Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Chemistry, Biotechnology and Food Science

Navigating Model Drift: A Case Study on Classifying Occupations Using Textual Data

Susie Jentoft
Data Science

Abstract

Data underlying any machine learning model is prone to change over time in a process called model drift. The extent of change and effect on the model performance should be monitored in production settings to avoid decreasing predictive performance. This thesis explores model drift in a small case study of occupation classifications based on text variables from the Norwegian Labour Force Survey. Kolmogorov-Smirnoff drift detection is tested, together with a novel multivariate approach, using the RV coefficient, to explore local changes within occupation classes. Drift mitigation is explored using four adaptive methods: fixed-windows, weighting, Hoeffding adaptive trees and a new targeted matching approach to create training data. Feature drift was detected using both descriptive and statistical methods for one of the groups explored; a model with occupations of very different natures. Using RV values, drift was visualized and seen within classes of several of the occupations investigated. Slight decreases in model performance were observed when models were trained on a fixed, early period. Specific adaptive methods to learn under drift did not perform better than a generic approach using all data. However, within classes where gradual drift was visually seen, an adaptive weighting algorithm performed best. In the occupation class that showed a recurrent drift pattern, the novel targeted matching algorithm performed slightly better than other methods. Further investigations on how these methods perform on larger classification models are recommended to generalize these findings.

Sammendrag

Data underliggende enhver maskinlæringsmodell er utsatt for endringer over tid i en prosess kalt modellavdrift. Omfanget av endringene og effekten på modellens ytelse bør overvåkes i produksjonsmiljøer for å unngå reduksjon i prediktiv ytelse. Denne avhandlingen utforsker modellavdrift i en case-studie av yrkesklassifikasjoner basert på tekstvariabler fra den norske arbeidskraftsundersøkelsen. Kolmogorov-Smirnoff modellavdriftsdeteksjon testes, sammen med en ny multivariat tilnærming ved bruk av RV-koeffisienten, for å utforske lokale endringer innen yrkesklasser. Modellavdriftsreduseringen sammenlignes ved fire adaptive metoder: fasttidsvindu, vekting, Hoeffding-tre adaptive metoder og en ny målrettet matchende metode for å skape treningsdata. Egenskapsendring ble oppdaget ved bruk av både deskriptive og statistiske metoder for en av modellene som ble undersøkt; en modell med yrker med svært forskjellige karaktertrekk. Ved å bruke RV-verdier ble modellendring visualisert og observert innen klasser av flere av yrkene som ble undersøkt. Små reduksjoner i modellens ytelse ble observert når modellene ble trent på en fast, tidlig periode. Spesifikke adaptive metoder for læring under endring presterte ikke bedre enn en generisk tilnærming ved bruk av all tilgjengelig data. Imidlertid, innen yrkesklasser der gradvis endring visuelt ble observert, presterte en adaptiv vektingsalgoritme best. I yrkesklassen som viste et gjentakende endringsmønster, presterte den nye målrettede matchende algoritmen noe bedre enn andre metoder. Videre undersøkelser om hvordan disse metodene presterer på større klassifikasjonsmodeller anbefales for å generalisere disse funnene.

Acknowledgements

I want to thank my supervisors at NMBU, Kathrine Frey Frøslie and Oliver Tomic for their valuable encouragement in working on this thesis and constructive comments on the drafts. Thank you also to Boriska Toth from Statistic's Norway for the good conversations and feedback and to my division leader at Statistic's Norway, Xeni Kristine Dimakos, for supporting me throughout my study. I am also grateful to others at Statistics Norway; the Division for labour market and wage statistics for allowing me to use data from the Norwegian Labour Force Survey, Anita Dal for technical support and background information and to all those that have contributed to previous internal pilot studies on occupation classification. Finally, thanks to my family, particularly Joachim, who has supported both me and our family on this journey.

Contents

List of Figures

List of Tables

Abbreviations

ADWIN	Adaptive window algorithm
ARF	Adaptive random forest
AUC	Area under the curve
CVFDT	Concept-adaptive very fast decision tree
DWM	Dynamic weighted majority
FN	False negative
FP	False positive
FPR	False positive rate
IDF	Inverse document frequency
ISCO-08	International standard classification of occupations, 2008 edition
KS	Kolmogorov-Smirnov
ML	Machine learning
MOA	Massive online analysis
NAV	Norwegian Labour and Welfare Administration
NLFS	Norwegian Labour Force Survey
NLP	Natural Language Processing
NSI	National Statistic Institute
ROC	Receiver Operating Characteristic
SAS	Scandinavian Airlines
STYRK-08	Norwegian occupation standard used from 2011
STYRK-98	Norwegian occupation standard used from 1996 to 2010
SVM	Support vector machine
TF	Term frequency
TFIDF	Term frequency inverse document frequency
TN	True negative
TP	True positive
TPR	True positive rate
VFDT	Very fast decision tree

Chapter 1

Introduction

1.1. BACKGROUND

Statistics Norway is the National Statistic Institute (NSI) for Norway, tasked with producing the majority of the country's official statistics. This includes everything from calculating the percentage of children without holes in their teeth, the average income for a nurse, to the number of potatoes grown, and the price of all exported oil. The statistics produced form an integral part of policy and decision-making for the country. The processes involved in statistical production vary as widely as the subject areas and include project planning, data collection, data editing and imputation, estimation and confidential protection. Statistics Norway follows a series of principles established by the European Union when performing tasks, which include a commitment to quality and ensuring sound methodology [1].

There is increasing interest in machine learning (ML) algorithms, like classification models, within NSIs for solving some of the challenges faced when producing statistics. Classification is a process that has gained considerable attention [2]. ML algorithms are data-driven ways to build classification models that are more robust than rule-based methods and require fewer ongoing resources than manual classification. However, ML classification models are often trained under static conditions and are susceptible to performance decreases in dynamic environments where data is evolving. Research into new methods for learning in dynamic environments is therefore important to ensure high accuracy in models used in production settings where data is changing.

Text classification problems are generally high dimensional and often susceptible to changes. The language we use in society is constantly evolving. New concepts, words and phrases may suddenly appear, for example in the case of "covid", or be introduced gradually over time. Other words will change their meaning or die out. Language processing and models must therefore also adapt in these dynamic systems.

Many new ML algorithms, designed to handle dynamic systems, have been presented in the literature in recent years [3, 4, 5]. With this increase in interest comes more choices for implementation. There is a lack of guidelines and frameworks for choosing between the ML classification models in dynamic environments, making implementation

increasingly overwhelming for statisticians at NSIs. Additionally, these algorithms have often been tested on limited synthetic data and most have no or sparse testing on text classification problems.

1.2. MODEL DRIFT

Model drift refers to changes in the underlying data distribution of a learned model, through time [6]. This phenomenon is sometimes referred to as dataset shift [7, 8], and can be a serious challenge in production settings that assume static environments. In most cases, the performance of a model will degrade with time, known as model degradation [9]. Because of this, there has been considerable focus on the topic in the last decade, with research on defining the causes, developing detection methods, and adaptive algorithms to mitigate its effects [5].

Model drift can be organized into two general groups: concept drift (or real drift) and virtual drift [10]. These refer to what is changing in the underlying data. Consider the joint distribution $P_t(y, \mathbf{x})$ in a classification problem where t, represents a specific time point, y is the target, class variable, and $\mathbf{x}$ is the set of features in the model. According to Bayes decision theory, the joint distribution can be written as $P_t(y|\mathbf{x})P_t(\mathbf{x})$ when the target variable y can be causally determined by the features $\mathbf{x}$. If there are changes in the distribution of the features, $P(\mathbf{x})$ without changes in the conditional distribution $P(y|\mathbf{x})$, this is often referred to as virtual drift [10]. In this case, conditional distributions are equal at two time-points $P_t(\mathbf{x}|y) = P_{t+1}(\mathbf{x}|y)$ while $P_t(\mathbf{x}) \neq P_{t+1}(\mathbf{x})$, where $t + 1$ is a specified time point after t. This may be caused by a dynamic environment, changes in the processing of the upstream data (known as upstream drift), or changes in the selection probability of the feature data [11]. While the conditional probability isn't affected in virtual drift, it can result in inadequate amounts of training data at specific ranges in the data leading to decreases in model prediction [12].

In contrast, concept drift refers to when the conditional distribution $P(y|\mathbf{x})$ changes, resulting in decreasing model performance if not addressed. Common causes of this include seasonality, changing personal preferences, quality of materials and language evolution [9, 13]. In addition, upstream processes, such as changes in secondary data sources used in data pre-processing, can influence the relationship between features ($\mathbf{x}$) and the target classes (y) leading to (upstream) drift. This type of drift is generally not considered true concept drift. Lu et. al. [5] describe two underlying mechanisms that result in concept drift. Firstly, it can be a change only in the conditional distribution whereby $P_0(\mathbf{x}|y) \neq P_1(\mathbf{x}|y)$ yet $P_t(\mathbf{x}) = P_{t+1}(\mathbf{x})$. In classification tasks this results in a shift in the decision boundary (Figure 1.1). A second mechanism is when both $P(\mathbf{x})$ and $P(y|\mathbf{x})$ change so that $P_t(\mathbf{x}|y) \neq P_{t+1}(\mathbf{x}|y)$ and $P_t(\mathbf{x}) \neq P_{t+1}(\mathbf{x})$. In this case, the system has a high complexity and dynamic nature. In 2012, Moreno-Torres et. al [11] describe this scenario only briefly and note that they consider them nearly impossible to solve. However, in production settings, this is the likely underlying situation.

A third type of drift is mentioned sporadically in the literature as prior-probability shift [9]. This is when there is a change in the class distribution, $P_t(y) \neq P_{t+1}(y)$ and occurs when new classes appear, evolve or disappear. This is common in classification tasks within NSIs as both national and international standards change on occasion. While

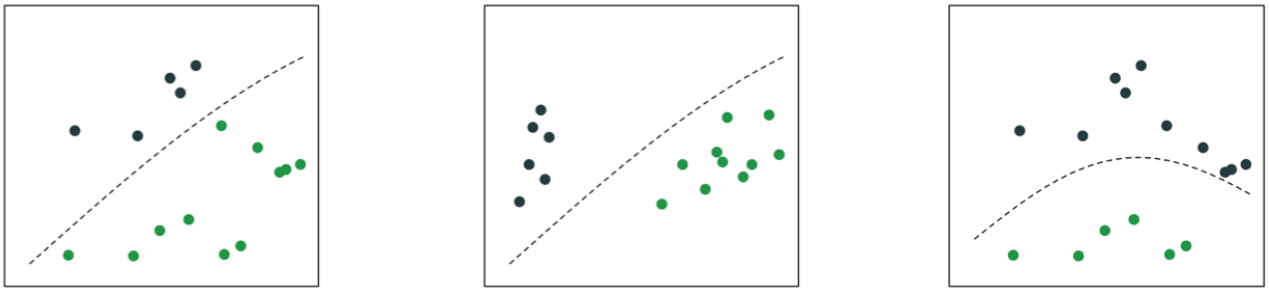


Fig. 1.1: Model drift patterns showing original with no drift (left), virtual drift (center) and concept drift (right). Two classes are shown: class 1 (●) and class 2 (●), with a decision boundary shown between them. This decision boundary is identical in virtual drift, even though the distribution has changed. The decision boundary changes when concept drift occurs.

this is an important consideration in classification problems, it is not a focus in this thesis.

The transition of the drift pattern will play a role in how easy it is to detect the change and how the learning algorithm should adapt. Bayram et. al [9] describe four main drift transition types: sudden (or abrupt [14]), gradual, incremental and recurring (1.2). Sudden drift occurs when there is a fast concept change, for example, the appearance of the term ”covid tester” when describing occupation in 2020. In text contexts, this usually represents the sudden appearance or sudden disappearance of a new term. In contrast, gradual drift occurs when a new concept replaces an old one over a longer period [15]. For example, the word ”pre-primary” was commonly used to refer to children before school age, however, in more recent years has been replaced by ”early childhood”. Similarly, incremental drift is where an old concept gradually changes over an extended time. For example the word ”monster” was often associated with books about monsters in receipt data, however, in more recent times is more associated with the energy drink brand. These first three drift types are regarded as permanent [10], where the concept change does not revert to a previous conditional relationship. The final drift type, recurring drift, refers to drift patterns where changes in concepts return to previous states. This cyclical nature can be periodic, for example, the names of Christmas products sold in shops, or may occur at random intervals [16].

