Utrecht University

Master Thesis

IDEAL: Integrating the Detection of changes, Exploration of instance space, and Assessment of performance into Active Learning

Author:
Misja Groen *(7024010)*

Supervisors:
Ass. Prof. Dr.-ing. Habil. Georg Krempl
Prof. Dr. A.P.J.M. Siebes

MSc Business Informatics
Graduate School of Natural Sciences
Utrecht University

June 27, 2024

Groen, Misja :
IDEAL: Integrating the Detection of changes, Exploration of instance space, and Assessment of performance into Active Learning
Master Thesis, Utrecht University, 2024.

Abstract

With Active Learning becoming more used in the context of data streams, there is also an increasing need to have an algorithm appropriate for this setting. As such, the active learner must be able to handle the challenges that arise in data stream-based settings.

This research aims to determine if an optimal Active Learning algorithm can detect changes unsupervised, assess its performance, and be representative of the distribution of the data stream. When combined, a trade-off in the weighting of the different components also needs to be considered.

This master's thesis proposes an active learning algorithm named IDEAL. IDEAL aims to operate in an evolving data stream where changes happen continuously based on three components. IDEAL aims to do this by **I**ntegrating the **D**etection of changes, **E**xploration of instance space, and **A**ssessment of performance into **A**ctive **L**earning.

One synthetic data set and three real-world benchmark datasets have been used to evaluate the performance of IDEAL. IDEAL performed quite well, but it was not the most accurate Active Learning algorithm.

For future research, IDEAL is a good baseline for making a better Active Learning algorithm. The components' weightings are scalable, and the option to add more or fewer components makes it easy to adjust the algorithm. More research into either the weightings between the three components or the Change Detector and Explorer components of IDEAL would be good.

Acknowledgements

I would like to express my deepest gratitude to Ass. Prof. Dr.-ing. Habil. Georg Krempl for his patience, expertise, support and feedback throughout this thesis. I would also like to thank my second supervisor, Prof. Dr. A.P.J.M. Siebes, for their expertise. Additionally, I am grateful to Noah-Manuel Michael for giving feedback and keeping my motivation high.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Statement and Research Objectives

Over the years, the creation, collection, and transformation of (digital) data have increased. In 2025, more than 180 zettabytes will be created, consumed, and stored, while only 64.2 zettabytes in 2020 [47]. Moreover, Google processes 100 billion searches per month [46]. These examples show that there is an increasing need for data querying, gathering, and transforming the data. Data labeling resources must be used more efficiently as the data volume grows. As such, there is a higher need to work with unlabeled data and ensure that labeled data is available when needed. Active Learning is one of the techniques that can help with the massive cost of labeling data and working with a small set of labeled data. Active Learning in Data Streams is a research direction that faces many challenges. Several papers outlined multiple challenges in Active Learning in Data Streams as described in Table 1.1. The problem explored is that many algorithms can do unsupervised tasks, sometimes combined with supervised methods. However, detecting changes in the labels or distribution unsupervised is challenging [6]. This thesis aims to integrate unsupervised and supervised components to efficiently resolve this task. The algorithm should select new unlabeled instances for labeling. Labeling them is expected to improve performance, considering the moment of observation and spatial location. The algorithm should also explore the feature space to detect changes that did not alter the feature distribution.

1.2 Research Questions

In this section, I describe the research questions of this study. The study's main objective is to uncover an active learner which performs well in a data stream-based setting. I propose one main research question and four sub-research questions that will help explore the different characteristics an optimal active learner should have. The main research question is *Is there an optimal active learner*

Tharwat et al. [48]	**Cacciarelli and Kulahci [6]**	**Krempl et al. [28]**
Concept Drift	Data Drift	Skewed Distributions (Due To Drift)
Noisy Labeled Data	Labeling Quality	Handling Incomplete Information
Low Query Budget	Algorithm Scalability	Delayed Information
Variable Labeling Costs	Model Interpretability	Costly Information
Initial Knowledge Training Model	Evaluation	Uncertainty Convergence
Imbalanced Data		Perpetual Validation
Multilabel Applications		Temporal Budget Allocation
Stopping Criteria		Performance Bounds
Outliers		
High-Dimensional Environments		
ML-based Active Learners		
Crowdsourcing Labelers		

Table 1.1: Challenges described

in an evolving stream-based setting? The four sub-research questions will discover how to detect changes in an unsupervised manner, explore the feature and instance space, assess the performance, and integrate the three different aspects.

SRQ1: *How can an algorithm detect changes in the unlabeled data in an unsupervised manner?*

Active Learning and Data Streams algorithms need to be able to detect changes in an unsupervised manner. There is a lot of unlabeled data with free access available, and labeled data is costly to get. It needs to detect changes in the distribution of data so that it gets triggered to investigate what these changes mean. The changes might indicate noise, but they might also mean a shift in the distribution, concept drift, or that a new unknown class has been identified.

SRQ2: *How can the Active Learning algorithm ensure that it is representative of the current distribution while not running out of labeled instances itself?*

At any given time, there is a training set of labeled instances. However, these instances age and might become outdated. So, there needs to be a forgetting mechanism. As a consequence, at some point, there might not be enough labeled instances in the training set anymore, unless new instances are queried and labeled. However, the training set of labeled instances needs to be representative of the current distribution of unlabeled and labeled instances.

SRQ3: *How can an active learner estimate the current classification performance at a given location in the feature space?*

An active learner in a data stream-based setting needs to be able to assess the performance of the current classification model at spatio-temporal locations (that is, instances sampled at particular moments with particular feature values).

SRQ4: *How can the different components be combined into one algorithm to evaluate it?*

The result of each of the previous sub-research questions will lead to a certain component of the algorithm that needs to be integrated. This will lead to an imperfect solution. An investigation into the effects of each of the components and its trade-off is warranted. For example, the three components could have a weighted average, which needs investigation, as a threshold or the threshold could be maximized for each of the components. This sub-research question explores the integration between the detection of change, the exploration of the instance space, and the assessment of performance in Active Learning.

These sub-research questions aim to answer the main research question and find the most optimal active learner in an evolving data-stream-based setting.

This research project focuses on Active Learning, evolving data streams and the most optimal algorithm for this setting.

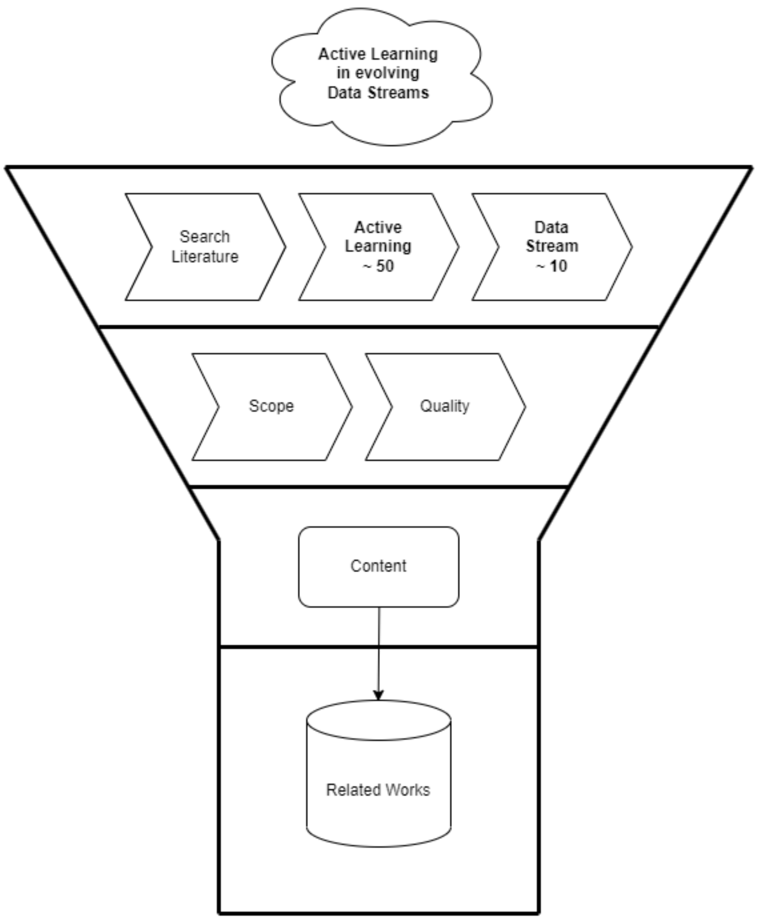


Figure 1.1: Literature Review procedure

1.3 Literature Review

The literature review is exploratory for this research. The first step is broad, where the question is to look for the current state-of-the-art regarding Active Learning and Data Streams without any restrictions. This results in many topics, such as Deep Learning and Bayesian Active Learning. The focus is to find surveys on the topics of Active Learning and Data Streams. I have found about 50 papers on Active Learning and 10 on Data Streams. However, many of those papers are not in the scope of this research because they do not relate to the topic or do not have an algorithm that can be of use. I also rejected a few papers because they were not good in terms of quality. I picked a few surveys and other papers as a baseline and papers to explain the concepts of Machine Learning, Active Learning, and Data Streams. When additional literature has to be used to fill gaps, I will use snowball sampling or look into other papers by known authors in the field of Active Learning and Data Streams. See Figure 1.1 for an illustrative overview.

1.4 Structure

The remainder of this paper is structured as follows: In Chapter 2, the related works and background are discussed and the current state-of-the-art is explored with regards to Active Learning and Data Streams. It critically reflects on which challenges remain for Active Learning in Data Streams. In Chapter 3, the experiment is outlined. The three different components and their integration are outlined. Which is used for the experiments. In Chapter 4, the results of the experiments are discussed. The results are split into two parts: the experiments on the synthetic dataset and the experiments on the real-world benchmark datasets. In Chapter 5, the results are discussed, and directions for future research are given.

2. Background and Related Work

This chapter will discuss the relevant literature for this thesis and give an overview of the current state of the art. First, Active Learning is covered, where its origins and relation to other fields are presented. Active Learning can be used in different use cases and scenarios. It is most appropriate in situations where getting labeled data is costly. Different strategies are presented for some use cases or scenarios, and different Active Learning approaches are discussed.

2.1 Machine Learning

One of the definitions of Machine Learning is *the study of computer systems that improve with experience and training.* This means that, over time, a computer system should improve by trial-and-error or knowledge of other parties that it can use as input. A Machine Learning problem can consist of a Task T, a Performance measure P, and the Training experience E. For example, a task T can be to predict what number a dice will roll. The performance measure is the probability that a dice rolled a given number, and the training experience E can be a dataset of 100 dice rolls. As such, the algorithm can predict what the probability is going to be for a dice roll of 5 [35]. Another definition is done by one of the pioneers of Machine Learning, Arthur Lee Samuel, Machine Learning is *the field of study that gives computers the ability to learn without explicitly being programmed* [39]. Without explicitly being programmed is implying that a computer system is learning or improving, which should not be hard-coded and should be learned organically. Machine Learning is based on algorithms that create a model based on training data and can execute certain tasks. These tasks can consist of making predictions and decisions. In some contexts, Machine Learning will also be referred to as predictive analytics, which has several appliances on its own. Predictive analytics is a field of business analytics that encompasses a variety of statistical techniques from modeling, Machine Learning, and data mining that analyze current and historical facts to make predictions about future or otherwise unknown events [16].
Machine Learning has several approaches: Supervised, Unsupervised, and Reinforcement Learning. Supervised Learning is based on training a model on

labeled data, where the training data guides the learning process and the model executes it on unlabeled data, which is often referred to as test data [24]. Supervised Learning can exist between Transductive Learning and Inductive Learning. In Transductive Learning, the instances the model tries to predict have already been seen during the training phase. In Inductive Learning, the model tries to predict it based on instances it has not seen yet. Supervised Learning algorithms learn a function *(f)* by assigning input variables or features *(X)* from training data to a target *(Y)* [38]. Supervised Learning involves labeled training data employed in a model and then produces a certain output. In Supervised Learning there are different categories, the two most well-known of them are classification problems and regression problems. In classification problems, the output data is a class, and as input data, it uses discrete data, which is data that can only take certain values and has a limited number of values. For example, a classification problem can be detecting whether emails are spam or not. Also, one can have regression problems where the output data is a number; however, they differ from the classification since the range of numbers does not have to be limited and can be continuous. For example, regression can be used to predict sales revenue based on historical sales data from the company. The main advantage of Supervised Learning is that the algorithm can predict based on already labeled data and learn from that. However, the drawback is that this labeled data needs to be available for training and be of good quality to be accurate. As such, it can be expensive or time-consuming to label the data, and Active Learning might help in this case.
The definition of Unsupervised Learning is to directly infer the properties of the data without the help of a supervisor, so it does not require labeled data and processes unlabelled or raw data [24]. Unsupervised Learning is often considered the opposite of Supervised Learning since it does not need labeled data. It explores the data and discovers hidden patterns inside that would not have been observed otherwise. This differs from Supervised Learning since the goal is to predict an outcome, while unsupervised Learning aims to describe the patterns and associations. Examples of Unsupervised Learning techniques are association rules and cluster analysis. Association rules have been a popular method for commercial businesses to gain insight into customers' purchases. In this field, it is also referred to as *market basket* analysis, where the value can only take up to two values. [2, 24]. Another technique used in Unsupervised Learning is cluster analysis, where all instances are grouped or segmented into clusters based on how closely related they are to one another. This can be done on certain input parameters or the position of the instances on a plot, for example. The key takeaway for cluster analysis is that it will not look at the details and labels of the data but will try to infer groups or clusters based on the data. As such, it is used as a form of descriptive statistics to discover certain groups of instances. It can be interesting to see which groups of customers there are in your business. The advantages of Unsupervised Learning are that it has no explicit need for labeled data, it can be used in situations where labeling is costly or difficult to do, and it can uncover patterns and relationships that otherwise would be hidden. One of the limitations of Unsupervised Learning is the difficulty of measuring the performance of the algorithms since there is no correct data to compare the results to. The last category of Machine Learning

