcare Conference*. 2019, pp. 181–196.

[70] Saadallah, Amal, Priebe, Florian, and Morik, Katharina. "A Drift-Based Dynamic Ensemble Members Selection Using Clustering for Time Series Forecasting". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld et al. Cham: Springer International Publishing, 2020, pp. 678–694. ISBN: 978-3-030-46150-8.

[71] Saar-Tsechansky, Maytal, Melville, Prem, and Provost, Foster J. "Active Feature-Value Acquisition". In: *Management Science* 55.4 (2009), pp. 664–684.

[72] Salgado, Cátia M et al. "Missing Data". In: *Secondary Analysis of Electronic Health Records [Internet]* (2016).

[73] Serrao, Elson and Spiliopoulou, Myra. "Active Stream Learning with an Oracle of Unknown Availability for Sentiment Prediction". In: *Proceedings of the Workshop on Interactive Adaptive Learning co-located with European Conference on Machine Learning (ECML 2018) and Principles and Practice of Knowledge Discovery in Databases (PKDD 2018), Dublin, Ireland, September 10th, 2018.* Pp. 36–47.

[74] Settles, Burr. "Active learning literature survey". In: (2009).

[75] Sharma, Shiven, Bellinger, Colin, and Japkowicz, Nathalie. "Clustering based one-class classification for compliance verification of the comprehensive nuclear-test-ban treaty". In: *Advances in Artificial Intelligence: 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30, 2012. Proceedings 25*. Springer. 2012, pp. 181–193.

[76] Spitz, Andreas, Almasian, Satya, and Gertz, Michael. "TopExNet: Entity-Centric Network Topic Exploration in News Streams". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019, pp. 798–801.

[77] Spitz, Andreas and Gertz, Michael. "Exploring Entity-centric Networks in Entangled News Streams". In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 555–563.

[78] Thakkar, Harsh and Patel, Dhiren. "Approaches for sentiment analysis on twitter: A state-of-art study". In: *arXiv preprint arXiv:1512.01043* (2015).

[79] Unnikrishnan, Vishnu et al. "Entity-Level Stream Classification: Exploiting Entity Similarity to Label the Future Observations Referring to an Entity". In: *Data Science and Advanced Analytics (DSAA), 2018 IEEE International Conference on*. to appear. IEEE. 2018.

[80] Vakulenko, Svitlana, Weichselbraun, Albert, and Scharl, Arno. "Detection of Valid Sentiment-Target Pairs in Online Product Reviews and News Media Articles". In: *2016 IEEE/WIC/ACM Int. Conf. on Web Intelligence*. IEEE, 2016. DOI: 10.1109/WI.2016.24.

[81] Vinagre, JOao, Jorge, Alipio Mario, and Gama, Joao. "An Overview on the Exploitation of Time in Collaborative Filtering". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.5 (2015), pp. 195–215. ISSN: 1942-4795.

[82] Wagner, Sebastian et al. "Ageing-based Multinomial Naive Bayes Classifiers over Opinionated Data Streams". In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECMLPKDD'15*. Vol. 9284. 2015, pp. 401–416.

[83] Xiang, Qiuyan et al. "Concept drift adaptation methods under the deep learning framework: A literature review". In: *Applied Sciences* 13.11 (2023), p. 6515.

[84] Yu, Hang et al. "Automatic Learning to Detect Concept Drift". In: *CoRR* abs/2105.01419 (2021). arXiv: 2105.01419. URL: https://arxiv.org/abs/2105.01419.

[85] Yuan, Lanqin, Pfahringer, Bernhard, and Barddal, Jean Paul. "Iterative subset selection for feature drifting data streams". In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018*. 2018, pp. 510–517.

[86] Zhang, Lei and Liu, Bing. "Aspect and Entity Extraction for Opinion Mining". In: *Data Mining and Knowledge Discovery for Big Data: Methodologies, Challenge and Opportunities*. Ed. by Wesley W. Chu. Springer, 2014, pp. 1–40. DOI: 10.1007/978-3-642-40837-3_1.

[87] Zhang, Peng et al. "SKIF: a data imputation framework for concept drifting data streams". In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. 2010, pp. 1869–1872.

[88] Žliobaitė, Indrė et al. "Evaluation methods and decision theory for classification of streaming data with temporal dependence". In: *Machine Learning* 98.3 (2015), pp. 455–482.

A.3. Nutzung von generativer KI für Abschlussarbeiten

In dieser Arbeit wurde Grammarly[1] verwendet, um die Rechtschreibung und den Satzbau zu verbessern. Des Weiteren wurde ChatGPT-4o[2] verwendet, um Lesbarkeit und Verständlichkeit des Textes zu erhöhen. Dabei wurde wie folgt vorgegangen:

1. Einzelne Textpassagen wurden von mir voll ausgeschrieben.
2. ChatGPT-4o hat Anpassungsvorschläge unterbreitet
3. Vorschläge wurden überprüft, dass sie den Originaltext nicht verfälschen.
4. Je nach Qualität wurden Abschnitte oder einzelne Sätze übernommen.
5. Die übernommenen Teile wurden ggf. angepasst.

Generative KI kam nicht zum Einsatz, um Code, Abbildungen oder Auswertungen zu erstellen.

[1] `https://app.grammarly.com/`
[2] `https://openai.com/index/hello-gpt-4o/`

Ehrenerklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 24.09.2024

Christian Beyer