The ability to detect when a drift is occurring is an important part of performance monitoring to ensure the quality of the predictions. It has gained considerable attention, with a review by Bayram et. al. in 2022 listing more than 60 drift detection methods [9]. Many of these focus on changes in performance metrics to detect changes. However, performance metrics require fully labelled data which is not always available in the case of automated classification tasks. In contrast, detection methods focusing on changes in the distribution of the features ($\mathbf{x}$) have the advantage of not requiring labelled data. The Kolmogorov-Smirnov test (KS) is a traditional, non-parametric test from the 1930s that has made a resurgence as a drift detection method in recent years [17, 18, 19]. This method was extended by Dos Reis et al. [20] who proposed a speed-up of the efficiency of the algorithm to allow it to be used in an online context. Other popular algorithms used for drift detection based on feature distributions include the

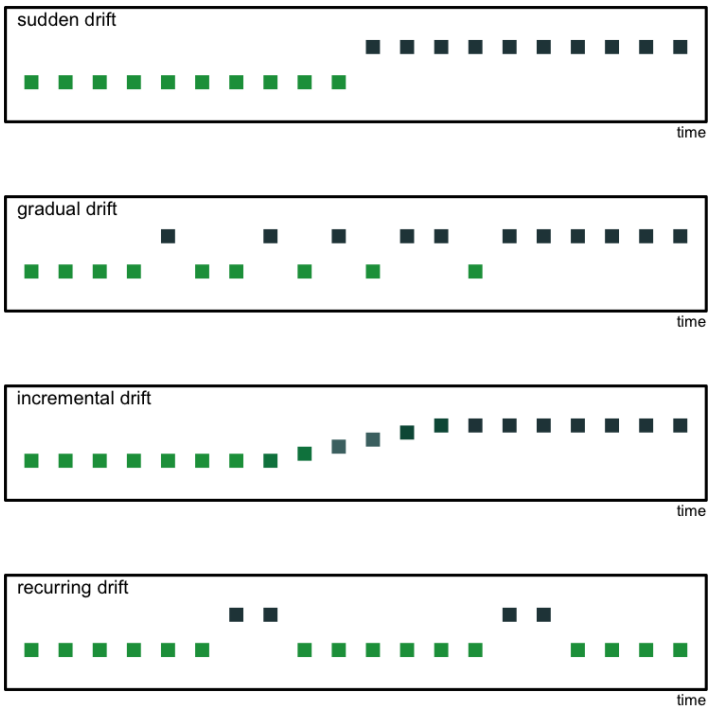


Fig. 1.2: Model drift transitions showing sudden (top), gradual (middle-top), incremental (middle-bottom and recurring (bottom).

population stability index [21], adaptive windowing (ADWIN) [22] and Least squares density difference [23]. All these methods take a univariate approach and extending them to a multivariate case generally involves a conservative penalty, often reducing their efficiency. The lack of multivariate approaches is particularly a problem with text classification tasks that regularly have thousands or more features.

An alternative approach for drift detection could be to use a multivariate technique. The RV coefficient is a similarity measure between two sets of observations; a matrix correlation coefficient [24]. While more commonly associated with comparing data on the same individuals over time, it provides a multivariate approach to compare similarities, not seen in other drift methods. We therefore test it in this novel context to explore drift detection and explanation in text data in this thesis.

1.3. LEARNING WITH DRIFT

Lu et. al. [5] discuss the problem of learning under drift in terms of three key areas: drift detection, drift understanding and drift adaption. Drift detection, as discussed in the previous section (section 1.2), refers to methods that try to determine if changes are occurring in features ($P(\mathbf{x})$) and concepts ($P(y|\mathbf{x})$). Drift understanding refers to exploring when, how and where the drift is occurring. This is important to decide the best adaption strategy. Most

drift detection methods have a built-in trigger mechanism that specifies the time point of a change. The other aspects of understanding are not as well researched in the literature and include the severity of the change, and where in the data it is occurring. Drift detection methods that compare the distribution of the features ($P(\mathbf{x})$) will sometimes calculate a distance that indicates the severity. However, this is often not communicated directly to the user in packaged algorithms and doesn't always reflect the severity of the impact on the model. There has been some research on monitoring local error rates in decision trees to address more specifically where drift is occurring [25]. However, this has not been extensively studied or implemented into the main public domain drift packages. In contrast, the final area of learning under drift (drift adaption), has been well studied in the literature. The remaining paragraphs in this section discuss drift adaption and the various approaches that have been proposed in the literature.

Retraining is by far the most popular method for combating model degradation. It involves gathering new labelled data and feeding it into an algorithm to use for new predictions. It will generally boost the performance for that time period (see Figure 1.3) but raises also the question of how much data to use for the training process.

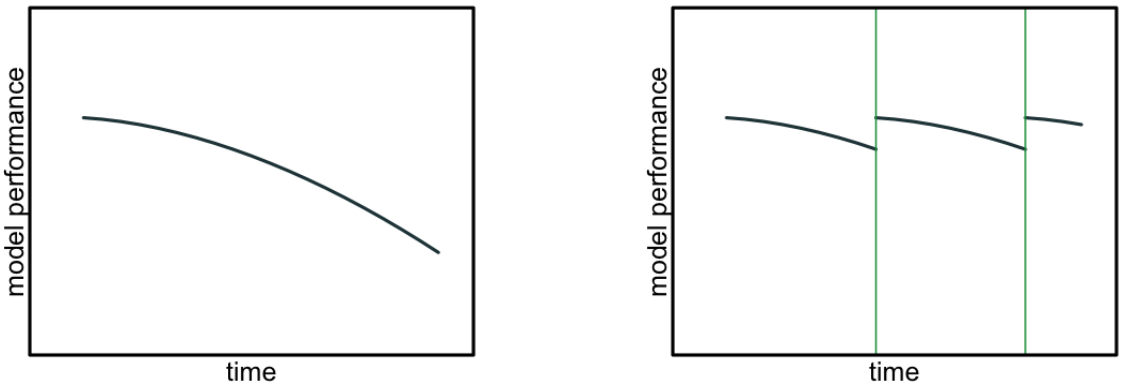


Fig. 1.3: Model degradation (left), and model degradation with retraining (right). Three retraining time points are given (—).

Over time, the algorithms should both learn from newly trained data and have some system to forget data that is no longer needed. Grossberg [26] was the first to describe this stability-plasticity trade-off. The algorithm should maintain the information it has previously learned which is relevant for the classification problem, or potentially relevant in the future (stability) while outdated information should be replaced with new knowledge (plasticity) [16, 9]. Different algorithms solve this balance in different ways.

Window-based approaches are common in adaptive learning [5]. These involve using a pre-defined time-width interval for training the algorithm. For example, all data from the most recent year may be used to train the algorithm to predict the next time period. A small window will likely contain the most relevant data as generally, data from the nearest time period will be most similar to that which is being predicted. However, a larger window will include more data, which in stationary environments will produce higher model performance. Deciding the window size can

be challenging which has led to adaptive window approaches such as the popular ADWIN (adaptive windowing) algorithm [22]. This increases the window size automatically when there is little drift and decreases it in times of change.

A similar approach to windows is the use of weighting training data. Again, labelled data collected at closer time points will generally be better at predicting classifications when there is a dynamic environment. Data closer to the time-point being predicted are therefore weighted higher than those further away. Weighting can be at a global level or combined with window techniques for local adaptation. These approaches have also been used to adapt at a spatial level [27].

Another new approach is to create training data that is most similar to the batch being predicted rather than simply using that which is the most recent. This approach was presented by Mallick et. al. using their *Matchmaker* algorithm [4]. Results from their work showed both accuracy and speed improvements. Since the algorithm is not implemented using open-source software, it is not considered further in the thesis.

Elwell and Polikar [16] classified adaptive learning algorithms into two groups; active and passive learning. In active learning, a drift detection mechanism runs to determine when the algorithm should actively learn. In contrast, passive learning refers to approaches where models are continuously updated. The ADWIN algorithm is an example of an active learning approach as it has a drift detection mechanism to determine the window size. In contrast, a fixed-window learning approach is an example of a passive learning approach. In general, active approaches may work best when there are sudden drift changes and it is clear when the change occurred to base new training data on. Comparatively, passive approaches may be preferred when there is gradual drift as this type of drift is often harder to detect and it is less clear what period of training data to base the new model on [10].

Adapting the training data through retraining with windows, weighting or matching isn't the only way to mitigate drift. Lu et. al [5] mention two additional approaches to adaptive learning: ensemble methods and adaptive models. Ensemble methods involve reusing older models that are extended or use adaptive voting rules. Ensemble methods are particularly popular in drift settings as the models are easily updated with new learners being added or removed without much extra computation or optimization. Gomes et. al. [28] proposed an adaptive Random Forest algorithm for classification problems in the presence of concept drift known as Adaptive Random Forests (ARF). These use a drift detection (for example ADWIN) per tree, together with re-sampling techniques to update in evolving data scenarios [28]. An alternative to adjusting the ensemble itself is to adapt the voting procedure. This idea was presented as a weighted majority in Littleston and Warmuth [29], whereby "expert" learners, that often predict correctly, are weighted higher than others. This idea was utilized by Kolter and Maloof [30] in their Dynamic Weighted Majority (DWM) algorithm. It maintains a weighted pool of "expert", base learners which is added to if the global algorithm makes a mistake. If an "expert" makes a mistake then its weight is reduced and may be removed. The Learn++ Non-Stationary Environment algorithm (Learn++NSE) was a significant further development of this, where voting depends on a time-adjusted accuracy [16]. It has been shown to perform well on a range of drift rates

and types including recurrent patterns, which are often very difficult for adaptive methods to learn [16].

A final approach for adaptive learning is adaptive models. The Very Fast Decision Tree learner (VFDT) is an example of this where the model itself adapts. VFDT uses decision trees and a Hoeffding bound to decide how many examples are needed to split each node [31]. It is very efficient for use on big data streams due to its online approach where examples are only ever processed once. This was extended by Hulten et. al. specifically for concept drift scenarios in the Concept-adapting Very Fast Decision Tree learner (CVFDT) [32]. Bifet and Gavalada [33] also extended the VFDT further by enriching it with an active change detection mechanism to create an adaptive approach which they called the Hoeffding adaptive tree.

Several authors have acknowledged the lack of framework around adaptive learning [10]. For those working in NSIs, tasked with setting up and testing ML classification algorithms, there is little guidance on which methods perform best under which drift conditions. New studies in this area often focus on creating more efficient techniques, particularly for big data streams. While some NSIs are using new large data sources, the majority of official statistics are produced on smaller, more stable batched-type data. Guidance on choosing these methods is lacking. Barros et. al. performed a large comparison study on concept drift but with a focus on drift detection methods [14]. The data sets used included a small number of numerical features and not textual data where there can be many thousands (or millions) of features in sparse matrices. When Gomes et. al. [28] presented their popular ARF algorithm, only one of the 16 test data used was textual and this was dropped from most of the analysis, due to memory difficulties. This seems to be common in comparative studies on model drift as it is generally easier to simulate numerical data. One comparison study that did include textual data classification was again limited to drift detection methods and not extended to adaptive approaches [13]. Of the example data used for drift experiments mentioned in Lu et. al. [5], none of the synthetic data was textual and the three real-world data with realistic feature numbers for textual analysis ($>$1000) were two-class problems. Text classification problems in NSIs are generally multi-class by nature with the intent of classifying to an existing (often international) classification framework. This lack of knowledge on how adaptive algorithms perform in this context is a key area that this thesis aims to address.

Finally, the ability to easily implement an adaptive algorithm on an NSI's data platform is crucial. NSIs often have large and rigid IT systems with limitations on what programming languages may be supported and limited access to license costs. Some new algorithms are presented by proprietary businesses [4] and may therefore not be easily implemented in NSIs where IT infrastructure may be both complex, but also limited in the options for language/tools allowed. MOA (Massive Online Analysis) is a popular open-source implementation framework that includes tools for learning in dynamic environments [34]. It has a huge array of implementations of algorithms. The software is written in Java and is therefore not easily implemented at Statistics Norway. The focus of this thesis is therefore on methods that are implemented in the open-source language Python where modules can be easily integrated into Statistics Norway's data platform.

1.4. CASE STUDY: OCCUPATION CLASSIFICATION IN THE NLFS

The Norwegian Labour Force Survey (NLFS) is a large, interview-based survey, that collects information to produce key statistics on the Norwegian labour market. The occupation of participants is classified based mostly on responses from open questions that are transcribed to text variables. While the classification task is currently done using mostly manual coding by a team of people at Statistics Norway, ML algorithms have been initially tested in pilot projects. However, NLFS is a long-running survey and it is important to consider how much drift has occurred in the data and how this might affect the performance of the ML algorithms. For example "ekspeditør" was a common word to describe a shop assistant before 2000, but has been almost entirely replaced by "butikkmedarbeider" in recent years. Determining the extent of this type of change and how it affects static models is important to assess before any implementation in a production setting. In addition to potential concept drift, changes in the production setting can lead to upstream drift. By comparing adaptive approaches to learning we hope to determine the best approach for classification in this type of NSI setting.

1.5. AIM OF THIS THESIS

The overall aim of this thesis is to investigate drift patterns in occupation data from the NLFS and to compare adaptive ML approaches for occupation classification models. A holistic approach is taken to explore learning under drift, looking at all 3 of the main areas; drift detection, drift understanding and drift adaption. For this task, we have chosen to reduce the scope to 2 classification models, each with 4 occupation classes. One model (referred to as Group A) uses 4 occupations with very diverse occupations while the other (Group B) includes more similar occupations. These groups provide a proof-of-concept, case study for testing drift detection, visualization and adaption techniques. The following research questions will be addressed in this thesis:

RQ1: Is drift occurring in the chosen occupation groups?

The first objective is to determine if drift is occurring in the data for the occupation groups chosen. We have a time series of 27 years and we want to see if occupations have been described in changing ways during this time.

RQ2: How does drift impact the performance of the occupation classification models?