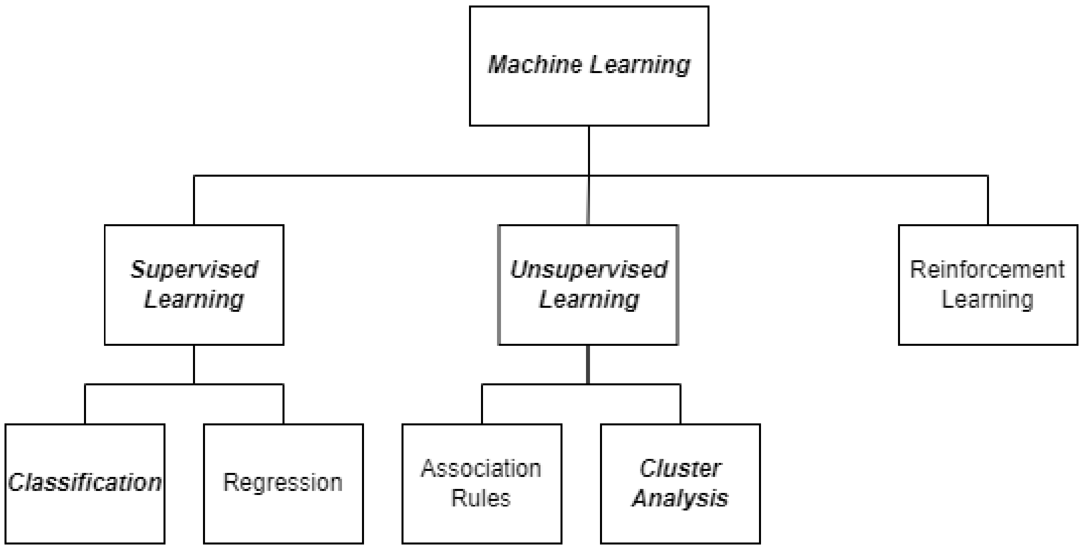


Figure 2.1: Machine Learning categories

is Reinforcement Learning, when a computer program does not get any training data and learns based on exploring an environment through actions and states. The learner receives feedback and receives a form of reward if the model performs well. This last category is out of the scope of this research and, as such, will not be discussed.

In Figure 2.1, the different branches of Machine Learning are plotted. However, what happens when you have a dataset consisting of both labeled and unlabeled instances? Semi-Supervised Learning *(SSL)* can help, which finds itself between Supervised and Unsupervised Learning. This learning technique can sometimes handle unsupervised, unlabeled data and supervised, labeled information. Consider for SSL the dataset $X = (x_i)_{i \in [n]}$. This dataset can be divided into two parts, X_l, in which the labels Y_l are provided, and X_u in which the labels are unknown. This is described as the typical SSL setting [10]. There are two approaches when using Semi-Supervised Learning as described by [10], the first being a model that can be used for prediction as seen in supervised learning, or the second one being a model that can be used to find structure or patterns in the data as seen in Unsupervised Learning. Semi-supervised Learning has several assumptions to work. Firstly, the semi-supervised smoothness assumption: "If two points x_1,x_2 in a high-density region are close, y_1,y_2 should also be close". Secondly, the cluster assumption: "If points are in the same cluster, they are likely to be of the same class". Thirdly, the manifold assumption: "The data lie on a low-dimensional manifold". This last assumption is specifically helpful with the curse of dimensionality, which is related to the fact that the volume grows exponentially with the number of dimensions. As a result, the volume may grow too large for (generative) models. Semi-Supervised Learning can have several methods and goals, one of them being classification using Expectation-Maximization techniques. The book Semi-Supervised Learning by Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien [10] describes an experiment with classification using EM. They claim that SSL with labeled and unlabeled data are more accurate in-text classifiers than Supervised Learning with only labeled data. However, the model selection process is difficult between the complexity of the model and the data sparsity. It is difficult to make a trade-off between generalizing all the topics of the texts or getting the topics of all the individual texts. The other challenge is that the local maxima that are discovered with a few labeled examples are far lower than the classification accuracy with a lot of labeled data. So one should look into how the accuracy can increase with a small amount of labeled data. One of the proposed discussion points is to include a human factor into the equation which is labeling the class correspondence. This can then help to map a fraction of class labels to have higher accuracy. A framework for this exists in the form of *Active Learning.*

2.2 Active Learning

Active Learning is a subfield of Machine Learning where an algorithm chooses the data it learns, and then an oracle labels the data. For example, an oracle can be a human expert or another information source. Active Learning aims not to label as many instances as possible but only those instances that benefit the model. An oracle labels the unlabeled instances that are queried by the Active Learning algorithm and as such, it can reach a high accuracy with only a small fraction of labeled data. This can be helpful in situations where the labels are costly to obtain [42]. Active Learning often starts with a large set of unlabeled instances and a small set of labeled instances. Based on the query strategy, the labels are queried for some instances. It can select the instances that the active learner finds the most informative and query those to the external oracle to label. There are multiple ways to measure usefulness, and each Active Learning strategy has a different way of querying the most informative instance. After that, the model modifies its hypothesis based on the labeled instances to correspond to the newly added labels [42, 48].

Definition Active Learning:
Consider X to be the whole set of data. For example, when performing a sentiment analysis for customer reviews, X can be the total set of customer reviews available. The idea is that customer reviews should be annotated for whether they are positive or negative. The reviews are being labeled by humans, and this is costly in terms of time and money. Active Learning assumes that for every candidate instance, there is a label that can be acquired. In each iteration of the Active Learning process, X is divided into two subsets:

X_{Ui} is the set of all unlabeled instances where the review sentiment is unknown;
$X_{Ci} \subseteq X_{U},_{i}$ is the set of all candidate instances that can be labeled by the external oracle. [1]

Tharwat [48] describes that any active learner has four different components, which are:

[1]An oracle can also be referred to as an *expert, labeler* or *annotator* in other papers; however, in this thesis, the term oracle is used.

- Data, which are the unlabeled and labeled instances;
- Learning algorithm, which is the algorithm used to evaluate the model and find the instances the model needs;
- Query Strategy, which is the strategy it uses to select and query the most informative instances;
- Oracle, which is the machine or human expert labeling the queried instances.

2.2.1 Active Learning Strategies

There are three main Active Learning strategies described by Settles [42, 43], which are Membership Query Synthesis, Stream-Based Selective Sampling, and Pool-based Sampling. An Active Learning strategy is a scenario in which the technique is being applied.

In the *Membership Query Synthesis* strategy, the learner requests synthesized queries and requests labels for those queries. These queries are then passed through an external oracle, which labels them. One of the advantages of this strategy is that the model is completely free to generate any query it wants. The active learner can choose any label of any instance that belongs to the input space or which is synthetically generated [6, 48]. This strategy can be used efficiently when problems are finite since it can then select the most informative instances from a problem space [42, 48]. The main advantage of this strategy is that it can quickly generate new queries because there is no processing on the unlabeled data, and as such, the algorithm can generate queries quickly. The drawback is that it can generate instances for which the oracle finds impossible to label. An example of this can be handwritten characters that can be impossible to distinguish or when having to predict the absolute coordinates of a robot hand. There are scenarios when Membership Query Synthesis can be applied, such as in Natural Language Processing. In this field, Membership Query Synthesis has successfully generated queries using Variational Autoencoders, a generative model used in neural networks [41]. The authors used text classification, and the strategy generated instances with a claim that the model can generate instances in the form of sentences. The sentences are easy to read and not hard to distinguish. Out of the ten example instances generated, one of them is skipped due to being difficult to label. In short, Membership Query Synthesis can generate a (synthetic) instance from the *(instance space)* and then queries it for the oracle to label

In the *Stream-Based Selective Sampling* strategy, the learner samples an instance, which is drawn over time in a data stream. When it observes an unlabeled instance, it will then consider its usefulness and decide to query it or discard it. Because the instances come in the form of a data stream and, as such, come in one by one, this method is also called Sequential Active Learning. This scenario can be used in situations where computational options are limited, and since instances are sampled sequentially, it does not require large processing power [48]. This strategy selects the sample instance from a real data distribution instead of Membership Query Synthesis, where it can also be a synthetic instance. One method to do selective sampling is to set a minimum threshold on how informative an instance should be, and every instance that is above that threshold is queried [11, 42]. One example of stream-Based Active Learning is in the financial domain, where they adopted the SVM classifier for sentiment analysis. The adaptation of the SVM classifier is done by determining the best querying strategy. They applied Active Learning because there is no or only a limited number of labels available for tweets in the financial domain, and labeling all of them manually is costly. The drawback of the method is that it is prone to drifts, sarcasm and irony [45]. This is also a drawback of Stream-Based Selective Sampling in general since this strategy is prone to concept drift, which is going to be discussed in more detail later. Concept drift may lead to a model not being fully applicable anymore since the data distribution has changed. In short, stream-Based selective sampling can sample an instance sequentially, where based on some threshold an instance is either discarded or queried to an oracle.

In the *Pool-Based Sampling* strategy, the learner samples a pool of instances of which the informativeness is measured and queries some of the available unlabeled data in the pool to the oracle. In Pool-Based Sampling, there is a set of labeled data D_L and a large pool of unlabeled data D_U.The labeled data is used to train the model and evaluate the informativeness of the unlabeled points in D_U. In pool-based sampling, the unlabeled instances are queried, and a score is computed for each instance to evaluate its informativeness. The active learner then queries one or more instances, depending on the strategy. After labeling it, the instance is added to the labeled set. The difference between Pool-Based Sampling and Stream-Based Selective Sampling is that all instances are put together in a pool in Pool-Based Sampling. The active learner then evaluates the instance by a score before querying it. Stream-based selective sampling will query the instances sequentially and decide whether to query them or discard them one by one. The main advantage of Pool-Based Sampling as opposed to Stream-Based Selective Sampling is that it chooses unlabeled instances from a selected pool of unlabeled instances. Then, the active learner ranks them, and as a result, it can efficiently process a lot of unlabeled instances. The drawback of using this strategy is that it may be computationally intensive since it has to rank or order a pool of instances. [42, 43, 48]. Pool-based sampling is common in a lot of real-world adaptations of Active Learning, such as text classification [34], image classification [29], and in the classification of cancer [30].

When looking at the three scenarios in Active Learning, each of them has its own purpose, advantages, and drawbacks, see Table 2.1. These scenarios are different methods of how Active Learning can be applied, but for each of the scenarios, there is also a query strategy before the instance to be queried reaches the oracle. The next chapter covers different frameworks to discuss different querying strategies and their current research gaps.

2.2.2 Active Learning Approaches

When looking at Figure 2.2, one can see that an Active Learning process, in its most simple form, consists of an instance space to which all the instances belong.

Active Learning Scenario	**Advantage**	**Drawback**
Membership Query Synthesis	Quickly generate new (synthetic) instances	Difficult to label instances
Stream-Based Selective Sampling	Generally little need for large processing power	Prone to concept drift
Pool-based Sampling	Efficiently process a pool of unlabeled instances	Can be computationally intensive

Table 2.1: Active Learning Scenarios

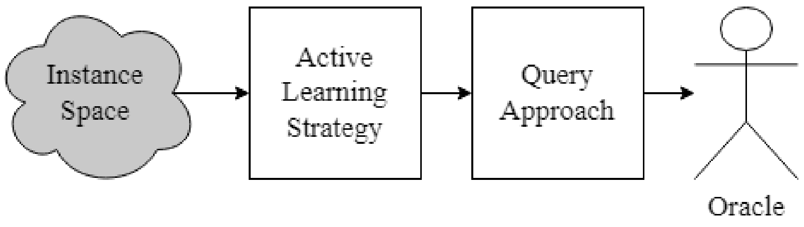


Figure 2.2: Active Learning overview

It also consists of an Active Learning strategy covered in Chapter 2.2.1. Lastly, it consists of an oracle, which is the expert that labels the instances queried by the query approach. In this chapter, the different Active Learning query strategies are discussed. Query strategies have been divided in several ways. The first query strategy categorization example is by Tharwat and Schenck [48], they have separated the strategies into information-based techniques and representation-based techniques:

- In the *information-based* techniques, the query strategy is looking at the most informative points, which means that depending on the strategy, an instance is evaluated on the information it holds. This can be, for example, how uncertain the strategy is about the instance with regard to its position in the instance space or which instances the queries disagree with the most.

- The *representation-based techniques* do not look at the instances individually but more at the instances as a whole. They look at the instance space and which information they hold all together. The goal is to discover which instances are the most representative to query. For example, the decision can be made to query the instances that are the least distant from other instances or query to discover which clusters exist in the instance space.

The second categorization, introduced by Schröder and Niekler [40], is one where the strategies are split up into the categories random, data-based, model-based, prediction-based, and ensembles.

- *Randomness* has been used as a baseline for Active Learning, where it just samples random instances and can be used to compare other techniques to.
- *Data-based* techniques only make use of the raw data and they can use labeled instances. They divide this category further into *data-uncertainty* which is concerned with the information about the data, and *representativeness* which is concerned with selecting the minimum amount of instances to represent the instance space.
- *Model-based* techniques utilize both the information about the data and the model.
- *Prediction-based* techniques use the prediction output to select the instances, so the algorithm asks how likely it is that a model is uncertain about a specific instance.
- The last category described are the *ensembles*, where multiple other strategies' outputs are combined. However, this category can include the data-based, model-based, and prediction-based strategies, and as such, is not as distinct of a category.

The last categorization made is proposed by Cacciarelli and Kulahci [6], where Online Active Learning strategies are split into four subcategories.

- *Stationary data stream classification approaches*, which are online classification methods that can handle data streams that do not change significantly over time;
- *Drifting data stream classification approaches*, which are classification methods that can handle data streams whose distributions change constantly;
- *Evolving fuzzy system approaches*, which are strategies that can respond to changes in the environment or new data by adapting the system;
- *Experimental design and bandit approaches*, which are strategies that select the most informative points and are used for prediction.

In this section, I will discuss two of Tharwat and Schenk's categories, information-based and representation-based.