In addition to investigating whether drift is occurring, we want to find out if these changes would affect the model performance. This will let us determine if drift patterns are of a concept drift type of virtual drift, without affecting the relationship between features and class ($p(y|\mathbf{x})$).

RQ3: How well do standard adaptive approaches including fixed-window, weighting, and Hoeffding adaptive trees, mitigate drift in our occupation classification problem?

Given that Statistics Norway is interested in using a classification model in a production setting, we are interested in determining which of these 3 approaches is best suited to our occupation classification problem. In the presence of drift, can these approaches help our classification model adapt to changes?

RQ4: Can our novel implementation of targeted matching create better training data to mitigate drift effects in occupation classification models compared to standard approaches?

Our final objective is to implement a new targeted matching algorithm to create more relevant training data. We compare the performance of this method against standard tools for adaptive learning to see if this approach performs better under drift conditions.

This thesis is organized into the following sections. Chapter 2 describes the data, methods and algorithms tested. Chapter 3 gives an overview of the results from the two occupation models with case studies of four dissimilar occupations and four similar occupations. A discussion of these results is given in chapter 4 and the final chapter 5 provides some concluding remarks.

Chapter 2

Methods

Model drift poses significant challenges in maintaining model performance over time. As a consequence, a robust strategy for model adaptation and monitoring should be used. The following chapter outlines the methodologies used in this thesis research. Our data is described in further detail followed by an outline of the pre-processing techniques used for the textual data. The methods for visualizing changes in the data and determining critical deviation points are defined. An explanation of the model tuning and adaptive learning approaches is then given.

2.1. DATA: THE NORWEGIAN LABOUR FORCE SURVEY

The Norwegian Labour Force Survey (NLFS) is a large, quarterly, sample survey, run by Statistics Norway since 1972 [35]. Residents are interviewed about their labour market activities to establish statistics for the country's unemployment rate, the number of people in and outside the workforce and the number of hours worked. Around 24 000 people, between the ages of 15 to 89 years, are invited to participate in the survey each quarter. The survey has a rotating panel design, meaning there is around 3000 new participant each quarter, while the rest have been part of the survey in previous quarters [36]. The data is used for both national and international statistics and is regulated by law including its precision requirements [37]. It is of high interest to policymakers, economists, private companies and researchers.

Statistical indicators are divided into domains such as regions, age, sex and occupation. These domain variables are collected at the time of the interview or linked from administrative data shortly after. Occupation is collected from the interviewer, and written as free text (in Norwegian) based on self-reported occupation and the main activities for the person's job. The text is then classified to the Norwegian occupation standard (STYRK-08); a standard based on the International Standard Classification of Occupations (ISCO-08) [38, 39]. The STYRK-08 occupation standard was first published in 2011, however has some similarities in structure to the previous standard (STYRK-98). The previous STYRK-98 was introduced into the NLFS in 1996. Before this, a national standard for occupation was used which differs significantly from both STYRK-98 and STYRK-08 [40]. This early standard is not discussed further

in this thesis.

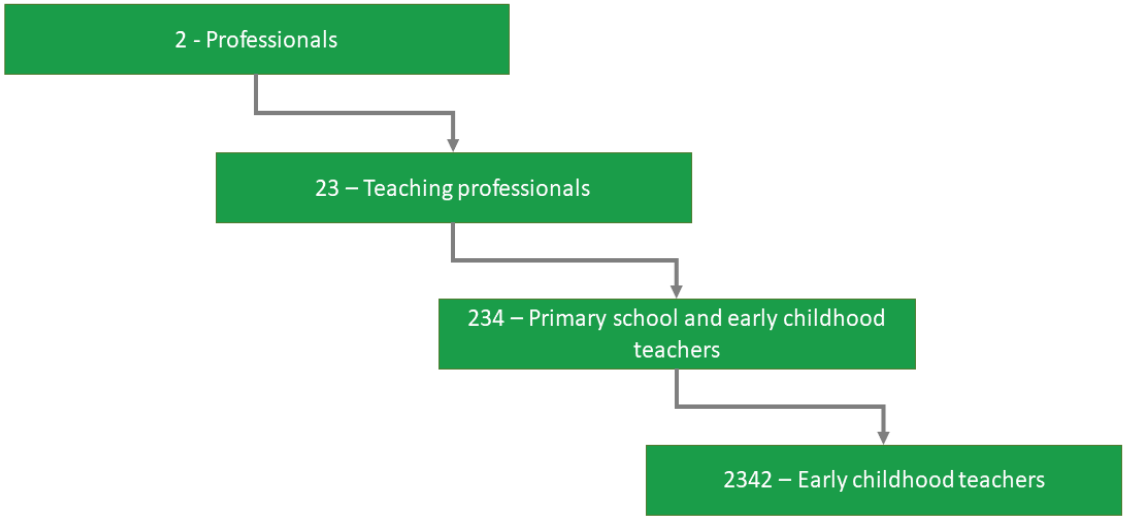


Fig. 2.1: Example of the hierarchy of the four-digit in the occupation standard (STYRK-08) for early childhood teachers.

The STYRK-08 occupation standard is hierarchical and consists of 4 digits. Figure 2.1 provides an example of an occupation code *2342*; the code for early childhood teachers. The first digit indicates which of the 10 major groups the occupation belongs to (see Table 2.1). The job's required skill level is a main consideration in defining the first-digit groups. Within each major group, occupations are grouped into sub-major (2nd digit) and minor (3rd digit) groups. These groupings have been established based on the job's skill specialization, i.e. the knowledge required, and/or the tools, materials and goods used and produced [39].

Table 2.1: Description of the 10 major occupation groups in STYRK-08.

First occupation digit	English description	Norwegian description
0	Armed forces and unspecified	Militære yrke og uoppgitt
1	Managers	Ledere
2	Professionals	Akademiske yrke
3	Technicians and associate professionals	Høyskoleyrke
4	Clerical support workers	Kontoryrke
5	Service and sales workers	Salgs- og serviceyrke
6	Skilled agricultural, forestry and fishery workers	Bonder, fiskere mv.
7	Craft and related trades workers	Håndverkere
8	Plant and machine operators and assemblers	Prosess- og maskinoperatorer, transportarbeider mv.
9	Elementary occupations	Renholdere, hjelpearbeidere mv.

The current method of classifying occupation text to the STYRK-08 standard for the NLFS includes manually

choosing a classification category based mostly on text variables collected in the interview. Variables include the company name, company activity, self-reported occupation, and the main tasks of the job. The questions have been consistent since 1996, despite changes to other questions in the survey [41, 42, 43]. They are generally asked and written in Norwegian, however, occasionally respondents are recorded in English. The following questions (with variable names in bold) are asked in the survey:

- (**B_NAVN1**) Hvor hadde du arbeid i uka X? Vi ønsker navnet på bedriften. (Where did you work in week X? We want the name of the establishment).
- (**B_ART1**) Hva slags virksomhet drives i bedriften? (What is the main activity of the establishment?)
- (**Y_TITL1**) Hva er ditt yrke i denne bedriften? (What is your occupation in this establishment?)
- (**Y_TEK1**) Hva er dine viktigste arbeidsoppgaver? (What are your main tasks?)

The responses to these questions provide the text used in the classification models in this thesis. These questions are asked multiple times if a survey participant has more than one job. We have chosen to only use data from the main job. We have also chosen to only include the first instance of the occupation for each participant. The NLFS is a panel design and individuals participate a total of 8 times. They are only asked the four questions above in the first interview; in subsequent interviews, occupation information is copied from the previous quarter if they have remained in the same job. This will therefore only appear once in our data. If the participant has changed jobs, they will be asked the questions again and this will be included in our data.

In addition to the text variables from the interview, coders have (in recent years) access to register information. While we have chosen not to include this information in the classification models, it can introduce drift to our data. Changes occurring in the register information available to those coding can affect how the occupation is classified. Previous studies have shown that when a classification code is available, coders are more likely to choose that provided, compared to when coding blindly. This is a part of the upstream data processing which can introduce upstream drift. The following paragraph describes the register data used by the coders and provides background information on the upstream data processing.

Since 2015, participants in the NLFS have been linked directly to the information in the *A-ordning* register using personal identification numbers. The *A-ordning* register is a coordinated system between Statistics Norway, the Norwegian tax authority (Skattetaten) and the Norwegian Labour and Welfare Administration (NAV). All employers must report information on employees' hours worked, earnings and occupation every month [44]. Employers report occupation in the older standard (STYRK-98) but with an additional 3 digits; a 7-digit code. These are then converted to 4-digit STYRK-08 codes. Prior to 2015, register information on occupation from the Arbeidsgiver/arbeidstaker register (*Aa-register*) was linked to the NLFS. This register was maintained by the Norwegian Labour and Welfare Adminstration (NAV) and had compulsory requirements on registering workers. A copy of the register was sent

to Statistics Norway every week. From 2001, it was compulsory for companies to report occupation codes for their employees to the Aa-register [45] except for those working for municipalities and the government. They reported job titles rather than occupation until 2007 for municipality workers and until the introduction of *A-ordning* for government workers [40]. The quality of the occupation codes was challenging in the beginning but with the introduction of both automatic an manual controls and coding, improved over the first years [46]. Several challenges remained before this register was phased out, replaced by the *A-ordning*. The occupation codes in the registers are based on those provided by companies and are generally still associated with the position description rather than tasks completed by the worker. Therefore, they are still limited in quality as they consider the actual tasks performed by the person.

The data used in this thesis is based on occupation classifications from the Norwegian Labour Force Survey (NLFS) for a period of 27 years; from 1996 to 2022. A selection of 8 occupations were chosen to use in this thesis to build 2 classification models. Four of the occupations were significantly different from one another (referred to as Group A) and occurred with relatively high frequencies providing a good amount of data in all years. The other four were more similar (Group B) and were less frequent, yet still present in all years of the data. These groups provide a proof-of-concept, case study for testing drift detection, visualization and adaption techniques. Table 2.2 provides a description of the 8 occupations and the number of observations in our data.

Table 2.2: Description of the 8 occupations used in this thesis

Occupation code (STYRK-08)	Occupation description (STYRK-08)	Previous occupation code (STYRK-98)	Privous occupation description (STYRK-98)	Number of observations	Group
2224	Registered Nurse for the Mentally Subnormal (RNMS)	3232	RNMS	1373	A
2342	Early childhood teachers	3320	Pre-primary education teaching associate professionals	3577	A
4322	Production clerks	4132	Logistical clerks	1084	A
6130	Mixed crop and animal producers	6130	Crop and animal producers	3809	A
3151	Ship engineers	3141	Ship engineers	578	B
3152	Ships deck officers and pilots	3142	Ships deck officers and pilots	1068	B
3153	Aircraft pilots and related associate professionals	3143	Aircraft pilots	268	B
5111	Travel attendants and travel stewards	5111	Travel attendants and travel stewards	302	B

2.2. TEXT PRE-PROCESSING

The four textual variables mentioned in 2.1 were used to learn prediction models for occupation classification. However, text variables cannot be directly sent as input to models. They need to be pre-processed and tokenized to create features that can be read as input to the model. Pre-processing and feature engineering are vital parts of model building based on text variables and can have effects on model performance. In this section, we outline the steps taken to process the textual data.

2.2.1. Cleaning texual variables

Text variables were cleaned by first removing all numbers. While in some settings numbers in text strings may be useful, in many contexts they will contribute to excessive features after tokenization with little effect on performance. While some studies have shown this pre-processing step may not specifically improve a model's performance [47], decreasing the number of features can still improve the speed of a model and we have therefore chosen to remove them completely.

Next, special characters and symbols were replaced with a space. In many situations, the description of the main activity was described as a list, separated by commas, dashes, periods or forward slashes. As we want our model features to be extracted from clean words, it was important to replace them with a space so that meaningful words could be used. During this process, double spaces may be introduced in the texts. We removed all double spaces with single ones at this point.

When features are created from text, the process is case-sensitive. This means that the word ”Barnehage” will be considered as a completely separate feature from the same word without an uppercase: ”barnehage”. We therefore converted all text to lowercase before further processing. See Figure 2.2 for an example of the pre-processing stages. Text cleaning was performed using the statistical program, R [48], and the *tm* R-package for text mining [49].

2.2.2. Removal of stop words

Stop words are words that are commonly used in text but generally don't help in natural language processing (NLP) tasks like our occupation classification problem. Examples of stop words include ”and” and ”of” in English or ”og” and ”av” in Norwegian. These help the reader understand the context and enable the flow of the text. However, as they are often not useful for solving classification and other NLP tasks, they are generally removed from textual data before feature extraction. This aims to better promote features with meaning and can improve the performance of the model [50].

There is no universal list of stop words, as they are language (and in some cases context) dependent. However, attempts have been made to create commonly used lists of stop words, that can easily be used by others in pre-processing. A common such repository of stop words is the snowball repository [51] which is implemented in the

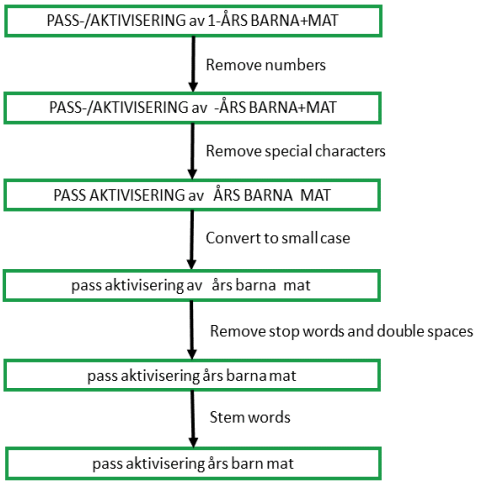


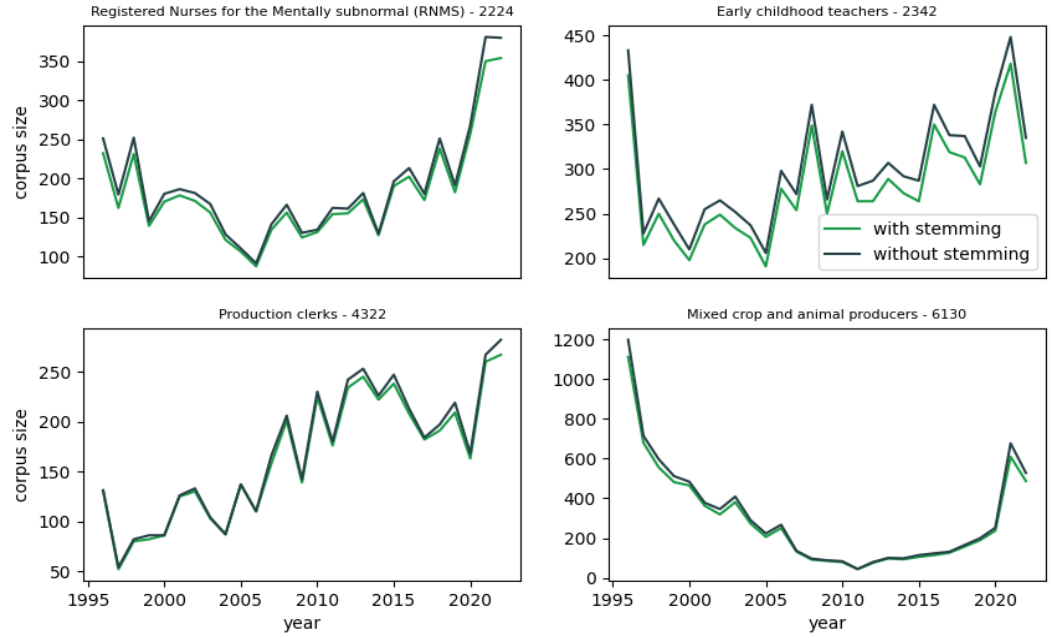
Fig. 2.2: Example of text pre-processing stages for a given example.

tm R-package [49]. This includes 176 commonly used Norwegian stop words which we have removed from our text. One word in the list was not used as it can be associated with an occupation. The word ”dykk” means ”your” in Nynorsk however is also the root form for the verb to dive. As this word may be used to describe occupations where diving is involved we chose to remove it from the stop word list. An additional word ”as” was added to the stopword list. This is the abbreviation used to indicate a limited liability company (”Aksjeselskap”). As we used the company name in the classification models, ”as” occurred very frequently and had no real value in the models and was therefore removed. For a full list of stop words used see Appendix A.

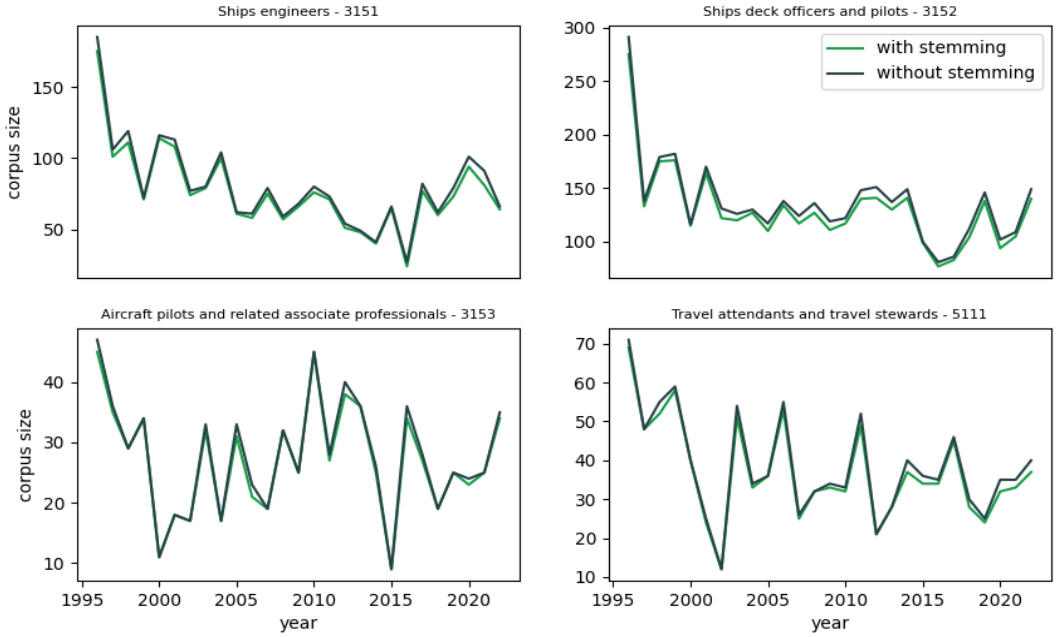
2.2.3. Stemming

The process of stemming aims to return a word to its root form. A word can have many different forms, for example, in many languages verbs are conjugated to give context on when and by whom the action occurred. Stemming is an important part of pre-processing to align similar words with different forms. Again, this alignment results ideally in fewer, more meaningful features. Figure 2.3 shows the number of words in the corpus with and without stemming for the 8 occupations. We can see that stemming reduces the number of words slightly in our data.

Norwegian is no exception when it comes to using multiple forms of words to express context. Verbs are conjugated based on when the action occurred (but not by the object doing the action). For example, ”pleie” is the verb to care for which changes to ”pleier” in the present tense and ”pleid” in the past tense. Norwegian declension is rather intricate, involving alterations in nouns based on factors such as singular or plural forms, definiteness, and the gender of the noun. To illustrate this, the Norwegian word for child ”barn” changes to ”barnet” in the definite form when referring to a specific child, or ”barna” when referring to plural, specific children. In addition to nouns



(a) Group A



(b) Group B

Fig. 2.3: Number of words in the corpus each year for 8 occupations. Corpus size is given with and without stemming.

and verbs, Norwegian has alterations in adjectives, adverbs and superlatives. Stemming tries to find the root of all these types of words. Figure 2.2 gives an example of the pre-processing stages including stemming.

The majority of text in the occupation descriptions is in Norwegian. We used the Porter stemming algorithm implemented in the R-package *tm* [49]. This algorithm works by stripping the suffix of words using a rule-based system to find the root word [52]. It was originally designed for the English language but has been adapted for other languages including Norwegian. The stemming in this implementation does not always result in meaningful and real words. For example, ”barnehage”, meaning kindergarten in Norwegian, is a compound word made up of ”barn” (child) and ”hage” (garden). This stems to ”barnehag” which while isn't a real word, will reduce variations (for example ”barnehager”) down to the same root. Verbs in Norwegian are reduced not to the infinitive form as in English but to the imperative form. For example, the verb ”lære” (to learn), stems to ”lær”, which is a command. The word ”lærer” is both a noun (teacher) and a verb (learn) in present form. Stemming doesn't consider the context of the word and in this case, stems the word as if it is a verb. While this is not necessarily the correct root of the word, the process still aids in reducing the number of features created that have identical or very similar meanings.

2.2.4. Tokenization

Tokenization is the process of converting a corpus of text into a matrix form. This is the matrix of features ($\mathbf{x}$) that is then utilised to train the models. ”Bag-of-words” is one simple way to achieve this by counting up the occurrences of each word in a text string. This is performed for each text in the data to form the matrix. Table 2.3 provides an example of this process. The matrix will generally be sparse with many zeros and a handful of ones and higher integers scattered throughout.

Table 2.3: Example of pre-processing and tokenization of text using ”bag-of-words”

Original text	Pre-processed text	Bag-of-words			
		pass	barn	barnehag	barnehagelær
Passer på barn	pass barn	1	1	0	0
Passer på barn i barnehage	pass barn barnehag	1	1	1	0
Barnehagelærer	barnehagelær	0	0	0	1

We performed tokenization with ”bag-of-words” using the cleaned versions of the three variables for occupation. The variables were pasted together prior to tokenization, into one string. Additional white spaces were removed prior to tokenization.

An advantage of ”bag-of-words” is its simplicity, however, there are some disadvantages. The order of the words is lost in the process which can considerably reduce the meaning of the text. An alternative is to use combinations of words (n-grams) where several words are used to create features. In this way, the order of the words is partially preserved. The data on occupation codes is relatively short and there are not many instances where the order of the words would change the meaning and therefore occupation code. Additionally, using word n-grams increases the

number of features creating larger matrices and more complexity. This was therefore not pursued further.

A larger problem in our data is the presence of spelling errors and abbreviations. For example, "Data prog." and "Data programmer" will result in different features although referring to the same activity. A way to address this is to create tokens based on a small set of letters, sometimes referred to as k-grams. Table 2.4 Shows an example of using 3-gram characters to build tokens. This may be done using spaces (as represented by "_" in the example in Table 2.4 or not. Müller [53] explored both word n-grams and character k-grams in a classification task at Statistic Norway with similar short text descriptions. He found that k-grams performed best in this context, with a range of 2 to 3 characters, a similar result to other text classification tasks of this type [54]. We tested both word and 3-gram character tokenization of our text variables in this thesis.

Table 2.4: Example of pre-processing and tokenization of text using 3-gram character tokenization.

Original text	Pre-processed text	3-gram								
		pas	ass	ss_	s_b	_ba	bar	arn	...	lær
Passer på barn	pass barn	1	1	1	1	1	1	1	...	0
Passer på barn i barnehage	pass barn barnehag	1	1	1	1	2	2	2	...	0
Barnehagelærer	barnehaglær	0	0	0	0	0	1	1	...	1

2.2.5. Weighting of terms - TFIDF

A challenge seen in the tokenization process is that common words get a lot of weight. Words such as "jobber" (work) may occur often in occupation descriptions but not contribute to the performance of the classification model. Term frequency-inverse document frequency (TFIDF) is a method to weigh the tokenized data to overcome this problem. The idea of weighting words/terms differently is not new and has been around for at least 25 years [55, 56]. The term frequency part (tf) refers to the number of occurrences of a specific word or n-gram/k-gram (t) within the string or document (d). This is multiplied by the inverse document frequency (idf) which is the inverse of the proportion of documents (d) with that term (t), where n_d is the number of documents/observations in the data.

$$TFIDF(t,d) = tf(t,d) \cdot \frac{n_d}{df(t,d)} \tag{2.1}$$

Variations on this have been developed over the years, but this technique is still commonly used in NLP problems like our classification task. We implemented this in Python using scikit-learn [57]. This uses a variant of TFIDF where

$$idf(t,d) = log\frac{1+n_d}{1+df(t,d)} \tag{2.2}$$

$$TFIDF(t,d) = tf(t,d) \cdot (idf(t,d)+1) \tag{2.3}$$

TFIDF was implemented using the *TfidfTransformer* class in the *sklearn.feature_extraction.text* module. This transformation also simultaneously normalizes the data [58, p. 266]

2.3. DRIFT DETECTION METHODS

In recent years, the number of drift detection methods has exploded [5]. Many of the newer techniques focus on online methods used on big data streams [9], however, data used in NSIs is often small and in batches, as is the case for the data used in this thesis. We therefore explore the use of two methods, both designed for comparing data in batches: Kolmogorov-Smirnov (KS) and RV.

The Kolmogorov-Smirnov (KS) test is a non-parametric technique to compare the distribution of two samples. It is named after Andrej Kolmogorov and Nikolai Smirnov from their work in the 1930s and 1940s [17, 18]. While an older technique, it has gained new interest in recent years in the context of model drift detection [20]. As a non-parametric test, it makes no assumptions about the data distribution but instead tests for whether two samples come from the same underlying distribution. It is a significance test, the Null hypothesis being that the two samples come from the same distribution. The KS test statistic (D) is calculated as the maximum distance between the two empirical cumulative distributions [59]. This is compared to a null hypothesis distance (D^*) calculated as

$$D^*(d) = \left(\sqrt{ne + 0.12 + 0.11/\sqrt{ne}}\right) \cdot d \tag{2.4}$$

where d is the observed value and $ne = \frac{n \cdot m}{n+m}$ where n and m are the sizes of the two samples [59]. The significance level can then be calculated using a function, Q for the probability that the null hypothesis is false:

$$P(D > d) = Q(D^*(d)) \tag{2.5}$$

as defined further by Stephens [60]. The KS test allows us to compare feature distributions and determine if they come from the same underlying distribution, assuming a static, non-drifting environment. A significant result indicates there is drift at that point in time. This method takes a uni-variate approach, so an adjustment is made for multivariate situations. Implementation was done in Python using the alibi-detect package, which uses a Bonferroni correction to adjust the significance level of the test statistic [19]. The Bonferroni adjustment aims to preserve the global type I errors (false positives) to the specified significance level for the tests. It does this by dividing the significance level (eg. $\alpha = 0.05$) by the number of tests, ensuring the significance but in a known conservative manner [61].