2.2.2.1 Information-based Strategies

The first category is the information-based query strategies, which search for the most informative points. Examples are the uncertainty strategy, where the most informative points are measured by uncertainty, or the Query By Committee, where a set of models looks for which classes belong to which points, and the one they disagree the most upon is queried. The next section briefly introduces these query strategies.

Uncertainty Sampling is one of the most common query strategies used in Active Learning. The query strategy looks for the most informative points around the decision boundary. Around the decision boundary, instances are often close to each other, and as such, a model is less certain about their classes. If an instance is far from the decision boundary, the model would be fairly certain about its class. As such, it does not make sense to query points that are far from the decision boundary. That is why it will query the instances close to the decision boundary. In this strategy, there are multiple approaches to measuring uncertainty. The first approach to measure uncertainty is *least confident*, where the instance is queried about which the model is least confident or most uncertain. The drawback of this method is that only the least confident point is taken into account, so the information about the rest of the instance space is ignored. The second approach is *margin sampling* where not only the least confident point is measured, but also the second least confident, and as such, these two points can be compared to each other. However, the drawback is that the rest of the instance space is still ignored when there are many second least confident instances. The last and most commonly used approach is *entropy*, where an instance is covered by all possible labels, and a distribution of all probabilities can be plotted. Entropy then selects the highest value since this is the most uncertain instance. This approach considers the whole set of predicted probabilities, and the other two approaches only use the highest two probabilities [43, 48].

Query By Committee is another commonly used query strategy. This strategy consists of a set of different models which are all trained on different subsets of the instance space samples. After the training, the models are combined to see which instances the models disagree the most on. These are the most informative points and are queried. This query strategy is also referred to as version space where hypotheses are tested against one another. A query strategy closely related to QBC is *Query By Bagging*, where as opposed to Query By Committee, the same classifier is used by each committee member and the disagreement is measured by the Kullback-Leibler divergence, vote entropy, and posterior probability entropy. The aforementioned measures of disagreement can also be used for QBC [1, 48].

One can also look at other information-based strategies such as the expected model change, where points that produce the largest change in the model without considering the label are queried. These techniques are expected gradient length and weight changes [48]. Another one is the expected error and variance reduction, where the goal is to query the instance that has the lowest expected error or variance when the values and probabilities of all predicted labels are calculated and aggregated as a sum [43].

2.2.2.2 Representation-based Strategies

While the information-based strategies search for the most informative points, the representation-based strategies try to query the points based on the distribution. The takeaway from this is that the points that are queried are representative of the distribution to some extent. The first representation-based strategy is the *density-based* strategy, where the regions with the highest density are the most representative points. One of the techniques that can be used is to calculate the distance between the points, such as the Euclidean distance [43]. The second representation-based strategy is the *cluster-based* strategy. Where density-based looks at high-density regions, cluster-based looks at clusters. The strategy tries to make clusters of the data and queries some points within the clusters to indicate how many classes there are. One of the techniques used is hierarchical sampling [13], which is similar to a tree-based method where a tree is found with subtrees and random instances are queried from a subtree. After that, it is calculated if the tree should require further pruning or if the tree's current clusters are sufficient. The last representation-based strategy is the *diversity-based* approach [5], which focuses on mitigating the problem when labeling is done in parallel. The problem is it will sometimes be redundant since the same instances can be queried. This approach queries the instances where the points are different enough from the rest of the instances. However, the drawback of this method is that it results in outliers being queried, and as such the performance needs improving [48].

2.2.3 Active Learning Settings

A foundation for Active Learning has been laid, including how it works, which approaches it can take, and which query strategies it can deploy. This section will cover the current state-of-the-art research, applications, and research directions of Active Learning relevant to this thesis.

When first looking at Semi-Supervised Learning, it was stated that Active Learning can help when a human factor needs to be included. However, it does not have to be that one works without the other, and both techniques can be employed. For example, in the paper by Zhu, Lafferty, and Ghahrama [53], they discuss the possibility of executing a Semi-Supervised Learning task and employ an Active Learning task on top of it in order to mitigate the classification error. They find that the active learner is much more efficient because it needs fewer labels than the randomized query strategy. Another example of combining Semi-Supervised Learning and Active Learning is to employ co-training, a Semi-Supervised Learning method, which selects the most confidence and nearest neighbors instances. Then Active Learning is employed on top of it by using human annotation to also increase the classification error [52]. Both methods looked at employing Active Learning on top of Semi-Supervised Learning; however, there is a possibility that no training set is available. In that case, Semi-Supervised Learning would not be able to be employed, and a human annotator would have to select some queries before Semi-Supervised Learning techniques are applied. However, applying only Active Learning techniques or combining them with unsupervised ones would make more sense.

Another situation where Active Learning can be applied is with Deep Learning, which uses Artificial Neural Networks where multiple layers are used for feature extraction from the input data. However, while doing this, it requires a lot of labeled data, so the labeling cost is very high. Combining Active Learning and Deep Learning offers the possibility of getting labeled data through oracles. There are many challenges in this field to overcome one of the main challenges is that the processing of the features and classifiers is not aligned between Active Learning and Deep Learning. In Active Learning, the classifiers are trained, but in Deep Learning, the classifiers and features are trained [48].

As already described, Active Learning can be used in Data Streams. Active Learning is also referred to as Stream-Based Selective Sampling or Online Active Learning. In this approach, the instances arrive one by one or in a batch sequentially. In Figure 2.3, one can see that from the instance space, the data stream unlabeled instances are observed and are queried either one by one or stored in a batch before querying. After that, it may be queried to the oracle. Where after being queried the model used is updated to the current dataset. There is an assumption made in this model, that is, every instance observed is queried and labeled. The batch or window-based data stream is useful when decision-making or labeling of the instances is not time restrictive, whereas the one-by-one-based data stream is useful when the labeling has a much higher need [6]. Active Learning in data streams has several properties, such as:

- the aforementioned difference in one-by-one or batch-based;
- the data stream distribution
  - stationary data streams, a data stream where the distribution remains fairly consistent;
  - non-stationary data streams, a data stream where the distribution tends to drift;
- can have a *label delay*, the delay that occurs when an oracle has to label an instance;
- training efficiency
  - incremental training, updating the model with new data while conserving current knowledge;
  - complete re-training, creating a new model with the data using the labeled dataset.

Data streams and Active Learning overlap and can be used in many ways. However, there are still open research directions. In the next chapter, those topics are discussed.

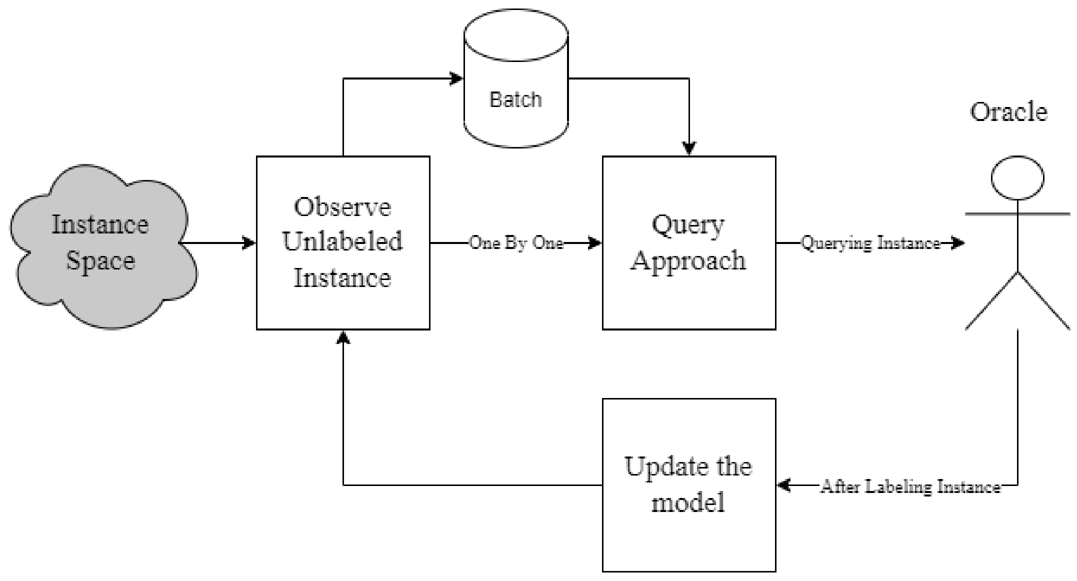


Figure 2.3: Active Learning data stream

2.3 Non-stationary Environments

As already described, a data stream can be either stationary or non-stationary. However, the definition of a data stream is still ambiguous. In this thesis, a data stream is defined as:

Definition data stream:
Consider S to be the whole set of all possible instances in a data stream. Where a data stream is a sequence of instances (s_1,s_2, ..., s_n) where each instance belongs to S. Any instance with the data stream comes in sequentially and, as such can be accompanied with a timestamp of arrival. Consider A to be the set of all instances $s \in S$ and timestamps $t \in T$ with the elements ($\{s_1,t_1\}$, $\{s_2,t_2\}$, ..., $\{s_n,t_n\}$) where s_n and t_n are the nth element in the data stream.

When looking at data streams, three challenges are present, *volume, velocity* and *volatility* as described in the introduction. One of the challenges with volatility is that data will eventually become outdated and be of limited use. This is because of three aspects, *change of target variable, change in the available feature information* and *drift* [28]. Changes in target variables can happen due to changes in requirements. For example, in a business, what is successful in one year can be unsuccessful in another year because of the requirements. Changes in available feature information can happen due to new knowledge becoming available. For example, when a new MRI machine with new technology replaces the old one. Lastly, drift can happen when the distributions of the instances change over time. One example is when criminals change methods or patterns every time because crime detection is becoming more efficient. The problem of changing distributions exists in Unsupervised Learning, Supervised Learning, and Active Learning.

Before continuing with drift, there are multiple approaches to handling stationary data streams in Active Learning. One method is the Selective Sampling

Perceptron proposed by Cesa-Bianchi [9] where the probability of a certain instance being queried is calculated. When an incoming instance is observed, the label is predicted and then a Bernoulli random variable with a parameter, a binary value between 0 and 1, is drawn. When that value is 1, then its label is queried to an oracle. However, one drawback of this method is that it is inefficient since for every instance its label is predicted. Research has built further on this method, where Dasgupta et al [14] suggests a margin threshold, Lu et al [32] suggests a passive-aggressive version which updated the model also on correctly classified labels but with low confidence with regards to its prediction. Challenges with these methods are problems such as class imbalance, noisy labels, and difficulties with capturing complexity and diversity in the data representation. Ensembles and committees of approaches have been researched to overcome the difficulties of complexity and diversity. One of the examples is Online Active Learning with expert advice [22]. This method minimizes the number of queries so that the predictions are closer to the fully supervised setting using advice from a pool of experts. They used the exponentially weighted average forecaster and greedy forecaster with a confidence threshold to know when to query for certain labels. However, this method's challenges include noise or experts who provide labels of insufficient quality [6].

2.3.1 Concept Drift

Drift or concept drift is a phenomenon heavily researched in Machine Learning, data streams, and other fields such as Process Mining [4]. In process mining, four classes of concept drift are distinguished:

- *sudden drift*, when a new process all of a sudden substitutes the old process;
- *recurring drift*, when a set of processes seem to reoccur every now and then;
- *gradual drift*, when a new process gradually replaces an old process;
- *incremental drift*, when substitution of an old process is done in incremental changes.

One can also classify these same drifts for data streams, such as:

- *sudden drift*, when a new data stream all of a sudden substitutes the old data stream;
- *recurring drift*, when a set of data streams seem to reoccur every now and then;
- *gradual drift*, when a new data stream gradually replaces a data stream;
- *incremental drift*, when a new data stream replaces a data stream in increments.

This is also how Cacciarelli et al. [6] categorized the concept drifts. The paper by Krempl et al. [28], also classifies drifts which can affect the posterior (the probability of an event given some observed data), the conditional feature (a variable used to predict other variables of interest), the features (the attributes), and the class prior distribution (the initial distribution of the classes). They categorize the drifts into four different categories:

- *smoothness of concept transition*, the distinction between sudden or gradual concept drifts;
- *singular or recurring contexts*, the distinction between a data stream being replaced or data streams reoccurring;
- *systematic or unsystematic*, the distinction between concept drifts having a pattern or being randomized;
- *real or virtual*, the distinction between model adaptation or observing outliers and noise.

Much research has been done on concept drifts, Active Learning, and Data Streams. The first batch of research is concerned with combining Active Learning strategies with drift detectors, which are algorithms that can detect drift, such as changing distributions or changing environments.

There have been several attempts to detect drifts in drifting data streams, such as the use of the drift detection method or the adaptive window strategy (ADWIN). Caciarelli et al. state that real concept drifts can hardly be detected in a completely unsupervised manner [6]. Krawczyk et al. [26] propose a method using the ADWIN strategy to detect drifts and a threshold to balance the budget over time. This is done because when a concept drift is detected, more budget needs to be allocated to label instances. Castellani et al.[8] also proposed a similar threshold but also focused on the verification latency or the labeling delay. By looking into the spatial information of an instance, it does not make sense to query labels for instances in which the label is already known. Another look at the verification latency is proposed by Pham et al. [37], where a method is proposed to forget outdated information and to simulate the delayed labels since it may be the case that an instance is already outdated because it is using the current labeled dataset. However, there is still a need to detect all the different kinds of concept drifts defined earlier, the sudden, recurring, gradual, and incremental drift [6]. Another method to handle drifting streaming data is proposed by Zliobaite et al. [54] where three requirements are given: labeling should obey the budget, probability for labeling should not be zero, and the labeled data and unlabeled data should not be equal with regards to its probability. They propose a method where a split strategy is used. A data stream is split into two different streams, one labeled with the randomization strategy and the other labeled with the uncertainty strategy. Only the random data stream is used for change detection; however, both are used to train the model. This proposed strategy checks all the requirements they have. However, this is a very basic method, and more advanced methods for detecting drifts and changes should be developed. Another remaining challenge is that it is still difficult to see when a model has reached convergence, so stopping criteria are difficult to gather. As such, it may never stop or stop at the wrong moment. The model might lock itself into a wrong hypothesis without noticing, the labeling budget might still not be optimal over time, and boundaries for errors and label requests are difficult to set [28]. Because of these challenges, this thesis wants to see if there is a way to detect unsupervised changes, which actively forget labels when they are outdated and keep on exploring whenever the amount of labeled data is needed for the model to train.