We implemented the KS drift detection in two ways: 1) with a fixed reference point for vectorization, using a continuous detection model, and 2) using a moving reference for vectorization and new detection models each year. A significance level of 0.05 was chosen.

The RV coefficient is a multivariate extension of Pearson's correlation coefficient and was first described by Escoufier in 1973 [62]. It is a way to measure the relationship between two matrices for a set of subjects [24]. The RV coefficient is not generally associated with measuring drift patterns, however, we test its use in this novel setting with our time series of textual classification data. An advantage of using the RV coefficient is that it is specifically tailored for multivariate data, as is the case for processing textual data. If $\mathbf{X}$ is a n x p matrix and $\mathbf{Y}$ is a n x q matrix then the RV for the two matrices can be calculated as

$$RV(\mathbf{X}, \mathbf{Y}) = \frac{tr(S_{\mathbf{XY}} S_{\mathbf{YX}})}{\sqrt{tr(S_{\mathbf{XY}}^2 S_{\mathbf{YY}}^2)}} \tag{2.6}$$

where $S_{\mathbf{XX}}$ is the empirical covariance matrix of $\mathbf{X}$ defined as

$$S_{\mathbf{XX}} = (1/(n-1))\mathbf{X}'\mathbf{X} \tag{2.7}$$

($S_{\mathbf{YY}}$) is the empirical covariance matrix of $\mathbf{Y}$ as

$$S_{\mathbf{YY}} = (1/(n-1))\mathbf{Y}'\mathbf{Y} \tag{2.8}$$

and the empirical covariance matrix of $\mathbf{X}$ and $\mathbf{Y}$ ($S_{\mathbf{XY}}$) is

$$S_{\mathbf{XY}} = (1/(n-1))\mathbf{X}'\mathbf{Y} \tag{2.9}$$

The method is often used to compare a set group of subjects with different sets of variables or at different time points. The matrices must have the same number of rows (n). In our case, we are interested in comparing the matrices of a set of words or features. We therefore transpose the data before calculating the RV to attain two data sets with an equal number of rows. In addition, we double-centered the data, meaning we centered the data around the mean, both row-wise and column-wise.

The RV coefficients were calculated by comparing yearly batches of data. Pair-wise comparisons were performed for all years within each occupation. Calculations were performed in Python using the *hoggorm* package [63].

2.4. BASE ALGORITHMS

We selected three machine learning algorithms to test as the base predictive model for occupation: Random forest, Support vector machine and logistic regression. These have been tested on similar classification problems within Statistics Norway [53, 54]. This section provides a summary of their methodologies.

2.4.1. Random forest

Random forest is a supervised machine learning method that has become popular for classification tasks due to its good performance and simplicity [64, p. 587]. They are an example of an ensemble method where multiple decision trees are grown and combined using a voting system for classification. The following paragraphs summarise decision trees and the key concepts of how they have been extended to random forests.

The base structure of a classification decision tree is to recursively split the data into increasingly similar (or pure) groups with respect to the classification variable (y) based on the criteria of one of the features ($\mathbf{x}$). At each split, the most appropriate variable and threshold value is chosen according to a splitting measure. After each selection, the data is then divided again until no further gain can be made, or a stopping criterion is met [65]. The information gain refers to the splitting measure we want to maximize at each split and can be defined as

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \tag{2.10}$$

where D_p is the data at the parent node, D_j is the data at child node j, $I(D_p)$ and $I(D_j)$ are the impurity measures for the parent and child nodes respectively, and N_p and N_j are the number of samples in the parent and child nodes [58, p. 91]. The information gain is essentially the difference between the purity (how similar the samples are) of the parent and the sum of the child nodes. When the split leads to a higher purity of the child nodes the information gain will be higher. The Gini index is often used to measure the impurity of the nodes and is defined as

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{2.11}$$

where p_i is the relative frequency of class i in data D at a particular node [65].

The random forest method utilizes two important additional concepts: bagging, or bootstrap aggregation, and random selection of features [66]. Bagging involves the creation of multiple datasets through a technique called bootstrapping, wherein samples are drawn from the original data with replacement. Results from models trained on the bootstrap samples are then aggregated. This process of bagging has been shown to improve the accuracy of the classifications [67]. In addition to bagging, a random subset of features is chosen for consideration at each node within the random forest trees. This forces less favourable features to be used and increases the variation among the trees. Bagging and the random selection of features work together to create many noisy, overfitted and different trees in random forests, which often leads to higher accuracy classifications with lower variance than single decision trees [66, 64, p. 588].

2.4.2. Support vector machine

Support vector machine (SVM) is a powerful supervised ML method, commonly used in classification tasks. They have previously been shown to perform well in similar text classification tasks at Statistics Norway [54]. While first described in the late 1970s, they gained increased interest in the late 1990s as a tool for solving multidimensional estimation problems [68]. The general idea of SVM classification is to fit optimal hyperplanes to split the data into classes. If the classes are linearly separable, there will be an infinite number of solutions for the position of the hyperplane [64, p. 129]. The optimal location of the decision boundary is thereby determined by maximizing the margin distance between the boundary and the closest observations, known as the support vectors. In the case of binary classification, assume x_i is the feature vector for observation i and y_i is the class variable being 1 for a positive class and -1 for a negative class. The points, x, which lie on the decision boundary will then satisfy

$$w \cdot x + b = 0 \tag{2.12}$$

where w is the weight vector, normal to the decision boundary, and b is the offset from the origin. When classes are linearly separable, all observations x_i satisfy the two conditions

$$w \cdot x_i + b \geq 1 \quad for\ y_i = 1 \tag{2.13}$$

$$w \cdot x_i + b \leq -1 \quad for\ y_i = -1 \tag{2.14}$$

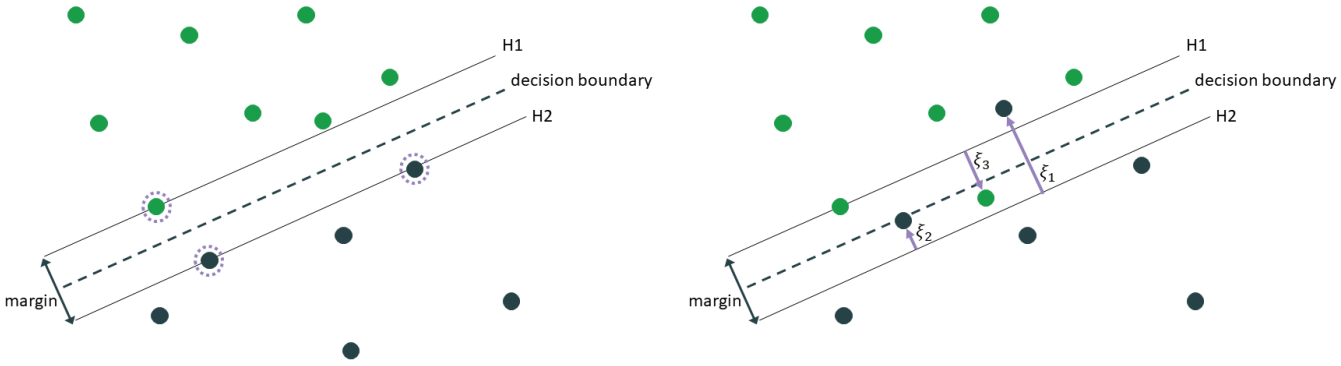
These two equations can be combined to give the overall constraint

$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i \tag{2.15}$$

The points that satisfy the equivalents of equations 2.13 and 2.14, will form the support vectors and lie on the hyperplanes (H_1 and H_2), parallel to the decision boundary [69]. The perpendicular distance between each of the hyperplanes and the decision boundary is $|1-b|/\|w\|$ for the positive class and $|-1-b|/\|w\|$ for the negative class. Together, this distance is called the margin and is $2/\|w\|$ [69]. SVM aims to maximize this distance with constraint shown in equation 2.14. However, in practical terms, it is easier to minimize the reciprocal term, $\|w\|^2/2$ using quadratic programming [58, p. 81]. Figure 2.4 (a) provides a diagram of an SVM with separable classes. Vector supports are shown lying on $H1$ and $H2$ and are circled.

In the more common case that classes are not linearly separable, slack variables (ξ_i) are introduced to allow for some misclassifications. The constraints from equations 2.13 and 2.14 are then

$$w \cdot x_i + b \geq 1 - \xi_i \quad for\ y_i = 1 \tag{2.16}$$



(a) SVM with separable classes. Support vectors are indicated with cirlces

(b) SVM with non-separable classes. Slack variables are shown for the 3 observations that lie on the wrong side of the decision boundary. Slack values for correctly classified observations are set to 0.

Fig. 2.4: Diagrams of SVM decision boundaries for a two-class example. Hyperplanes are shown as H1 and H2.

$$w \cdot x_i + b \leq -1 + \xi_i \quad for\ y_i = -1 \tag{2.17}$$

where

$$\xi_i \geq 0 \quad \forall i \tag{2.18}$$

which includes the slack variables ξ_i for each observation, i [69]. The new objective is now to minimise $\|w\|^2/2 + C(\sum_i \xi_i)$ where C determines the severity of the penalty term for misclassifications [58, p. 82]. Large values of C lead to a smaller margin and can cause overfitting of the model. Therefore, the value of C should be tuned as part of the hyperparameter tuning process.

SVM models were implemented using functions *SVC* and *LinearSVC* in the *sklearn.svm* package in Python. The function *SVC* was used to test a non-linear kernel ('rbf') while the *LinearSVC* was used as a faster implementation of a linear kernel model.

2.4.3. Logistic regression

Logistic regression stemmed from the need to model and predict binary variables using multiple explanatory variables. The method is popular and used in a broad range of contexts such as modeling health outcomes [64, p. 122], credit rating [70], and text classification tasks [53]. Logistic regression is based on estimating the odds of a particular event or belonging to a particular class. The odds can be written as $\frac{p(\mathbf{x})}{1-p(\mathbf{x})}$ where $p(\mathbf{x})$ represents the probability of belonging to class $y = 1$ given the values of $\mathbf{x}$ [58, p. 61]. The logit function is then the natural log of these odds which can be expressed as a linear combination of the values of $\mathbf{x}$, similar to that seen in equation 2.12 as an adapted linear regression:

$$ln\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = w \cdot \mathbf{x} + b \tag{2.19}$$

where w is the weight vector and b is the bias or offset from the origin. This can be seen as applying a logit scale to transform a linear regression and ensuring predicted values lie in the $[0, 1]$ range [71]. The equation above can then be rearranged to give

$$p(\mathbf{x}) = \frac{1}{1 + exp^{-(w \cdot \mathbf{x} + b)}} \tag{2.20}$$

After estimating the weight vector (w) in this model using training data, values for new observations can be predicted ($\hat{y}$). A probability is returned from the model, which using a threshold is turned into an estimated class value. For example, if the threshold is 0.5, then

$$\hat{y_i} = \begin{cases} 1, & \text{if } p(x_i) \geq 0.5. \\ 0, & \text{otherwise.} \end{cases} \tag{2.21}$$

Logistic regression was implemented using the *LogisticRegression* function in the *sklearn.linear_model* Python package.

2.4.4. Multiclass approaches - one vs rest

Many of the methods and cases in the previous sections have referred to binary outcome variables. In our text classification problem, we have a multiclass problem. In this thesis, we focus on predicting observations to one of four classes (in two separate models) but in a real application for occupation classification, there are several hundred codes to predict. We therefore need to adapt the methods to a multiclass outcome.

One vs rest is a common approach in multiclass problems. It entails building separate binary models for each class, each one being modeled against all remaining classes. The models are then compared, with the most confident used for the classification prediction. This approach is used with the algorithms described in this thesis.

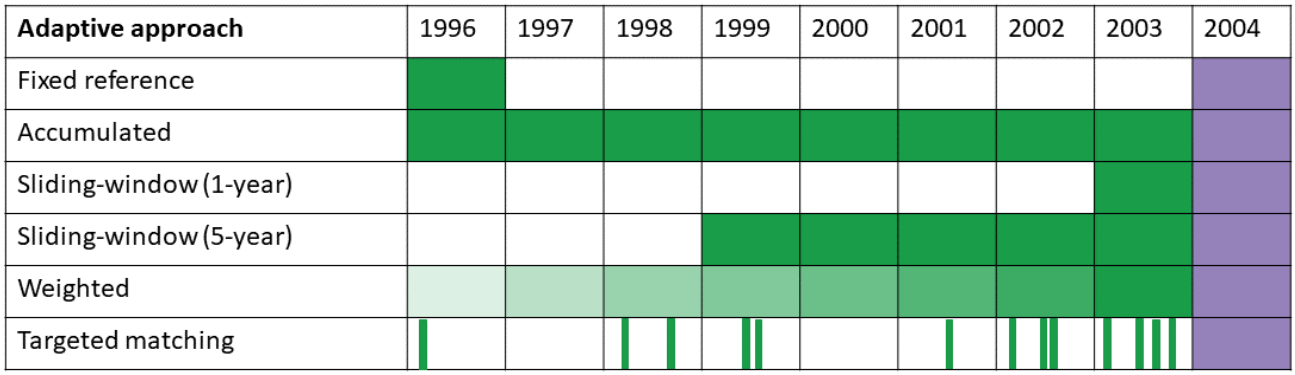
2.5. DRIFT ADAPTION METHODS

In addition to testing for drift patterns in our data, we tested and compared several approaches to adapting the algorithms to changes. Most adaptive approaches focus on which training data is used to learn the models. The most relevant training data is often that which is closest in time to what we want to predict. This must be balanced with the trade-off that more training data often leads to better predictions. This stability-plasticity trade-off is explored by comparing 4 adaptive approaches: sliding-windows, weighting, an adaptive algorithm (Hoeffding adaptive tree), and a matching approach. These are compared to two methods that are not adaptive: using a fixed reference and using all available training data (accumulative).