2.3.2 Active Forgetting

Pham et al. [37] describe forgetting outdated information, where their models forget information when it becomes obsolete. They use a sliding window to see which instances are still relevant. Forgetting techniques are already widely used in data stream mining [21]; however, in the context of Active Learning, it is used to a much lesser extent. Lughofer and Angelov describe one method in the context of fuzzy systems [33] where they propose a gradual forgetting mechanism that *"forgets"* instances from the old model when drifting the new model in the case of a sudden drift or a gradual drift. This method works by gradually forgetting instances depending on the drift's behavior. It is proposed to have a fixed parameter for how strong the drift is anticipated to be, as well as a forgetting factor depending on the intensity of the drift. Another method described by Benkert et al. [3] is in the context of neural networks, where they claim that a forgotten instance is a correctly classified instance at time t and misclassified at time t'. Their method is built on sampling those forgotten instances since they are the most informative. However, when it is misclassified at time t', it may not be worth putting the labeling budget towards the model. Because it is possible that the data is outdated and, as such, irrelevant to the current model. However, it is also a possibility that a drift occurred, and a labeling budget should be put towards it. In the paper by Wang et al. [50], they claim that it is worth forgetting the original data if task A and task B are interfering with one another. This can also be used in the context of Active Learning. If in the case of having an instance A and an instance B over time having the same label, but at time t' instance A changes its label but instance B is not changing its label. The challenge is that they are still closely related but can now interfere with each other. A possibility is that one of those instances can be causing noise since it became outdated. One can also forget instance B to not let it interfere with the changed instance A anymore. The challenge remains since the model forgets labels, and getting new labels or instances is difficult. One of the possibilities is that a model should keep exploring to gather new instances.

2.3.3 Exploration

Since a model hypothetically can forget labels, it is also important to keep on exploring for new instances but also for new classes and clusters. One proposed method to do so is to perform active joint exploration-exploitation in Active Learning proposed by Loy et al. [31]. Its goal is to discover unknown and rare classes independent from any experience, which is defined as the exploration part. The other part, the exploitation part, is learning the concept boundary by searching for instances with very low confidence regarding their class. One assumption made is that the model should be able to handle class imbalances. The goal is to minimize the human labeling budget and learn a model for some classes where not all classes are known yet. The model iteratively receives an instance from the data stream and then draws two random hypotheses for forming a committee. The posterior is then computed for each hypothesis, and when the two committee members disagree on its posterior, the label is queried, which, as a result, is used to refine the model. They discover new potential classes using Pitman-Yor Processes, which result in the classes that already have many instances getting assigned more. However, by using PYP, the processes distribute some of the large classes to potential new classes. The main outcome of these processes is that the more classes are being observed, the more data is added to new classes. Another view on exploration is by Dimitriadou et al. [18]. They propose AIDE, which is an Active Learning approach for data exploration. The data exploration part is done by a user in iterations, where in each iteration, the user is giving feedback on whether the current sample is relevant or irrelevant. The drawback of this method is that a large portion of the labeling budget is needed. The last method is by Wasserman et al. [51]. They propose to introduce Reinforcement Learning concepts to the Active Learning strategy. More precisely, it rewards the system's query behavior based on how useful the querying is. Caciarelli et al. [7] proposed a method to keep exploring unseen regions while also not being influenced by outliers. They used a D-optimal algorithm to set a scope for the exploratory area and a robust estimator to select representative data for querying when outliers are present. This is a good method to detect whether or not there are unseen areas and a way to keep exploring the instance space, which is also protected against outliers.

3. Method

A data stream consists of instances that are arriving over time. In an ideal scenario, all these instances have a label, and their distribution is known. This means the model knows whether a change is happening in the distribution of these labels. When the distribution changes, the model also needs to change. Otherwise, the accuracy can drop and instances get the wrong labels. However, a limited number of labels are usually known in the real world. Also, changes in the feature distribution may be detected with a delay or not at all. The method employed for this thesis is divided into four sections. First, the unsupervised change detection is discussed in Section 3.1. Section 3.2 elaborates on exploring new instances and labels. Section 3.3 discusses the performance classification and its assessment. Lastly, Section 3.4 shows the integration of the three different components. All three components will be combined into one algorithm, IDEAL.

3.1 Detection of Changes (Unsupervised)

This section discusses the unsupervised detection of changes. Firstly, unsupervised change detection in data streams will be elaborated on why change detection is necessary. Secondly, it illustrates the risks of having only unsupervised change detection and provides reasoning why this would not be beneficial. Lastly, several methods a change detection method will be proposed for further experiments.

Change detection is the process of identifying differences in the state of an object or phenomenon by observing it at different times [44]. In this thesis, the focus is on unsupervised change detection, defined as the process of identifying differences in the distribution without any labeled data.

This master's thesis uses a technique to calculate change scores based on Kernel Density Estimation (KDE). KDE is used in statistics to construct an estimate based on the observed data [36, Davis et al.]. This estimate can be calculated as a probability density estimation used to estimate the data population. In KDE, kernels are used as weights for the probability density estimation.

The Change Detector of IDEAL works in the following way: It considers one candidate c and a window w of L previous instances $(x_t, x_{t-1}, x_{t-2}, ..., x_{t-L})$. I set the default value for this window w 100, and the window always has to be positive and even. The Change Detector splits up w into two equal-sized sub-windows, w_1 and w_2. For each of the two sub-windows, its respective density is calculated. After the density has been calculated, the log-likelihood of the candidate for w_1 and w_2 is calculated. This candidate score for both is then increased exponentially by itself such that it is always a positive number, which results in the values ws_1 and ws_2. The change score is calculated as follows:

$$\frac{(ws2 - ws1)}{ws1}$$

It is an unsupervised Change Detector since it does not examine the data's (previous) labels but solely examines the densities of the time points in the data stream. This component does not decide if something is relevant enough to be queried. This is handled by the integration of the other components to be introduced. For the experiments, a bandwidth of .75, the default value for window size 100, and a Gaussian kernel are used for the KDE. The pseudo-code for this component can be found in Algorithm 1.

Algorithm 1 Change Detector

```
    Function Calculate Change Score (w,c)
    $w1, w2 = split(w, 2)$

    if $w1 \% 2 = 0$ then
    $ws1$ = score $c$ on KDE $w1$

    if $w2 \% 2 = 0$ then
    $ws2$ = score $c$ on KDE $w2$

    Change_score = $(ws2 - ws1)/ws1$
```

3.2 Exploration of Time and Space

One component of the algorithm keeps exploring the instance space over time. When more instances are incoming, there is a risk that the sampling distribution might not be up-to-date, and as such, the changes in the distribution might be identified too late. For this component, the current sampling distribution and the observed distribution are compared.

The equation for comparing the current sampling distribution and the observed distribution is

$$\frac{D(L) + \epsilon}{D(L \cup U) + \epsilon}$$

$L \cup U$ corresponds to all the instances in the most recent window defined in the previous component. The candidate is excluded from U, which is the set of unlabeled instances, and has no possibility of being in L, which is the set of labeled instances. If the candidate was included in U, there would be higher densities at the candidate's location and the scores would be less prone to change. The average KDE of the previous component's two windows is $D(L \cup U)$, where D is density. $D(L)$ is calculated by selecting the labeled instances from the window and using those to calculate the KDE. This ensures that $D(L \cup U)$ can never be higher than $D(L)$. At most, the two densities can be equal to each other. ϵ is included to prevent the scores from dividing by zero since there is the possibility that there will be no labeled instances during the window, and as such, $D(L)$ will be zero. However, by adding a constant value ϵ, $D(L)$ will always be higher than zero. The pseudo-code for the algorithm can be found in Algorithm 2.

Algorithm 2 Explorer

```
    Function Calculate Explorer score ($ws1$,$ws2$,$c$,$\epsilon$)
    $D(L \cup U) = (ws1 + ws2 / 2)$

    Remove unlabeled instances from $w$
    $D(L)$ = score $c$ on KDE of $D(L)$

    Explorer_score = $\frac{D(L)+\epsilon}{D(L \cup U)+\epsilon}$
```

3.3 Assessment of Classification Performance

The last component of the algorithm includes a Performance Assessor. It needs to be able to assess the performance of the current classifier. In Active Learning, multiple methods exist to classify and evaluate the classification performance. Examples of these are Uncertainty Sampling or Probabilistic Active Learning (PAL).

For the last component of IDEAL the score needs to be calculated that a new instance would be informative to query. This component also needs to have code available to be implemented in the algorithm. I am going to use Probabilistic Active Learning [25, 27] for this thesis and use the utility score the algorithm calculates. PAL models the label of the candidate instance and the instance by adding information (posterior) in the neighborhood as random variables. Then, it looks at which of the two instances increases the classification performance the most and selects that instance as a result. PAL has a better or comparable classification performance than uncertainty sampling but, at the same time, a higher complexity. In a data stream, the temporal usefulness is also added to PAL, which means that over time, an assessment is made of which instances are useful to query within a sliding window and a budget. In the paper by Kottke et al. [25], there is a description of the pseudocode. Lines 5 until 8 are being used as a component to determine the utility score. The paper also mentions the Balanced Incremental Quantile Filter technique to determine the temporal usefulness over time. However, for IDEAL this part is removed. The algorithm can be found in Algorithm 3.

Algorithm 3 Performance Assessment

```
Function Calculate utility score
{determine spatial usefulness value}
$\hat{p} \longleftarrow P_C(+ \mid X_i)$; $n \longleftarrow \mathrm{KFE}_C(x_i)$
Utility_score $\longleftarrow$ pgain($\hat{p}$,$n$)
```

3.4 Integration of Components

The three different components introduced need to be integrated into a singular algorithm. However, the challenge for this singular algorithm is to identify a way the three different components are supposed to be weighted. For example, if all three components were of equal weight, then one component might have the issue that it could become obsolete at times and use resources that the other components could need. The risk of having solely a Change Detector in the algorithm is that it might not detect changes in the labels. The Change Detector will benefit from seeing a change in the prior; however, since it is unsupervised, it will not look at the labels themselves but more at the density of the instances. The risk of having solely the Explorer part of the algorithm is that the active learner might run out of instances at some point and query instances too soon. The benefit is that insights into the current sampling distribution versus the currently observed distribution are obtained and, as such, a change when comparing the two distributions can be detected. The last component of the algorithm is the Performance Assessor, where if this is solely executed, the risk is that changes happening (in the distribution) might not be detected. The benefit of the Performance Assessor is that one knows what the performance of a classifier is and how well or badly it is performing. As a result, one can investigate and adjust the parameters in such a way that the algorithm performs better again.

Algorithm 4 Ranking

```
Function Query by ranking
{determine the weights}
{determine the ranks for each of instances from the last window and candidate}
{determine the average rank for the candidate}
{determine the threshold using average rank and budget}
Acquire instance = $ranks_{avg} >= threshold$
```

The three components produce one score with different values and scales. However, they all have the property that a higher score means that the score is more meaningful for that component. As such, a rank can be given to each score, and their score can be determined by a ranking. Then a weighted average is calculated amongst all three rankings and that final rank is used to determine if it is above a certain threshold or not. The threshold is controlled by the budget, which is a percentile value. The more budget you allocate, the lower the overall rank needs to be for an instance to be queried. So if the budget would only query the top 10% of the ranks (*budget* = 0.1), then in a dataset with 1000 instances, 100 will be queried. One can also decide to shift the weights of each of the ranks which puts more weight on one component and less on the other components. The sum of the weights should always be 1 (or correspond to 100%). In the formula below, the $\frac{1}{3}$ can be changed to any value to put more or less weight as long as the sum of the weights equals 1. Also, with an increasing number of criteria over which the ranks are averaging, the distribution of the average rank will approach a normal distribution.

$$Weights = \{Change, Explore, Utility\} = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\} = 1$$

4. Experimental Evaluation

In this chapter, the experiments are explained and evaluated. The results are reported from three different real-life datasets and one synthetic dataset.

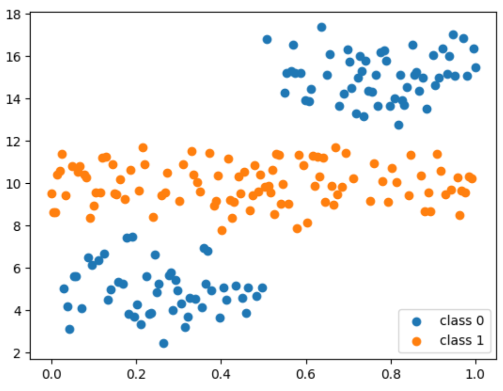
4.1 Experimental Design

This section discusses the different datasets that are used for the experimental evaluation. It discusses the origins of the datasets, the distribution, the goal, and the properties the datasets have. This section also discusses how the evaluation of the experiment is executed. Which Active Learning strategies are used to compare IDEAL to and how the performance is being measured.

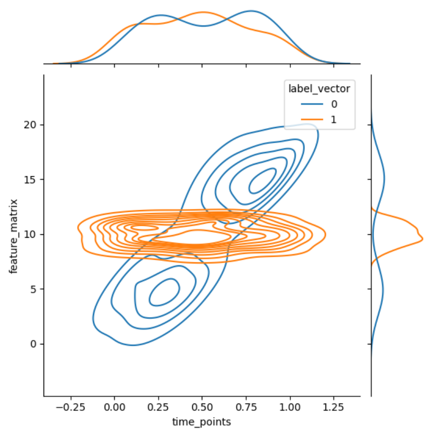
4.1.1 Datasets

The synthetic dataset used for this experiment consists of a data generator that generates two different classes and multiple instances over time. One of the two classes consistently remains around the same location in the instance space. The other class shifts its position over time. One example of this can be found in Figure 4.1a. The dataset has 210 instances over time, where class 0 is multimodally distributed over the temporal and feature space. Class 1, however, was more concentrated in one area in the feature space but had instances coming in over but with one peak instead of class 0 having two peaks 4.1b. For the experiments comparing all the components and the different algorithms, the classifier received 10 instances on which it could train itself. The sliding window that limited the training data was 100, which means that every time the classifier trained itself, the classifier trained on the last 100 instances.

Three real-world benchmark datasets are used to evaluate IDEAL and other algorithms. *Electricity, Airlines* and *Creditcard.* Since IDEAL currently cannot process textual data and multi-features, I decided to use Principal Component Analaysis (PCA) to reduce the multitude of features into one feature by capturing the largest amount of variance. It is important to note that this also bears the risk that it is not as easy to predict the classes as if all the features were



(a) Synthetic dataset instances



(b) Synthetic dataset distribution

Figure 4.1: Synthetic dataset instances and distribution

available. A section will be dedicated to this challenge in the chapter future research. All datasets are taken from the OpenML Framework [49], with ids for: *Electricity* 151, *Airlines* 1169 and *Creditcard* 1597. Since there is only one method to extract the data, it ensures that all the real-world datasets are exported in a uniform manner, and the uniform manner risks no changes or losses in the data.

The *Electricity* dataset was first published in the papers by Harries [23] and Gama [19]. The goal of the Electricity dataset is to predict whether the prices of electricity will go 'up' or 'down'. The data was collected between the 7th of May 1996 and the 5th of December 1998 from the states New South Wales and Victoria in Australia. The features used to predict are the date, day of the week, period, the electricity price and demand in New South Wales, the electricity price and demand in Victoria, and the electricity transfer between the two states. The PCA uses the following features for the experiment: The electricity price and demand from New South Wales and Victoria and the transfer. This result in 5 numerical features used in PCA to reduce it to one feature; *nswprice, nswdemand, vicprice, vicdemand and transfer*, 2 classes; *up, down*, and 45,312 instances. The number of instances that the classifier received to train on was 10. The sliding window that the classifier used to train was 300 instances.

The *Airlines* dataset is inspired by Elena Ikonomovska, who used the dataset from the Data Expo 2009[15]. The original dataset consists of nearly 120 million records. The goal of the dataset is to predict whether a flight will be delayed or not. The features used to predict this are the airline, flight number, the origin airport and the destination airport, the day of the week, the time, and the length of the flight. For the experiment, I use the following features for the PCA: The time of the instance and the flight length. The airline, origin, destination airport, and day of the week are string features. The flight number is omitted since it is not meaningful for the instance space and would result in outliers that do not hold much meaning. This results in two numerical features used in PCA to reduce it to one feature: *time, length*, 2 classes; yes, no, and 539,383 instances. The number of instances that the classifier received to train on was 100. The sliding window that the classifier used to train was 1000 instances.

The third dataset used is *Creditcard* [12], which is intended to be used to predict if a transaction is fraudulent. This dataset is highly unbalanced, where the non-fraudulent transactions, which account for 99.828%, are far greater than the fraudulent transactions, which account for 0.172%. This means it is very challenging to capture the transactions labeled as fraud. The dataset contains transactions made by European cardholders in September 2013. The features used to predict whether or not a transaction is fraudulent have been transformed using PCA and masked due to confidentiality. This results in 29 numerical features used in PCA to reduce it to one feature: *V1, V2.., V28, time*, 2 classes; fraud, no fraud, and 284,807 instances. The number of instances that the classifier received to train on was 100. The sliding window that the classifier used to train was 1000 instances.

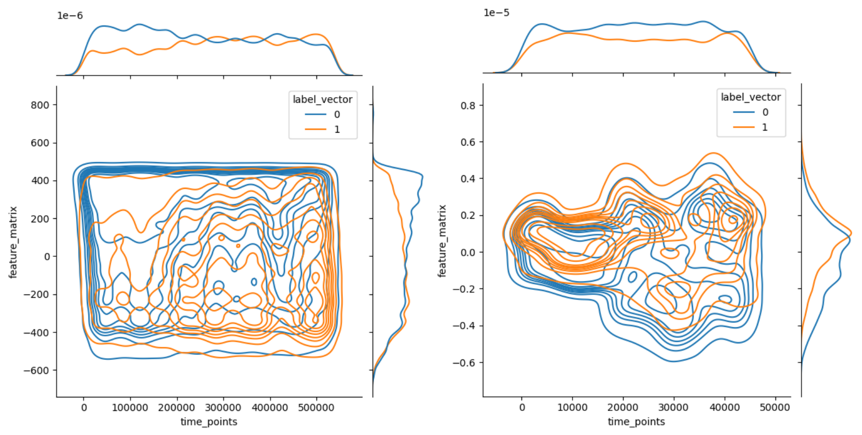
Dataset	n Instances	n Classes
Synthetic	210	2
Airlines	539383	2
Creditcard	284807	2
Electricity	45312	2

Table 4.1: Dataset characteristics

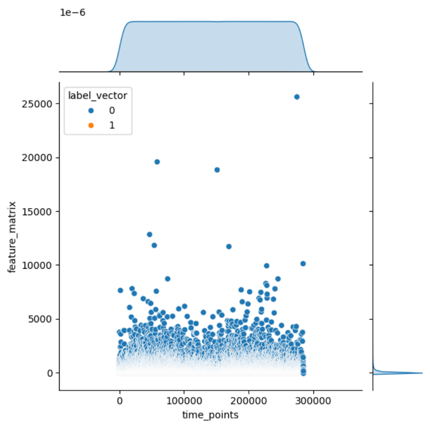
The real-world benchmark datasets are visualized in Figure 4.2. In the distribution of the *Airlines* 4.2a dataset, the blue class (no delay) has a higher distribution than the orange class (delay). The biggest challenge with this dataset is that there are clusters inside the data. It needs to be ensured that the active learner is able to query them. In the distribution of the *Electricity* dataset 4.2b, the blue class (prices go down) also dominates the orange class (prices go up) over time. However, when looking at the feature space, the orange classes tend to be higher placed than the blue classes. The challenge for the active learner is to query those instances that give the most information to it. In the distribution of the *Creditcard* dataset 4.2c, it is very clear that the blue class (no fraud) is dominating the orange class (fraud). There is a small amount of 0.172% that is classified as fraud. The challenge for the active learner will be to see if it can detect fraudulent cases, and in this scenario, it is desirable if the active learner is too sensitive and asks for more labels at certain points.

4.1.2 Evaluation Techniques

I will compare four other Active Learning strategies to IDEAL for the experiment. The first is *Random Sampling* as a baseline. This technique randomly requests a label. *Fixed Uncertainty* is the Active Learning strategy that queries the instances about which the classifier is the least confident. The data stream queries the instances with an uncertainty below a certain threshold [54]. Another uncertainty-based method is the *Variable Uncertainty* Active Learning strategy. The biggest challenge with a fixed threshold is that if the parameters



(a) Distribution of dataset *airlines* (b) Distribution of dataset *electricity*



(c) Distribution of dataset *creditcard*

Figure 4.2: Distributions of the real-world benchmark datasets

are set incorrectly due to the threshold being too high or too low, the active learner will stop learning. Any changes that occur after the active learner has stopped learning will not be detected, and the accuracy will decrease. The Variable Uncertainty strategy tries to query the least certain instances within a time window. Since it is a dynamic threshold, the number of instances the active learner queries remains the same. The fourth Active Learning strategy used is *Probabilistic Active Learning*, which is also used as one of the components in IDEAL. Instead of asking for which instance the classifier is most uncertain, PAL calculates how much the probabilistic gain is if that query is queried. In a data stream, it would then calculate that position in a quantile ranking system [25].

To evaluate the algorithms and the datasets, I first used prequential evaluation to set up the experiment. This method was first proposed by Gama et al. [20] and is currently the most common evaluation method for data streams. It is an interleaved test-then-train method where the candidate is first tested before it is used as a training instance. As a result, the accuracy is updated after each instance, and the model is tested on instances that it has not encountered before. In a data stream, this is common since the model never knows which instance will come next. This is the opposite of a pool-based setting, where the whole dataset is available simultaneously. I calculate the accuracy to evaluate how an Active Learning strategy is performing. The accuracy is calculated by retrieving the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). True Positives and Negatives are the instances where the classifier correctly classified the positive or negative class. False Positives and Negatives are the instances where the classifier incorrectly classified them as the positive or negative class. By applying the formula below, the accuracy can be calculated. Together with the accuracy, the confusion matrices are plotted for the synthetic datasets for IDEAL. A confusion matrix shows the number of TP, TN, FP, and FN in the prediction of the classifier.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

I use several graphs to further showcase the performance of the Active Learning algorithms. The first graph is the performance over time, which plots the accuracy of each algorithm over time. The time is on the X-axis, and the accuracy score is on the Y-axis. Another graph is the Pareto Front, which is an optimization function aiming to maximize or minimize something. I want to determine which active learner is the most accurate for the experiment. The Pareto Front in this experiment is a maximization function on accuracy. The graph does not show the intervals. However, every graph plots the best-performing algorithm at some interval. The last graph I use is the density graph for IDEAL. This density graph shows at which points IDEAL queries instances and which labels the instances received, and if the active learner queries a lot in an area, the background is darker, indicating a dense area. The time is plotted on the X-axis of this graph, and the feature space is plotted on the Y-axis.

4.2 Integration of Components

This chapter will discuss the integration of the components and evaluate how it performs with different weightings and how it performs against other algorithms. The chapter delves into two different types of datasets: synthetic and real-world benchmark datasets.

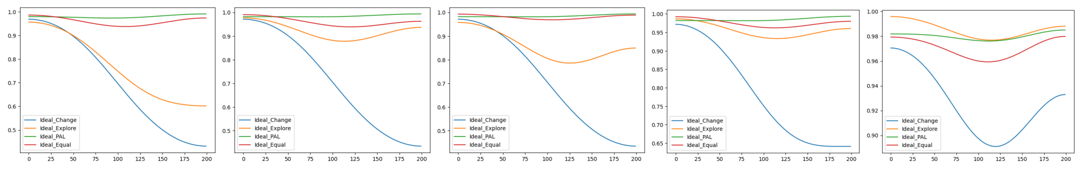
4.2.1 Synthetic Dataset

4.2.1.1 IDEAL

To evaluate the IDEAL algorithm on the synthetic dataset, I look at the different accuracies per different weights and per different budgets. The weights can be changed such that the sum of all the different components is equal to 1. The first run only turns on the Change Detector, and the second run only turns on the Explorer component. The third run only turns on the Performance Assessor component, and the last run has the three components equally weighted. I also ran the experiment per budget, which had a value between 0 and 1. With a budget of 0, no candidate is queried, and 1, every candidate is queried. When a budget of 0.1 is selected and 100 instances are available, the active learner should query about 10% of the instances. I run each weighting and budget 100 times; its average accuracy is shown in Table 4.2. For the experiment, the Performance Assessor is the best-performing component for all the budgets. The Change Detector component performs better with a higher budget. However, it still does not reach a high accuracy.

Weights \ Budget	0.1	0.2	0.3	0.4	0.5
1|0|0	0.701	0.707	0.708	0.782	0.926
0|1|0	0.767	0.920	0.946	0.955	0.977
0|0|1	**0.980**	**0.983**	**0.985**	**0.983**	**0.981**
$\frac{1}{3}\|\frac{1}{3}\|\frac{1}{3}$	0.829	0.958	0.975	0.977	0.976

Table 4.2: Comparison IDEAL accuracy per budget and component weighting synthetic dataset

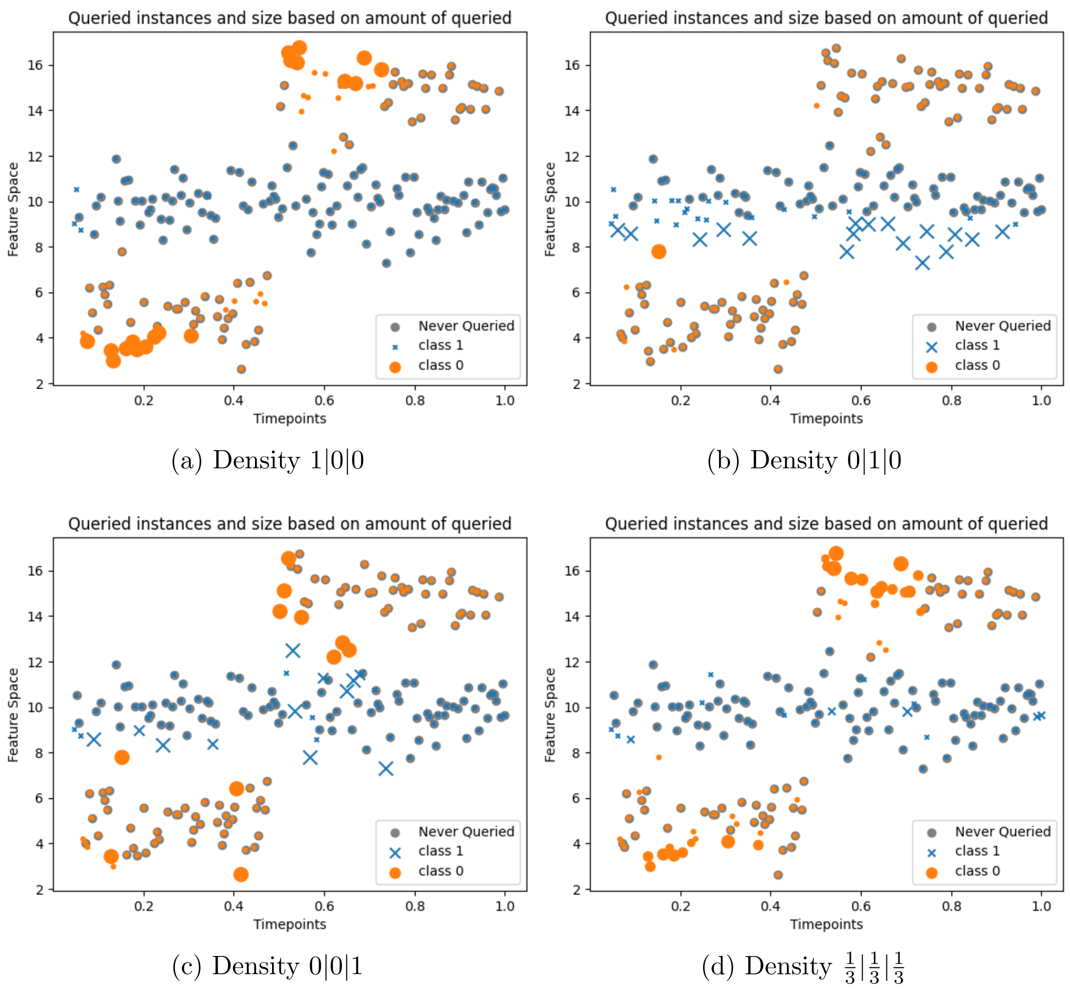


(a) Performance 0.1 budget (b) Performance 0.2 budget (c) Performance 0.3 budget (d) Performance 0.4 budget (e) Performance 0.5 budget