2.5.1. Sliding-window

Sliding-window approaches are based on a fixed-width size of training data. We used a sliding-window of 5 years and a sliding-window of 1 year prior to the prediction point. For each year where data is predicted, the training data moves to the nearest window. An illustration of this and the other adaptive methods is shown in Table 2.5.

Table 2.5: Example of training data for predicting classification in 2004 using different adaptive approaches. Green indicates training data and purple indicates the test data.



2.5.2. Weighting

An alternative approach to sliding-windows is weighting. We chose to test a global weighting scheme where all data before the prediction time was used but with varying weights. The weights for the training data were calculated as:

$$w_{ij} = \frac{1}{y_0 - y_j} \tag{2.22}$$

where w_{ij} is the weight used for training data i in year j which is inversely proportional to the difference between the prediction year, y_0, and the year of the observation y_j. So training data from the year before prediction would be given a weight of 1, whereas training data from 10 years prior would be given a weight of 0.1. Training observations with higher weights will push the algorithm to focus more on correctly classifying them.

2.5.3. Targeted matching algorithm

Mallick et. al. [4] presented a matching algorithm in 2022 where the data to be predicted is matched to similar training data for the learning process. They presented an algorithm with an online approach to learning with a fast runtime. However, there is currently no open-source implementation of this in R or Python available for testing. We have used a matching approach to build the training data for the learning process to test this concept. We matched features of observations to be predicted with the available training data to build a specific training dataset for each

time point. We used the function *SequenceMatcher* in the Python package *difflib* to match text to 2 observations in the training data to learn the algorithm. Python code for the adapted functions used is provided in Appendix B. *SequenceMatcher* is based on a matching algorithm called the Gestalt approach, described in the 1980s by John Ratcliff and John A. Obershelp [72]. It was developed first to identify matching text that may include spelling errors. Their approach was to compare two strings and first identify the longest common substring, called the anchor. The new strings to the left and right of the anchor are then examined as new strings in a recursive manner. The length of all the matching characters, K_m, is added together across the recursive anchors until all parts of the string have been analyzed. The distance function is then

$$D = \frac{2K_m}{|S_1| + |S_2|} \tag{2.23}$$

where $|S_1|$ and $|S_2|$ are the lengths of strings 1 and 2 that are being compared [73]. The distance values for D range from 0 to 1, where 1 is an exact match and 0 is completely not a match. This algorithm was used to identify the 2 most similar strings to build the training data for training a model.

2.5.4. Hoeffding adaptive trees

Hoeffding adaptive trees were also tested as an adaptive approach. This is an extension of the very fast decision trees (VFDT) where an active change detection (ADWIN) is implemented. This algorithm has shown promising results in drifting environments over other adaptive approaches [33]. The motivation for the Hoeffding tree was to create a decision tree for data streams with large amounts of data. It creates a more efficient decision tree by considering that not all observations are needed to select the feature to use at each of the nodes of the tree. Exactly how many observations are needed is determined by the Hoeffding bound. We first consider a random variable, r, with range R, where for an information gain $R = log(c)$ where c is the number of classes. The Hoeffding bound then states that with probability $1 - \delta$, the true mean of the variable, $\overline{r}$, is at least $\overline{r} - \epsilon$ where

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{2.24}$$

$delta$ is a confidence level and n is the number of observations [31]. The idea is then that the number of samples used n, is as small as possible while still assuring that the feature chosen is the same as that if all samples were used.

Bifet and Galvadà's [33] extension included a change detection mechanism to further limit the number of observations to use. The implementation in this thesis uses the *HoeffdingAdaptiveTreeClassifier* function from the Python package *River* [74]. This includes an ADWIN change detection algorithm which uses a window, W, of the most recent observations, so long as a drift has not been detected. When new data arrives, drift detection is tested by splitting the current window, W, into two parts W_0 and W_1 which have lengths n_0 and n_1 consecutively. The average values of the features ($\mathbf{x}$) in the windows are calculated as μ_{W0} and μ_{W1} and their difference is compared

to a cutoff value ϵ_{cut}, as:

$$\epsilon_{cut} = \sqrt{\frac{2}{m}\sigma_W^2 \ln\frac{2}{\delta'}} + \frac{2}{3m}\ln\frac{2}{\delta'} \tag{2.25}$$

Here m is the harmonic mean of the two windows, $1/(1/n_0 + 1/n1)$, and σ_W^2 is the observed variance of the observations in the window, W [22]. The process of cutting the window is repeated for all lengths of n_0 and n_1. As a consequence, $\delta^{'}$ is an adjusted from the confidence level, δ, and considers multiple testing

$$\delta^{'} = \frac{\delta}{n} \tag{2.26}$$

where n is the sum of n_0 and n_1.

2.6. PERFORMANCE METRICS

To compare models and approaches, a series of performance metrics were calculated including accuracy, F1-score, ROC-Area under curve (AUC), and the Brier score. All these metrics are sensitive to different aspects of a model's performance. They are therefore useful in different ways to measure the various performance aspects. The different metrics are described in the following sections.

2.6.1. Accuracy

Accuracy is one of the most common performance metrics used and is simple to calculate and interpret. Here we define accuracy as the number of correctly classified items in the prediction divided by the total number of predictions. In a binary case, a 2x2 confusion matrix is established as in Table 2.6. The number of correctly identified observations in the positive class is referred to as the number of true positives (TP), while the number of true negatives (TN) is the number of correctly classified observations in the negative class. False negatives (FN) refer to incorrectly classified observations, predicted to the negative class and False positives (FP) are those incorrectly classified to the positive class.

Table 2.6: Confusion matrix terms

		Prediction	
		Positive	Negative
Observation	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True negative (TN)

Accuracy it is then

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.27}$$

Accuracy works well as a performance metric for classification tasks where data is balanced, and there are similar numbers of observations in all the classes. However, in cases of high imbalance, the results can be misleading. For

example, if one class represents 95 percent of the data, predicting all to this class will result in an accuracy of 95 percent which could be misinterpreted as good performance.

2.6.2. F1-score

The F1-score addresses some of the weaknesses of accuracy by balancing both precision and recall. The precision is the number of true positives (TP), divided by the sum of the TP and false positives (FP)

$$Precision = \frac{TP}{TP + FP} \tag{2.28}$$

The precision measures how well the model correctly identifies the positive class. A high precision means that when the model predicts a positive result, it is likely to be correct. Table 2.6 provides the confusion matrix terms for a binary classification example.

In contrast, recall measures the ability of a model to correctly identify all relevant instances of a class. It is defined as the TP divided by the sum of the TP and the number of false negatives (FN).

$$Recall = \frac{TP}{TP + FN} \tag{2.29}$$

Recall is important in situations where false negatives are costly and/or have significant consequences.

The F1-score is then a combination of these two measures and is defined as

$$\text{F1-score} = 2\frac{precision \cdot recall}{precision + recall} \tag{2.30}$$

This is a useful performance measure in many circumstances, particularly when working with unbalanced data. For multi-class problems, like the one in this thesis, the F1-score is calculated for each class and then averaged over the classes. We have chosen to use the weighted average which weighs the class F1-scores by the number of observations.

2.6.3. ROC-AUC

The Receiver Operating Characteristic (ROC) is a way to assess the balance between recall, sometimes called the True Positive Rate (TPR), and the false positive rate (FPR). In binary cases, the FPR is the probability of an incorrect positive value when the observation is negative.

$$FPR = \frac{TP}{FP + TN} \tag{2.31}$$

The ROC is then a curve of the TPR (recall) against the FPR as the threshold of the classification rule is varied [64, p. 317]. In the binary case, a model such as logistic regression will calculate a float value between 0 and 1 for each observation it predicts. A threshold value is then applied to determine which class the observation should be predicted to. For example, a value of 0.5 may be applied where predicted values above this are predicted as positive and values (1) below this are predicted as negative (0). To create the ROC curve, this threshold value is varied, and the TPR and FPR are re-calculated. The values of TPR and FPR are then plotted on a graph to create a curve (see Figure 2.5). Curves that are a straight diagonal line indicate a model that is not able to discern the categories, whereas a curve that increases rapidly up to a high value in the top left corner indicates a model that is better at distinguishing the categories correctly.

The area under the ROC curve is used as a single metric to assess the model performance. Values close to 1 indicate a ROC curve that is near the top left corner. Values near 0.5 indicate ROC curves that are around the diagonal. Figure reffig:roc provides some example ROC curves with their equivalent AUC values.

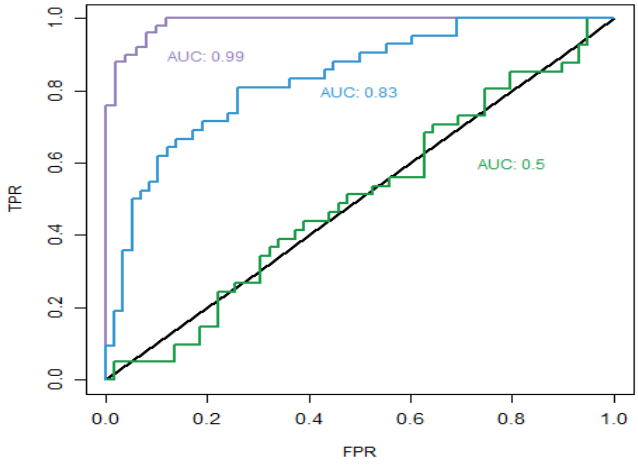


Fig. 2.5: Example of three ROC curves with corresponding AUC values. The diagonal line provides the baseline for a model that doesn't discern between classes.

2.6.4. Brier score

The Brier score is a statistical measure to assess the accuracy of probabilistic predictions in the context of classification tasks. It was originally used to assess weather predictions and was first described in 1950 [75]. The original model allowed for multiclass predictions and was defined as

$$BrierScore = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{R}(f_{ij} - o_{ij})^2 \tag{2.32}$$

where N is the number of observations, R is the number of classes, f_{ij} is the probabilities associated with the prediction of observation, i, in class, j, and o_{ij} takes the value of 0 or 1 according to whether the event occurred in class j or not. The score uses probabilities rather than predicted classes and therefore considers how certain the model is in its predictions. The score is essentially the mean square error of the prediction probabilities. Consequently, lower values of the Brier score, near 0, indicate a high performance, whereas high values, near 1, indicate the model is not able to distinguish the classes well.

2.7. MODEL TUNING

We want to test how different approaches to learning models in a drift situation perform. To do this, we need a base model for testing different window sizes and weighting approaches. An important part of using ML algorithms is tuning the hyperparameters of the model. We tested 3 base models (regression, random forest and SVM) with the hyperparameters shown in Table 2.7. In addition, we tuned a number of hyperparameters in the Hoeffding adaptive tree model.

Table 2.7: Hyperparameters and values used in the model tuning process.

Model	Parameter	Values
SVM	C	0.01, 0.1, 1, 10, 100
	Maximum features	500, 1000, 5000
	Use TFIDF	True, False
	Kernel	linear, rbf
	Terms	word, 3-gram (character)
Logistic regression	C	0.1, 1, 10
	Maximum features	500, 1000, 5000
	Use TFIDF	True, False
	Terms	word, 3-gram (character)
Random Forest	Tree number	50, 100, 500
	Maximum features	500, 1000, 5000
	Use TFIDF	True, False
	Terms	word, 3-gram (character)
Hoeffding Adaptive Tree	Grace period	10, 200, 500
	Delta	1e-07, 1e-03, 0.1, 0.9
	Terms	word, 3-gram (character)

A 25 percent sample of our data was used to tune the hyperparameters. This is much smaller than a standard approach which often uses 70 to 80 percent of the data for tuning and training. hyperparameters were tuned using all years in the full tuning data, the remaining test set was run using a year-wise approach to compare the various adaptive approaches. It was important to have a large amount of data in the test data to ensure a reasonable representation of all classes for all years. The training/test split was stratified by year and occupation class to ensure that all groups were represented in each year. We used 5-fold cross-validation in the tuning data to determine the best values

for the hyperparameters using accuracy as the performance metric. The process of model tuning was run separately to predict among the 4 different occupations (Group A) and the 4 similar occupations (Group B).