Figure 4.3: Performance over time IDEAL with different settings synthetic dataset
Blue: 1|0|0, Orange: 0|1|0, Green: 0|0|1, Red: Equal

The Explorer component of IDEAL does not perform as well with a small budget. However, as the budget increases, the component performs better in terms of accuracy. With a budget of 0.5, the change component is able to reach a higher accuracy than with a smaller budget. The Performance Assessor component is the best-performing component in the experiment using the synthetic dataset. With a small budget, the accuracy of the Performance Assessor is close to 100%. However, it is beginning to fall off slightly with a budget bigger than 0.3. When all three of the components are equally weighted, they perform better with a larger budget. In Figure 4.3, one run out of the 100 for each budget is plotted to showcase each component's performance over time. The Change Detector component, represented by the blue line, is not performing very well overall. The Explorer component, the orange line, is not performing well with the smaller budgets. Both the Change Detector and Explorer components have difficulty at the point where the locations of the features shift. The performance of the Explorer component is improving and is able to surpass when it had a budget of 0.5. The Performance Assessor, the green line, is performing well overall. The equally weighted IDEAL is performing well overall. However, it also starts to decrease when a bigger budget is given to the active learner.

I visualized the labeled and unlabeled instances in Figure 4.4 over 100 runs. If the instance is represented by a big circle or cross, it is queried often in the 100



(a) Density 1|0|0

(b) Density 0|1|0

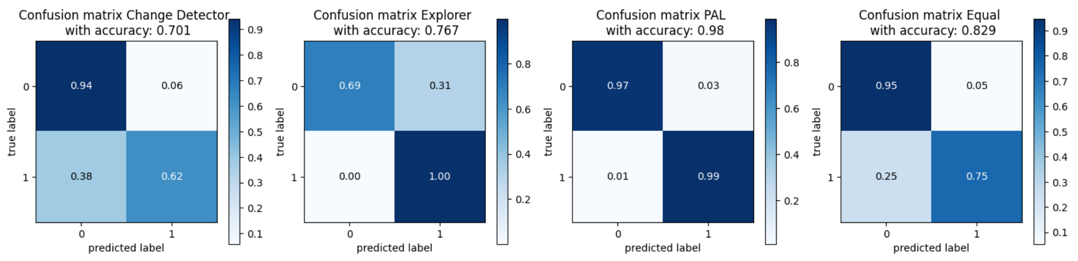
(c) Density 0|0|1

(d) Density $\frac{1}{3}|\frac{1}{3}|\frac{1}{3}$

Figure 4.4: Density of the queried instances at 0.1 budget and 100 runs synthetic dataset

runs; if the instance is small, then it is not queried often. If the instance also has a grey background, it is not queried. The Change Detector queries instances often at the beginning of the stream from the density of class 0 and queries around the flip from the density of class 0. The Explorer component prefers to query the instances from class 1, not the lower or higher feature locations. The Performance Assessor component queries all around the feature space with no real preference for the classes. However, this component prefers certain instances in the feature space indicated by the size of the circles and crosses. The equally weighted IDEAL algorithm prefers to query the same instances as the change component. However, the algorithm does not query it as much and queries more around the same areas.

Lastly, looking at the confusion matrices of the four different weightings at 0.1 budget over 100 runs visualized in Figure 4.5, the Performance Assessor component reaches high True Positive and True Negative rates. The Explorer component reaches the best True Negative rate, predicting class 1 (the negative class) correctly every time. This component has more difficulty predicting class 0 correctly and looking at the density of the queried instances; one can see that the active learner does not query as many instances from that class as it does for class 1.



(a) Confusion matrix 1|0|0 (b) Confusion matrix 0|1|0 (c) Confusion matrix 0|0|1 (d) Confusion matrix $\frac{1}{3}|\frac{1}{3}|\frac{1}{3}$

Figure 4.5: Confusion matrices of 4 different weightings at 0.1 budget synthetic dataset

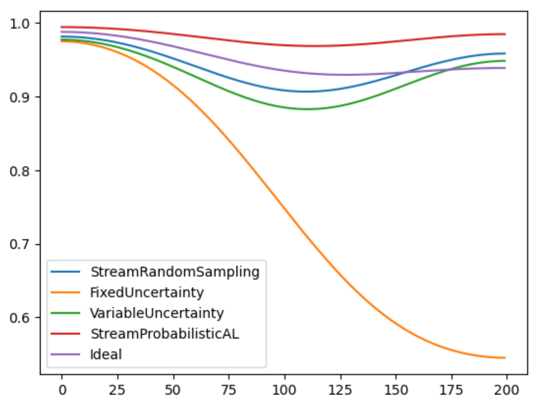
4.2.1.2 Comparison

Budget / Algorithm	0.1	0.2	0.3	0.4	0.5
Random	0.934	0.964	0.972	0.978	0.981
Fixed	0.755	0.730	0.730	0.730	0.730
Variable	0.946	0.967	0.980	0.987	0.986
PAL	**0.980**	**0.990**	**0.990**	**0.990**	**0.990**
IDEAL	0.956	0.980	0.988	0.985	**0.990**

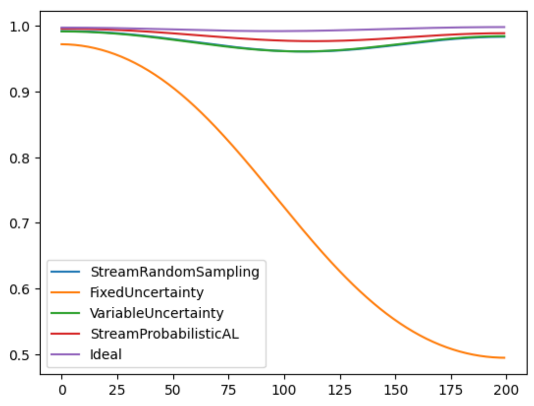
Table 4.3: Comparison accuracy of algorithms per budget synthetic dataset

Next to analyzing the individual IDEAL components on the synthetic dataset, I compare different algorithms and their accuracy over 100 runs. I use the Random query strategy as a baseline. In Table 4.3, the different accuracies over 100 runs are summarised in a table. The Fixed Variable strategy is performing the worst out of the 5 Active Learning algorithms. With 0.1, 0.2, 0.3, and 0.4 budgets, PAL is performing the best in terms of accuracy with Variable Certainty, and IDEAL is also exhibiting good accuracy. However, when looking at the 0.5 budget for the synthetic dataset, IDEAL performs equally as well as PAL. Overall, IDEAL does not perform badly compared to the other query strategies. One of the runs is used as an example in Figure 4.6. At 0.1 budget, IDEAL decreases in its accuracy over time and stabilizes at the end. Fixed Uncertainty has good accuracy at the beginning of the stream; however, it decreases over time. Variable Uncertainty and Random have a decrease in their accuracies around the concept drift of the feature space location. PAL remains nearly constant in terms of their accuracy. When looking at the 0.5 budget, Fixed Uncertainty has the same problem as the 0.1 budget experiment. Random, Variable Uncertainty, and PAL all perform equally. However, IDEAL is constant and achieves perfect accuracy over time.

In the confusion matrices of the 5 algorithms at 0.1 budget Figure 4.7, Random, Variable Uncertainty, PAL, and IDEAL have a high number of True Negatives.



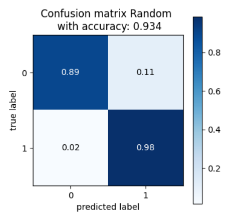
(a) 0.1 Budget



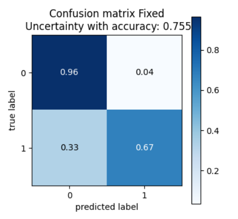
(b) 0.5 Budget

Figure 4.6: Performance over time algorithms at 0.1 and 0.5 budget synthetic dataset

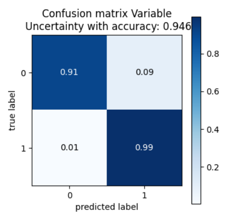
Variable Uncertainty has a number of 0.99 TN, and the other 3 strategies only score 0.01 lower. The difference and the challenge for the synthetic dataset lies in the True Positives. This is the class that flips in the feature space location from below the negative class to above the negative class. Interestingly, Fixed Uncertainty has a 0.96 True Positive value there. However, PAL scores higher, with a score of 0.98. At the confusion matrices at 0.5 budgets Figure 4.8, all 5 algorithms have a perfect amount of True Negatives. Interestingly, Fixed Uncertainty now has a low True Positive rate and predicts False Positives more often. PAL and IDEAL both have the same score in the confusion matrices. Variable Uncertainty only scores 0.01 less than PAL and IDEAL on True Positives, and Random scores 0.02 less.



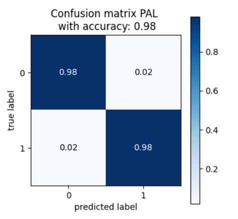
(a) Confusion matrix Random



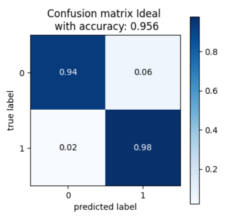
(b) Confusion matrix Fixed Uncertainty



(c) Confusion matrix Variable Uncertainty



(d) Confusion matrix PAL



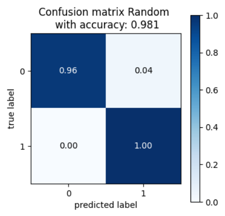
(e) Confusion matrix IDEAL

Figure 4.7: Confusion matrices of 5 algorithms at 0.1 budget synthetic dataset

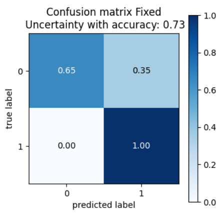
4.2.2 Real-World Benchmark Datasets

4.2.2.1 Electricity

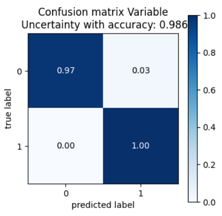
The first real-world benchmark dataset is the *Electricity* dataset. The objective is to predict if electricity prices will go up or go down. I run this experiment once for the budgets 0.1, 0.2, 0.3, 0.4, and 0.5. At 0.1 budget, PAL exhibits the highest accuracy, whereas IDEAL and Variable Uncertainty both have an



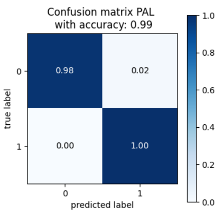
(a) Confusion matrix Random



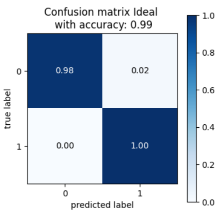
(b) Confusion matrix Fixed Uncertainty



(c) Confusion matrix Variable Uncertainty



(d) Confusion matrix PAL



(e) Confusion matrix IDEAL

Figure 4.8: Confusion matrices of 5 algorithms at 0.5 budget synthetic dataset

accuracy of 0.614, which is 0.017 lower than the accuracy of PAL. PAL has the highest accuracy at 0.2, 0.3, and 0.4 budget. The highest value is at 0.2 budget with an accuracy of 0.654. At 0.5 budget, the Variable Uncertainty has the highest accuracy with a score of 0.647. The IDEAL algorithm is the third best performing algorithm at all budgets, except for a budget of 0.1, where it is tied with Variable Uncertainty. In Table 4.4 and graph Figure 4.9, I show the accuracies per budget. Interestingly, for most of the algorithms, the accuracy always increases with a higher budget. However, for the PAL algorithm, it decreases at the 0.3 budget and keeps decreasing with higher budgets. IDEAL also decreases in accuracy when the budget is increased from 0.4 to 0.5. In With 4.10a plotting all the algorithms at the same time and 4.10b having the best performing algorithm colored at an interval of every 5,000 instances. At 0.1 budget, PAL has the highest accuracy at all interval points. IDEAL performs slightly worse than PAL but is not the worst in accuracy overall.