Chapter 3

Results

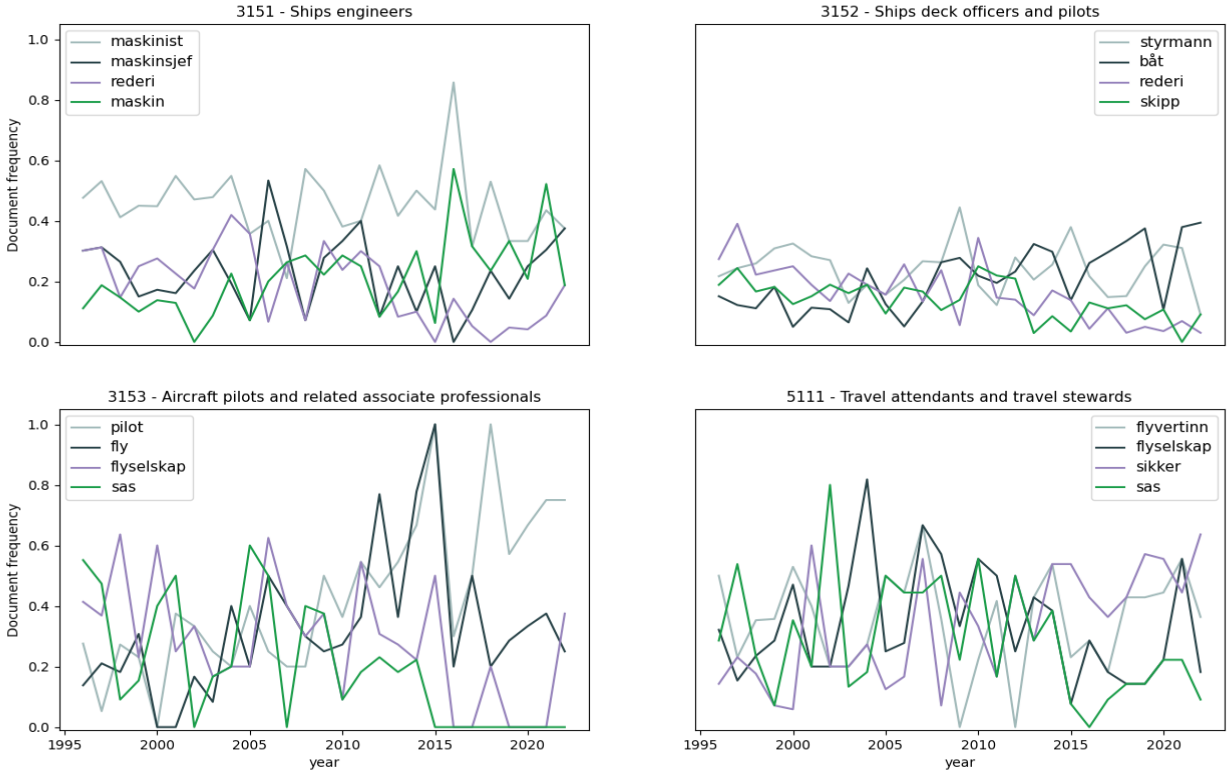
This chapter describes descriptive results, drift detection and model performance results from our analysis. Word popularity, trends and frequency over the time period are described in section 3.1, while results from RV computations and KS drift detection are described in section 3.2. A summary of model tuning and comparative results for the adaptive models is given in section 3.3.

3.1. DESCRIPTIVE RESULTS

We focus on 8 occupation classes in our data that are used to build two separate, predictive models. The text variables in the data, described in section 2.1 were pre-processed as described in section 2.2. Figure 3.1 shows the term frequency of the 4 most common words within each of the 8 occupation classes. The levels shown indicate the proportion of observations where that term occurs in the text. For example, the stemmed word ”barnehag” (kindergarten) occurs in approx 70 to 80 percent of the texts for Early childhood teachers. We see that in some classes, for example, production clerks, the popularity of the top 4 words has been relatively stable over the whole time period. The term ”logistikk” (logistics) has occurred in around 60 percent of the texts/observations throughout the time period. In other classes, we can see clear changes in the popular terms. For example, in early childhood teachers, the term ”førskolelær” (pre-school teacher) has declined from occurring in around 50 percent of texts down to around 5 percent. In contrast, ”pedagogisk” (educational) and ”led” (leader), which are often used together (”pedagogisk leder”) have increased in frequency over this time period within this same class. Among aircraft pilots, the term ”pilot” has increased in popularity to a frequency of around 80 percent of texts within this class. This is a borrowed English word that we see has increased in use within the Norwegian language. In contrast, Figure 3.2 indicates that the more traditional Norwegian words for a pilot: ”flyver” and ”flyger”, have decreased in popularity in recent years.

Among the most common words, we see no overlap in Group A. However, in Group B where the occupations are more similar, there is some overlap in the most common words. For example, we see the word ”rederi” (shipping

(a) Group A



(b) Group B

Fig. 3.1: Popularity of the top four words in each occupation class.

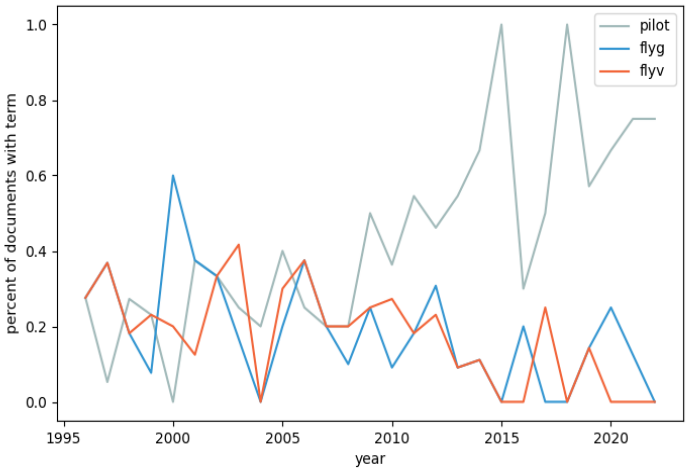


Fig. 3.2: Occurrence of ”pilot”, ”flyg” and ”flyv” terms within the occupation class for aircraft pilots.

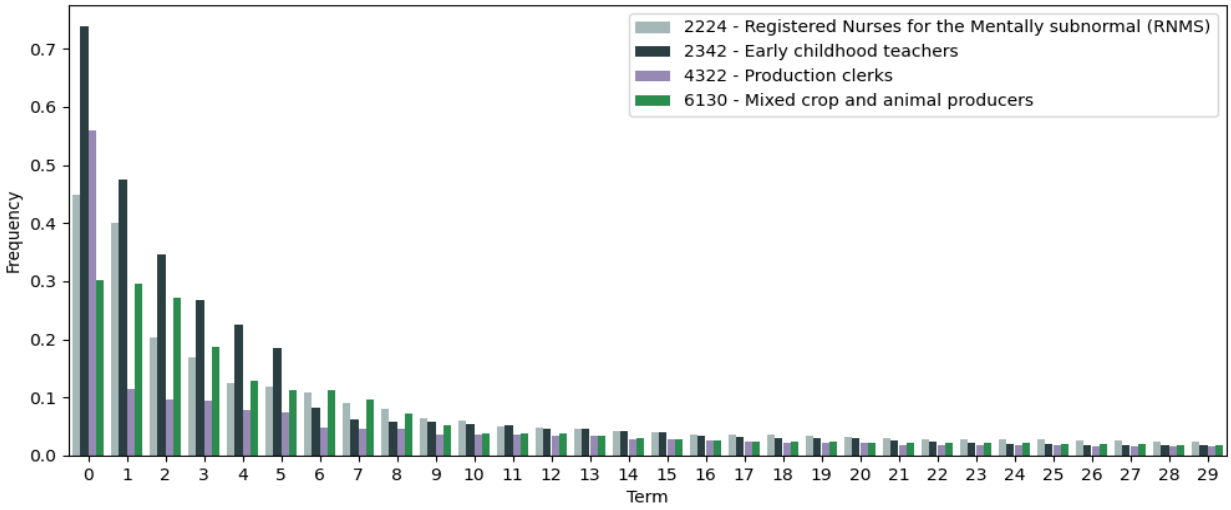
company) in both occupations 3151 (Ships engineers) and 3152 (Ships decks officers and pilots). The words ”sas” (SAS: airline company) and ”flyselskap” (airline) are both common in classes 3153 (aircraft pilots) and 5111 (travel agents and stewards).

The word frequencies in Figure 3.1 for Groups A and B are all shown on the same scale. We see that some occupation classes, for example, early childhood teachers, have words with a very high frequency, 80 percent or more in some years. Other classes, for example, crop and animal producers, have lower frequencies for their most common words.

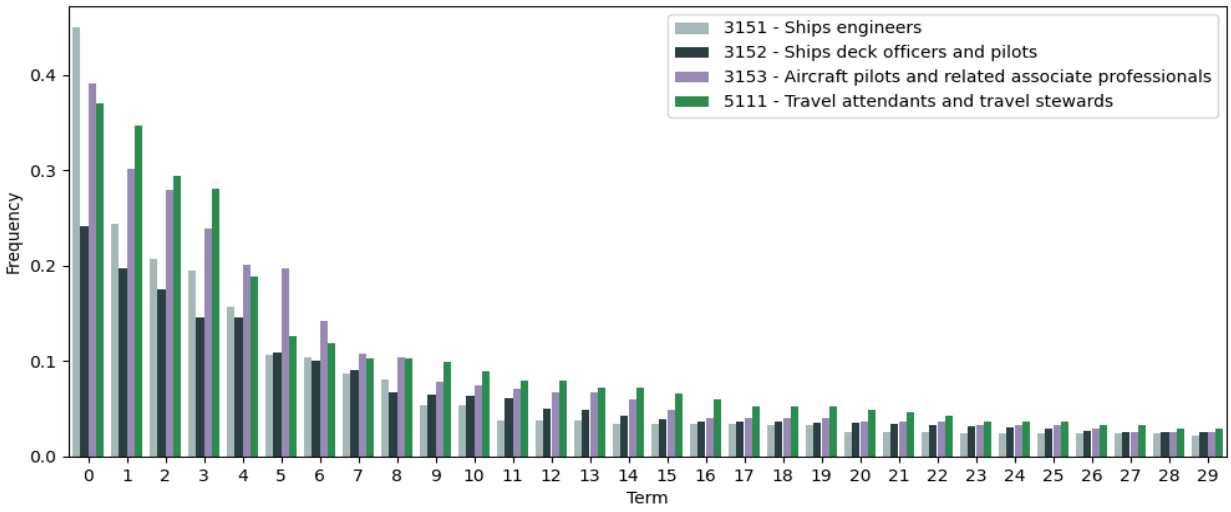
Figure 3.3 shows the percentage of observations that contain the 30 most frequent word terms within occupation classes in Groups A and B. The plots show frequencies irrespective of the actual word but ordered by their frequency. This emphasizes that for some classes, specific terms are important and the text is perhaps more homogeneous. Other classes with lower frequencies among their most common terms indicate a higher variation in how the occupation is described and perhaps also a higher variation in the tasks and jobs performed by people within these classes. In Group A, we see that early childhood teachers (2342) have the most frequent term for the most common word. However, when we look at the 30th most common word, the RNMS occupation has the most frequent term. In Group B, ship engineers (3151) had the most highest frequency of their most common term while travel attendants and stewards (5111) had the highest frequency of the 30th term.

3.2. DRIFT DETECTION

The textual data used in our analysis is from a 27-year period. Figures 3.1 and 3.2 indicate that common terms used to describe jobs have changed over this period within some occupations. We are interested to see if these changes



(a) Group A



(b) Group B

Fig. 3.3: Comparison of the percentage of occurrence of terms in each occupation using stemmed word terms

are detected and able to be visualized using drift detection methods. This section provides results from two methods; 1) using a quantitative Kolmogorov-Smirnov (KS) drift detection model and 2) using RV comparisons together with heatmaps to visualize which classes may be contributing to drift.

KS drift detection was run separately on the two groups of data (Group A and Group B) with results shown in Table 3.1. The results compare matrices based on 3-gram character features. We found that drift was only detected in the model with more diverse occupations (Group A). When the reference point for the vectorization was fixed to the first period and the drift detection model was allowed to run consistently, drift was detected at all time points from 2001, except for 2003. In contrast, when the drift detection only included the year prior to testing, only 2021 showed a drift pattern within Group A. For Group B, no drift was detected in either of the methods.

Table 3.1: Results from KS drift detection using both fixed and moving models. Results are from comparing matrices using 3-gram character features. Models from groups A and B are shown separately.

Group	Reference point	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
A	Fixed	0	0	0	0	1	1	0	1	1	1	1	1	1
	Moving	0	0	0	0	0	0	0	0	0	0	0	0	0
B	Fixed	0	0	0	0	0	0	0	0	0	0	0	0	0
	Moving	0	0	0	0	0	0	0	0	0	0	0	0	0
Group	Reference point	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
A	Fixed	1	1	1	1	1	1	1	1	1	1	1	1	1
	Moving	0	0	0	0	0	0	0	0	0	0	0	1	0
B	Fixed	0	0	0	0	0	0	0	0	0	0	0	0	0
	Moving	0	0	0	0	0	0	0	0	0	0	0	0	0

Results from KS drift detection indicate that for occupations within Group A, there have been some significant changes to the underlying textual data. To explore this further and gain a better understanding of which occupations were contributing to these changes, we calculated RV values for all year-wise comparisons. Figures 3.4 and 3.5 provide heatmap plots of the RV values for the 8 occupation classes, again using matrices based on 3-gram character features. The number of observations and corpus size for each year within the classes are also shown.