Budget / Algorithm	0.1	0.2	0.3	0.4	0.5
Random	0.606	0.615	0.620	0.621	0.623
Fixed	0.523	0.523	0.523	0.527	0.526
Variable	0.614	0.631	0.634	0.633	**0.647**
PAL	**0.631**	**0.654**	**0.643**	**0.638**	0.632
IDEAL	0.614	0.623	0.629	0.632	0.628

Table 4.4: Comparison accuracy of algorithms per budget *electricity* dataset

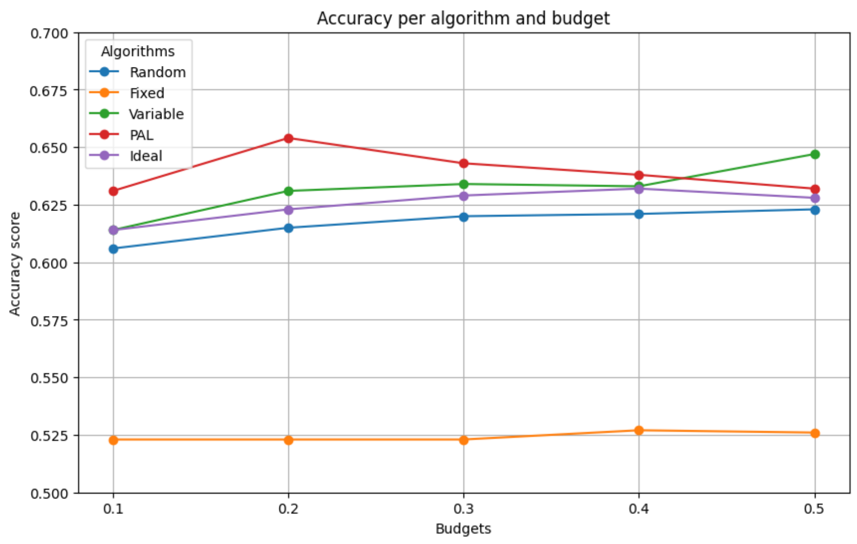
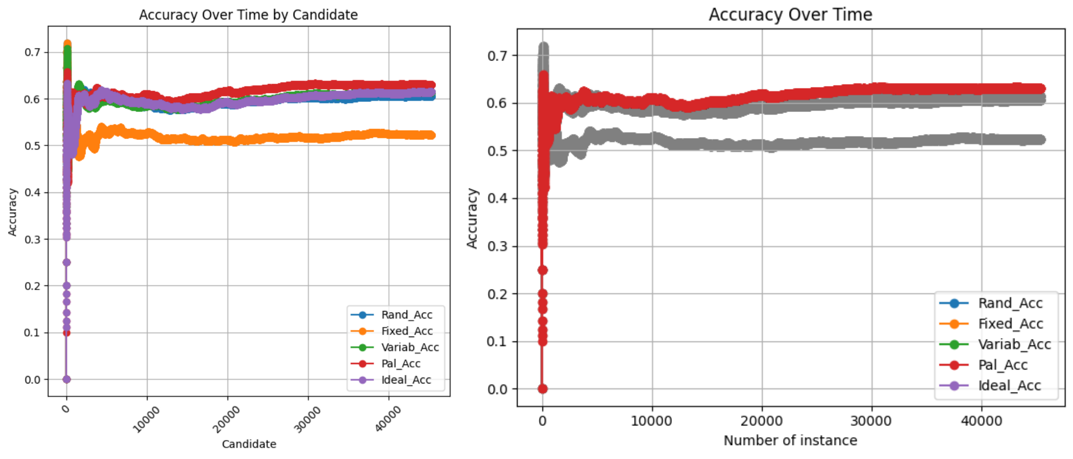


Figure 4.9: Accuracy per budget for each algorithm *electricity* dataset

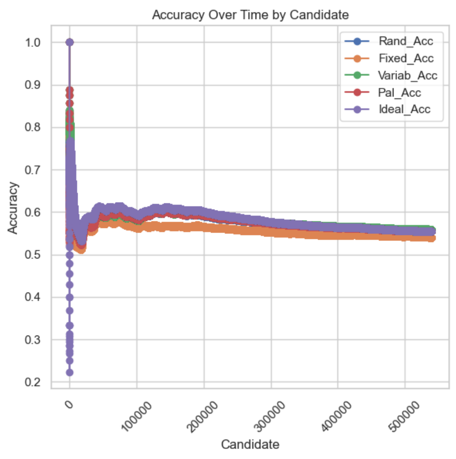


(a) 0.1 Budget accuracy (b) Best performing algorithm

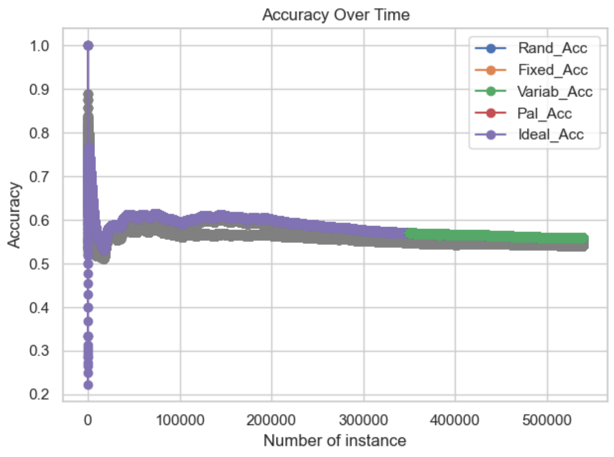
Figure 4.10: Performance over time algorithms at 0.1 budget *electricity* dataset

4.2.2.2 Airlines

The *Airlines* dataset is also tested with all five algorithms. However, due to time constraints and the dataset size, performing the experiments for all 5 budgets was not feasible. For the *Airlines* dataset, the algorithm with the highest overall accuracy is the Variable Uncertainty with a score of 0.559. However, Random, PAL, and IDEAL reach similar accuracy scores. Random has an accuracy of 0.557, and PAL and IDEAL have an accuracy of 0.554. All of the algorithms perform better at predicting the negative class as opposed to the positive class. In 4.11a, the performance over time is plotted, and in 4.11b, the algorithm with the highest accuracy over time in intervals is plotted. The interval is every 50,000 instances, and for the first 350,000, IDEAL has the highest accuracy. At the end of the data stream, the Variable Uncertainty has a better accuracy.

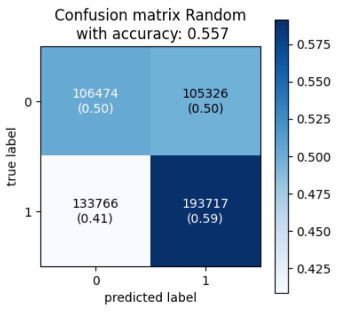


(a) 0.1 Budget accuracy

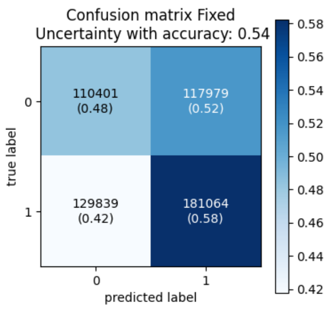


(b) Best performing algorithm

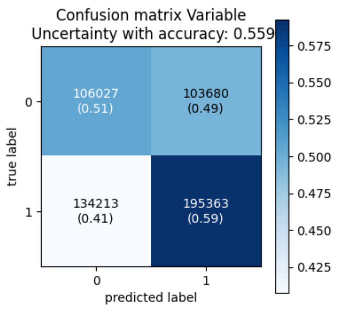
Figure 4.11: Performance over time algorithms at 0.1 budget *airlines* dataset



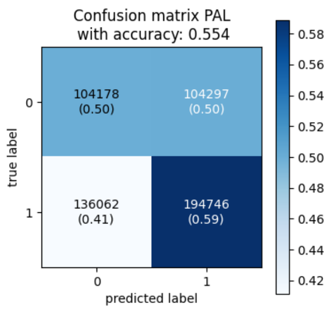
(a) Confusion matrix Random



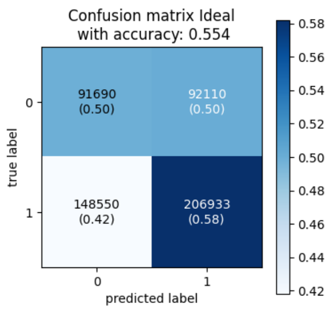
(b) Confusion matrix Fixed Uncertainty



(c) Confusion matrix Variable Uncertainty



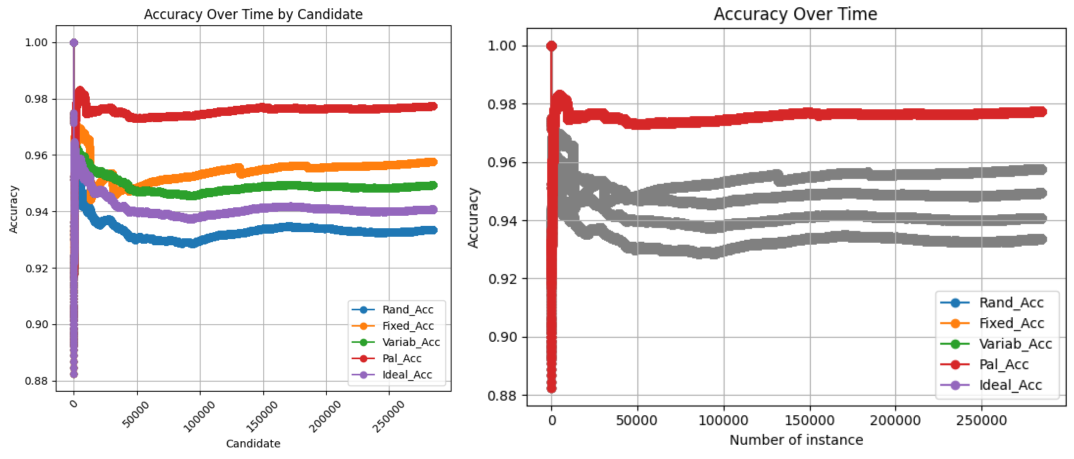
(d) Confusion matrix PAL



(e) Confusion matrix IDEAL

Figure 4.12: Confusion matrices of 5 algorithms at 0.1 budget *airlines* dataset

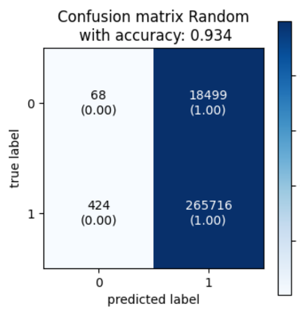
4.2.2.3 Creditcard



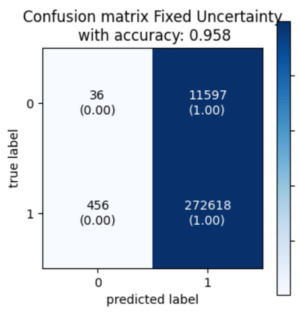
(a) 0.1 Budget accuracy (b) Best performing algorithm

Figure 4.13: Performance over time algorithms at 0.1 budget *creditcard* dataset

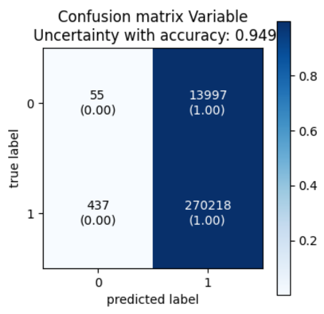
The classifiers performed on the *Creditcard* dataset have a high accuracy score because of the imbalance in the class labels. For this dataset, it is more interesting to look at the confusion matrices rather than the performance over time. Figure 4.13, plots the accuracies over time where PAL has the highest accuracy. In Figure 4.14, PAL has fewer False Positives than the other four algorithms but has slightly more False Negatives. IDEAL has a lot more False Positives but only has a small amount of False Negatives. For this dataset, it would be desirable to have an algorithm to have high True Positive and True Negative. However, in the case of fraud, if the algorithm would pass an individual instance as non-fraudulent while it was fraudulent, this is problematic and that transaction could potentially have a huge negative impact. Thus, while the difference is not that big, it is still noteworthy that IDEAL has fewer False Negatives than Fixed Uncertainty, Variable Uncertainty, and PAL.



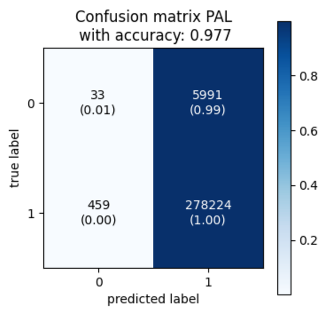
(a) Confusion matrix Random



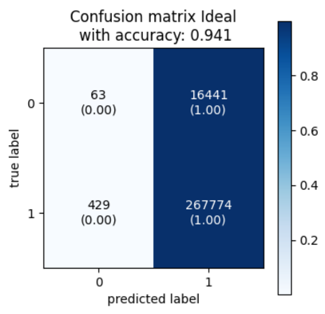
(b) Confusion matrix Fixed Uncertainty



(c) Confusion matrix Variable Uncertainty



(d) Confusion matrix PAL



(e) Confusion matrix IDEAL

Figure 4.14: Confusion matrices of 5 algorithms at 0.1 budget *creditcard* dataset

4.2.3 Discussion

While the conclusion highlights the possible answers to the main research question and sub-research questions, it is also evident that IDEAL under the settings explored in the experiments is not the best-performing algorithm. The Performance Assessor component is the best-performing component in IDEAL according to the experimental results. However, I have tested four different weightings for the experiments, and there might be an optimal weighting situation in which the weightings of the respective components are not equal. This may depend on the components that are used in IDEAL and may depend on the budget. Furthermore, Probabilistic Active Learning has shown better results compared to the equally weighted IDEAL algorithm results. Because all the components are weighted equally, the algorithm might not have been able to detect changes, explore the feature space or assess its performance based on those weightings. In the synthetic dataset, the Change Detector and Explorer components showed a decrease in accuracy at some points. Further research would be necessary to explore why the performance is dropping and take measures for those drops. Also, the PCA might influence the instances that IDEAL want to query since some of the features that are important could have been or made less impactful.
In this experiment, I compared IDEAL to four other Active Learning strate-

gies, but there are more available, such as Active Learning strategies and data streams. There are also other methods, such as Reinforcement Learning, that are worth exploring. While this research only focused on Active Learning itself, the accuracy might improve when combining IDEAL to other Active Learning strategies or Machine Learning methods not explorerd in this thesis. As such, more research is needed into an optimal active learner.

4.2.4 Limitations

In this section, I will discuss some of the limitations. For the experiments, I have used PCA to ensure that every data point is represented by a singular numerical feature. Currently, multiple features can be used as input in IDEAL. However, the inner workings are unclear and not tested. Having only a PCA feature for IDEAL but using multiple features for the other algorithms would not make for a fair comparison. However, when transforming multiple features into one singular one, you risk losing the informational value of the other variables. As such, some features which could have been important were at the risk of being manually removed. Likewise, some features that could not have been as important were at the risk of being made more important. The removal or increased importance for features could have introduced some bias towards certain locations in the feature space. However, the uniform treatment of the feature input for all data streams ensures comparability and a very interesting future research direction. Furthermore, for the *Airlines* and *Creditcard* datasets, experiments were conducted on the 0.1 budget only. This is due to time constraints since executing the algorithms on large datasets takes a lot of time. One run of the *Airlines* dataset takes 20 hours and one run of the *Creditcard* takes 8 hours on an Nvidia RTX 3070.

5. Conclusion and Future Works

This chapter will explore the conclusion of the main research question and sub-research questions and will delve into several open challenges after the experiments and conclusion for future research.

5.1 Conclusion

The research aimed to discover *is there an optimal active learner in an evolving stream-based setting*? I have tried to achieve this by proposing a new Active Learning algorithm named *IDEAL*. IDEAL is short for **I**ntegrating the **D**etection of changes, **E**xploration of instance space, and **A**ssessment of performance into **A**ctive **L**earning. This algorithm aims to detect changes, explore unexplored areas in the instance space, and assess the performance. There are still some open questions regarding how an optimal active learner exactly should operate, but the IDEAL algorithm is a good baseline. Even though IDEAL is not the best-performing algorithm, it is expandable with more components, and further research can be done on the Change Detector and Explorer.

The first sub-research question is answered by creating a Change Detector that can detect changes in the density without the need for labels. The results from the experiments indicate that the Change Detector detects a change by querying the change areas. However, performance decreases around the change point, and without a big enough budget, it does not increase.

The second sub-research question aimed to be able to explore the current area and not run out of labels. This thesis aimed to achieve that by comparing the density of the current labeled distribution to the density of the unlabeled and labeled distribution. The results show that while the Explorer can detect the difference in distributions, that component also needs a bigger budget. The Explorer has difficulties when the budget is too small to adapt to the differences in the distributions. For the second part of the sub-research question, IDEAL does not run out of labels because of the ranking system and the budget adapting to the ranking and already queried instances.

The third sub-research question focused on estimating the current classifier performance and querying an instance based on that. To achieve this, the utility score from Probabilistic Active Learning is used to assess what the probability gain would be when the instance is queried. The results show that this component performs really well and achieves the highest accuracy on the synthetic dataset. Even with a small budget, it is reaching an accuracy close to 1.

The last sub-research question aimed to integrate the three different components and evaluate which instances should be queried. By setting up a weighted ranking, IDEAL can determine which candidate has which rank in each of the components. By combining the ranks and putting that in a ranking of its own, the active learner can decide to query the candidate based on the threshold decided by the budget. The results showed that IDEAL is not the most optimal choice for most situations at this point and there is a need for additional research to see how the IDEAL algorithm can be improved.

This thesis shows that IDEAL, right now, is not the most optimal active learner overall. However, for some experiments, IDEAL is performing well, and overall, IDEAL is one of the better-performing algorithms. With additional research into the Change Detector and Explorer, IDEAL can be an optimal active learner that can detect changes, explore the feature space, and assess its performance.

5.2 Future Works

For future research, some challenges remain to investigate and improve upon. All challenges have been summarised below.

- Handle *multiple features*;
- Handle numerical and *textual* features;
- Ideal *weightings* tuning;
- *Reinforcement Learning* incorporation;
- *Components tuning*;
- *Additional components*;
- *Hyper parameter tuning*;

The experiments were executed on one feature for each candidate. However, as discussed, there is a risk of losing the value of importance for features. For future research the IDEAL algorithm has to be able to incorporate multiple features. Right now, IDEAL is already able to use more than one feature as input. However, I am unsure as to what it exactly does with the multiple features. I cannot be sure if it will pick one random value from the array of features or if the algorithm will handle multiple features correctly for each candidate. For example, the Change Detector has to be able to place each feature correctly and see from the multiple features if there has been a change. The goal for this challenge is: *Find out what the effects are of multiple features on IDEAL and* *how it affects each of the three components.*

Only the numerical features have been used for PCA and the textual features have been removed from the set of features for the experiments. However, these textual features could hold important values. For example, in the *Airlines* dataset, the airlines are a textual feature. It could be that one airline consistently has a delay, which would be important for a classifier to know to predict its class. There are methods to transform textual features into numerical features. However, because of the use of PCA, it was not feasible to do that transformation. The goal for this challenge is: *Identify how textual features can be implemented in IDEAL after IDEAL can use multiple features.*

IDEAL uses three different weightings to be able to make a ranking. The code already has the possibility to give different weights to different components. However, there are two challenges. The first challenge is what is the ideal weighting. Since every data stream has different characteristics, there is a need to adjust the weight based on the data stream and its characteristics. Also, since every component has its own goal of why it is necessary, most of the time, the component might not contribute to the active learner. For example, if there is no change happening, the value of the Change Detector over time might not mean much. This also ties into the second challenge. Is there one specific weighting setting that IDEAL needs to have or do the weightings need to adjust dynamically over time? Since an evolving data stream is continuously changing, it makes sense that, at times, different weightings have different importance to the rankings. The goal for this challenge is: *Identify if there is an ideal weighting distribution and discover if there is a possibility to adjust the weightings dynamically.*

Next to Active Learning, there is also the possibility to incorporate Reinforcement Learning, for example, to act in the phases when no change is happening. As soon as the active learner is detecting that one of the components is triggered then it shifts back to Active Learning for a period of time to train again. The goal for this challenge is: *Can the accuracy be improved when incorporating IDEAL with Reinforcement Learning?*

IDEAL has three different components with each a use-case on when each component is useful. However, there can be situations when three components can be too much and only two components would fulfill the goal. The opposite can also happen and more components are needed to be able to adjust to the data stream if a characteristic of the stream becomes important. IDEAL already supports this by its ranking system for the different components. However, the addition of components would need more research to be able to say what effect more or fewer components would have. The goal for this challenge is: *Identify the effect more or fewer components would have on IDEAL.*

The last challenge has more to do with the classifier. The experiments have been executed without additional tuning. Which can result in worse accuracy and more false Positives and Negatives. For further research, the classifier must be tuned to the data stream to make more correct classifications. The goal for this challenge is: *Tune the classifier and its parameters to make predictions more accurately.*

Bibliography

[1] Abe, N. and Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. pages 1–9. (cited on Page 15)

[2] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216. (cited on Page 8)

[3] Benkert, R., Prabhushankar, M., and AlRegib, G. (2022). Forgetful active learning with switch events: efficient sampling for out-of-distribution data. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2196–2200. IEEE. (cited on Page 21)

[4] Bose, R. J. C., van der Aalst, W. M., Žliobaitė, I., and Pechenizkiy, M. (2011). Handling concept drift in process mining. In *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings 23*, pages 391–405. Springer. (cited on Page 19)

[5] Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 59–66. AAAI Press. (cited on Page 16)

[6] Cacciarelli, D. and Kulahci, M. (2023). A survey on online active learning. (cited on Page 1, 2, 11, 14, 17, 19, and 20)

[7] Cacciarelli, D., K. M. T. J. S. (2023). Robust online active learning. *Quality amp; Reliability Eng.* (cited on Page 22)

[8] Castellani, A., Schmitt, S., and Hammer, B. (2022). Stream-based active learning with verification latency in non-stationary environments. In *Lecture Notes in Computer Science*, pages 260–272. Springer Nature Switzerland. (cited on Page 20)

[9] Cesa-bianchi, N., Gentile, C., and Zaniboni, L. (2004). Worst-case analysis of selective sampling for linear-threshold algorithms. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press. (cited on Page 19)

[10] Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542. (cited on Page 9)

[11] Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, 15:201–221. (cited on Page 12)

[12] Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE symposium series on computational intelligence*, pages 159–166. IEEE. (cited on Page 31)

[13] Dasgupta, S. and Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. (cited on Page 16)

[14] Dasgupta, S., Kalai, A. T., and Monteleoni, C. (2005). Analysis of perceptron-based active learning. In Auer, P. and Meir, R., editors, *Learning Theory*, pages 249–263, Berlin, Heidelberg. Springer Berlin Heidelberg. (cited on Page 19)

[15] Dataverse, H. (2008). Data expo 2009: Airline on time data. (cited on Page 30)

[16] Davenport, T. H. and Harris, J. G. (2007). Competing on analytics: The new science of winning. *Harvard business review press, Language*, 15(217):24. (cited on Page 7)

[Davis et al.] Davis, R., Lii, K., and Politis, D. Remarks on some nonparametric estimates of a density function, sel. work. murray rosenblatt.(2011) 95–100. (cited on Page 23)

[18] Dimitriadou, K., Papaemmanouil, O., and Diao, Y. (2016). Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856. (cited on Page 22)

[19] Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, pages 286–295. Springer. (cited on Page 30)

[20] Gama, J., Sebastiao, R., and Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine learning*, 90:317–346. (cited on Page 32)

[21] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37. (cited on Page 21)

[22] Hao, S., Hu, P., Zhao, P., Hoi, S. C., and Miao, C. (2018). Online active learning with expert advice. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):1–22. (cited on Page 19)

[23] Harries, M., Wales, N. S., et al. (1999). Splice-2 comparative evaluation: Electricity pricing. (cited on Page 30)

[24] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer New York. (cited on Page 8)

[25] Kottke, D., Krempl, G., and Spiliopoulou, M. (2015). Probabilistic active learning in datastreams. In *Advances in Intelligent Data Analysis XIV: 14th International Symposium, IDA 2015, Saint Etienne. France, October 22-24, 2015. Proceedings 14*, pages 145–157. Springer. (cited on Page 25 and 32)

[26] Krawczyk, B., Pfahringer, B., and Woźniak, M. (2018). Combining active learning with concept drift detection for data stream mining. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2239–2244. IEEE. (cited on Page 20)

[27] Krempl, G., Kottke, D., and Spiliopoulou, M. (2014a). Probabilistic active learning: Towards combining versatility, optimality and efficiency. In Džeroski, S., Panov, P., Kocev, D., and Todorovski, L., editors, *Discovery Science*, pages 168–179, Cham. Springer International Publishing. (cited on Page 25)

[28] Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al. (2014b). Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1):1–10. (cited on Page 2, 18, 20, and 21)

[29] Lang, A., Mayer, C., and Timofte, R. (2021). Best practices in pool-based active learning for image classification. (cited on Page 12)

[30] Liu, Y. (2004). Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of chemical information and computer sciences*, 44(6):1936–1941. (cited on Page 12)

[31] Loy, C. C., Hospedales, T. M., Xiang, T., and Gong, S. (2012). Stream-based joint exploration-exploitation active learning. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1560–1567. IEEE. (cited on Page 21)

[32] Lu, J., Zhao, P., and Hoi, S. C. (2016). Online passive-aggressive active learning. *Machine learning*, 103:141–183. (cited on Page 19)

[33] Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2057–2068. The Impact of Soft Computing for the Progress of Artificial Intelligence. (cited on Page 21)

[34] McCallum, A., Nigam, K., et al. (1998). Employing em and pool-based active learning for text classification. In *ICML*, volume 98, pages 350–358. Citeseer. (cited on Page 12)

[35] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math. (cited on Page 7)

[36] Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076. (cited on Page 23)

[37] Pham, T., Kottke, D., Krempl, G., and Sick, B. (2022). Stream-based active learning for sliding windows under the influence of verification latency. *Machine Learning*, pages 1–26. (cited on Page 20 and 21)

[38] Rashidi, H. H., Tran, N. K., Betts, E. V., Howell, L. P., and Green, R. (2019). Artificial intelligence and machine learning in pathology: The present landscape of supervised methods. *Academic Pathology*, 6:2374289519873088. (cited on Page 8)

[39] Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 44:206–227. (cited on Page 7)

[40] Schröder, C. and Niekler, A. (2020). A survey of active learning for text classification using deep neural networks. *arXiv preprint arXiv:2008.07267*. (cited on Page 13)

[41] Schumann, R. and Rehbein, I. (2019). Active learning via membership query synthesis for semi-supervised sentence classification. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 472–481, Hong Kong, China. Association for Computational Linguistics. (cited on Page 11)

[42] Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. (cited on Page 10, 11, and 12)

[43] Settles, B. (2012). Active learning: Synthesis lectures on artificial intelligence and machine learning. *Long Island, NY: Morgan & Clay Pool*, 10:S00429ED1V01Y201207AIM018. (cited on Page 11, 12, 15, and 16)

[44] SINGH, A. (1989). Review article digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, 10(6):989–1003. (cited on Page 23)

[45] Smailović, J., Grčar, M., Lavrač, N., and Žnidaršič, M. (2014). Stream-based active learning for sentiment analysis in the financial domain. *Information Sciences*, 285:181–203. Processing and Mining Complex Data Streams. (cited on Page 12)

[46] Sullivan, D. (2012). Google: 100 billion searches per month, search to integrate gmail, launching enhanced search app for ios. *Search Engine Land*. (cited on Page 1)

[47] Taylor, P. (2022). Total data volume worldwide 2010-2025. (cited on Page 1)

[48] Tharwat, A. and Schenck, W. (2023). A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820. (cited on Page 2, 10, 11, 12, 13, 15, 16, and 17)

[49] Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. (2014). Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60. (cited on Page 30)

[50] Wang, L., Zhang, M., Jia, Z., Li, Q., Bao, C., Ma, K., Zhu, J., and Zhong, Y. (2021). Afec: Active forgetting of negative transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22379–22391. (cited on Page 21)

[51] Wassermann, S., Cuvelier, T., and Casas, P. (2019). Ral-improving stream-based active learning by reinforcement learning. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) Workshop on Interactive Adaptive Learning (IAL).* (cited on Page 22)

[52] Zhang, Y., Wen, J., Wang, X., and Jiang, Z. (2014). Semi-supervised learning combining co-training with active learning. *Expert Systems with Applications*, 41(5):2372–2378. (cited on Page 16)

[53] Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3. (cited on Page 16)

[54] Žliobaitė, I., Bifet, A., Pfahringer, B., and Holmes, G. (2013). Active learning with drifting streaming data. *IEEE transactions on neural networks and learning systems*, 25(1):27–39. (cited on Page 20 and 31)