We see there are quite different patterns of RV among the occupations, particularly within Group A occupations. Early childhood teachers have the highest correlation in features over the time period. This indicates that these jobs are generally described in similar ways by many of the participants. However, the number of features (corpus size) in this class is quite high, around 1000 each year, and the number of observations is reasonably high, compared to other classes and consistent throughout the years. There is some indication of a drift pattern when comparing the most extreme years within this occupation. From around 2015, the correlation with earlier years appears to decrease as indicated by a lighter green colour. RV values for RNMS show also some signs of drift, where particularly the early years 1996 to 2002 and recent years 2015 to 2022 show higher RV values with the closer years. The corpus

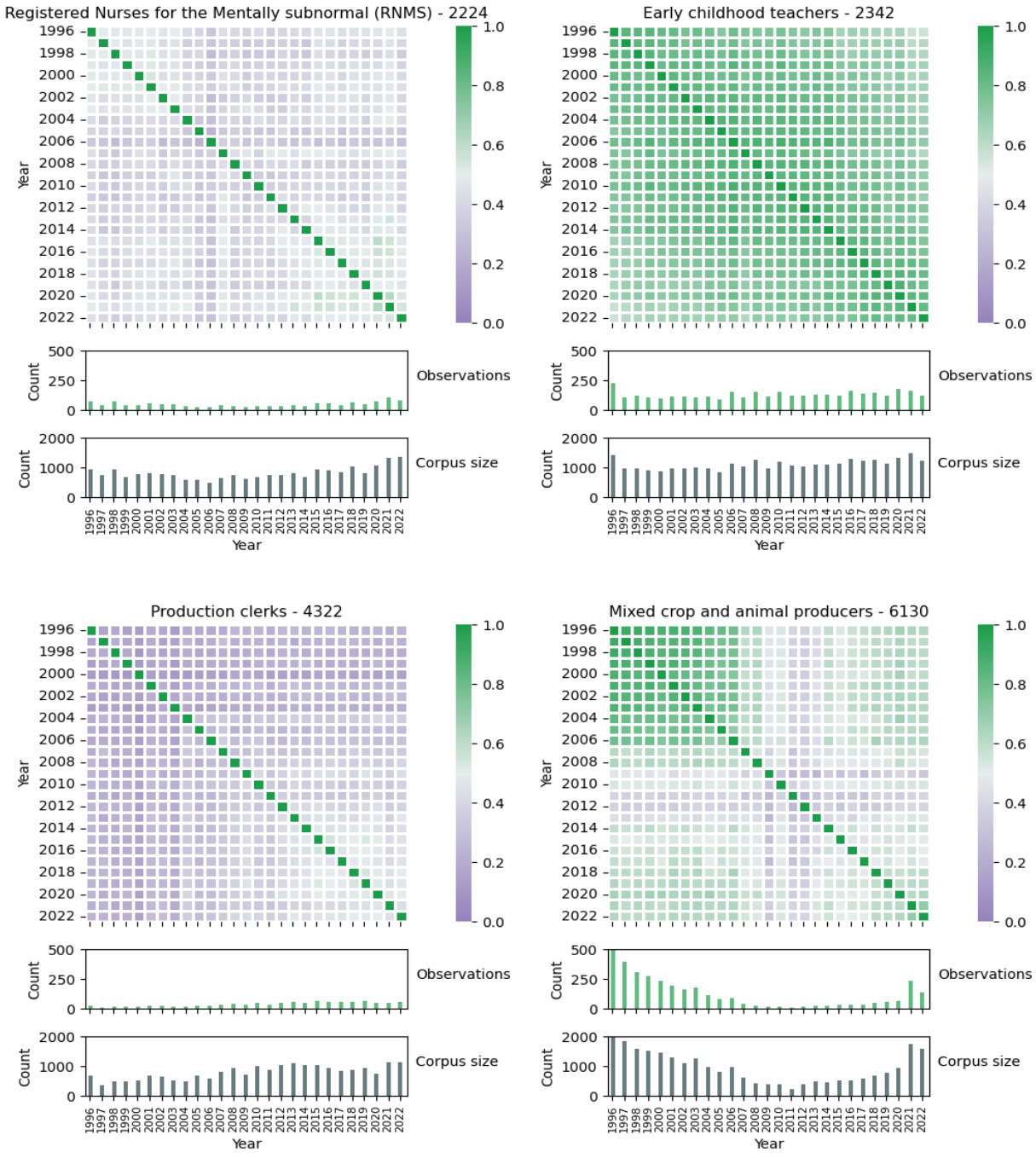


Fig. 3.4: Heatmaps showing the RV values for each comparative year for 4 occupation classes in Group A. Barplots below show the number of observations and corpus size.

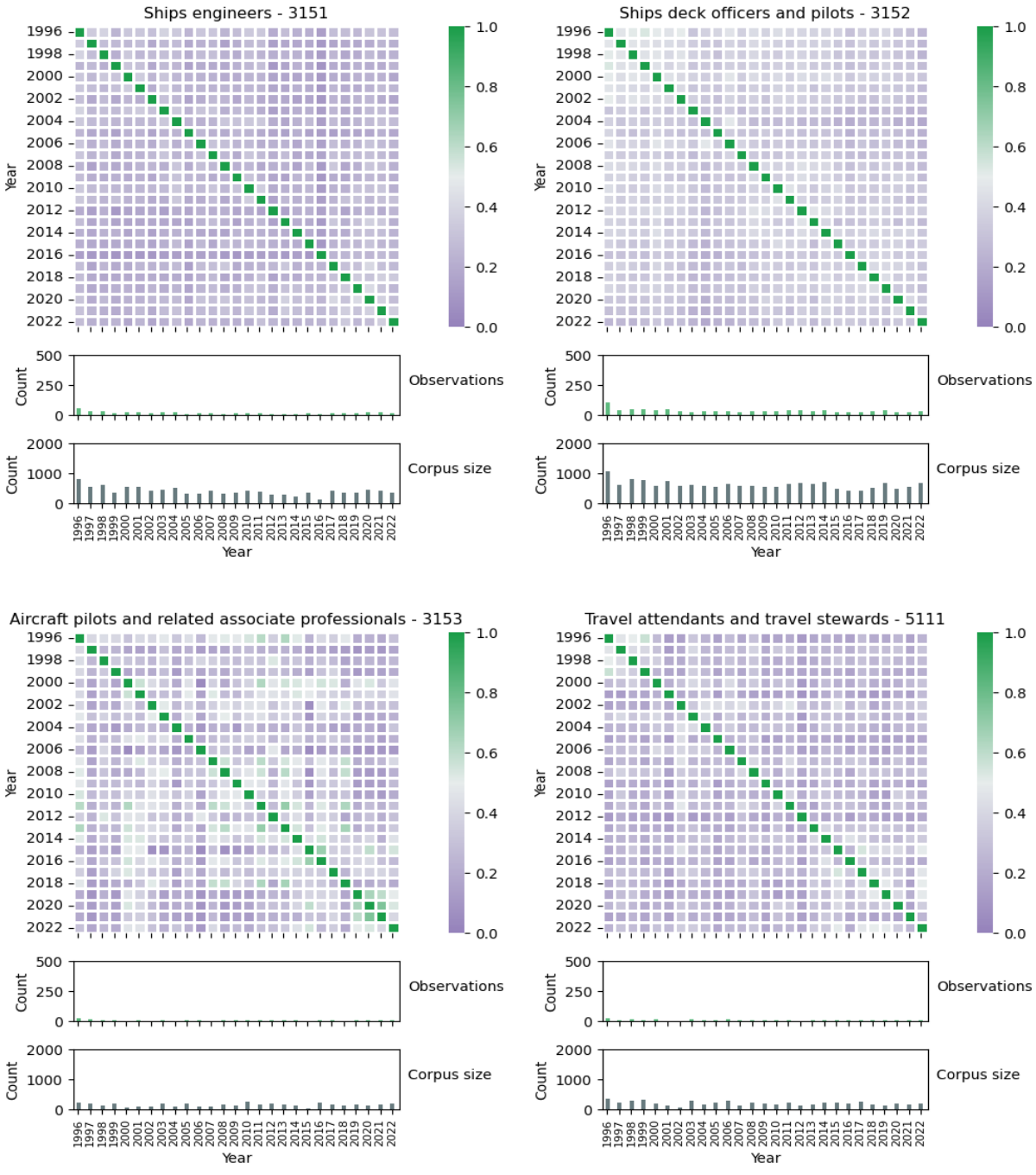


Fig. 3.5: Heatmaps showing the RV values for each comparative year for 4 occupation classes in Group B. Barplots below show the number of observations and corpus size.

size is slightly smaller and the observation numbers are lower compared to early childhood teachers. Within Group A, Production clerks have the lowest RV values overall. They have the fewest observations among those in Group A. The corpus sizes are smaller in the earlier years, before 2010, while similar to the other classes in the later years. This along with the small number of observations, indicates it is heterogeneous, where occupations are described in varying, different ways. While Figure 3.1 indicates there is one common term ”logistikk” within this class, it also shows the next most common words do not have a particularly high frequency. Again, this supports a low RV value and indicates diverse occupation descriptions. RV values for the occupation class on mixed crop and animal producers show signs of a recurrent pattern. From around 2008 to 2014 there was a change resulting in much lower RV values during this time period. From around 2015, occupation descriptions then reverted back to similar patterns to those prior to 2008. Both the observation numbers and corpus size dropped greatly during this period.

Among the occupations in Group B, RV values were generally low. Ships deck officers and pilots show a slight pattern of drift where RV values appear higher with closer years in the earlier period, 1996 to 2001, compared to later years. Travel attendants and stewards also showed some small signs of drift in the very early years with higher RV values in years 1996 to 1999. The remaining years show lower RV values and no great changes or patterns. RV values for aircraft pilots appear to be quite randomly spread with some higher values in years far apart, and some among years close together. However, there does appear to be a pattern of higher RV values among the most recent four years, 2019 to 2022 within this class. The final occupation in Group B; ships engineers, had low overall RV values with few discernible patterns, other than that in 2016 there appears to be a year of data that is less correlated with all other years than otherwise. The number of observations and corpus size in Group B is generally less than those in Group A. Ship deck officers and pilots (3152) had the most observations of those in Group B while aircraft pilots (3153) had the least.

Both KS drift detection and RV heatmaps indicate some changes have occurred in occupation description among the Group A occupations. Group B occupations, while more similar to each other in how they are described, do not show strong drift patterns from either the KS drift detection or visually using RV values.

3.3. PREDICTIVE PERFORMANCE OF THE MODEL

We are interested in building predictive models for classifying occupations that will work under both stable and drift conditions. This section describes the results from models tested for classifying occupations within Group A (4 very different occupations) and Group B (4 similar occupations). We first present results from model tuning, followed by results relating to model degradation, adaptive approaches and the new matching approach.

3.3.1. Model validation

Model hyperparameters were tuned using a 25 percent training set and 5-fold cross-validation. Data was amalgamated over all years, using standard base algorithms without adaptive approaches. Table 3.2 provides results from

the best models for each of the algorithms based on accuracy performance measures. Table 2.2 shows that while there are some differences in the number of observations in each class they are reasonably balanced, therefore, accuracy is an appropriate measure to use. For a full table of model tuning results see Appendix C.

SVM models performed best when using TFIDF feature weighting, a linear kernel, and 3-gram character features. The SVM for Group A performed best when the penalty value, C, was 10 whereas Group B performed equally best when C was 1 or 10. For further testing in the remianing part of this thesis we use the smaller value, 1, for C in Group B. Using a maximum of 1000 or 5000 features in the model for Group A performed equally, whereas, for Group B, 5000 and 10000 maximum features performed equally. Again, we chose the smaller values for further analyses, using 1000 features for Group A and 5000 in Group B.

For logistic regression we saw that using TFIDF feature weighting with 3-gram character features also performed best in both Group A and B. The best value for the strength of the regularization, C, was 10 for Group A and B. The model performance was best when using 5000 or 10000 features for Group A whereas Group B performed best with only 500 features.

Random forests also performed best when using 3-gram character features, however, TFIDF weighting was only prefered in Group A. 500 trees was prefered in Group B, whereas only 100 trees in Group A provided the best validation performance. The best number of features was 5000 in Group A and 500 in Group B.

Tuning of the Hoeffding adaptive trees resulted in the best model with a grace period of 1, a delta value of 0.1 and using words as features. TFIDF was used in all cases and wasn't a tuned hyperparameter in this model.

3.3.2. Model selection

SVM, Random forests and logistic regression models are not drift adaptive approaches by themselves. Results from cross-validation in section 3.3.1 were used to compare their performances to select one as a base model for testing adaptive approaches.

Table 3.2 shows that SVM performed best among Group A and equally best with logistic regression for Group B. For simplicity in programming, an SVM was selected to use for further comparisons of adaptive designs for both groups. Overall, Random forests performed slightly worse compared to the other base models: SVM and logistic regression. Logistic regression performed well in Group B and only very slightly worse than SVM in Group A.

The Hoeffding adaptive tree model is an adaptive approach where branches may be replaced based on a drift detector when their accuracy decreases. Table 3.2 shows that the validation performance (63 and 64 percent accuracy) is considerably lower than that of the other methods. The approach appears to be considerably underfitting the model and unable to utilise the features in a good way. We have therefore chosen not to include it further in the analysis.

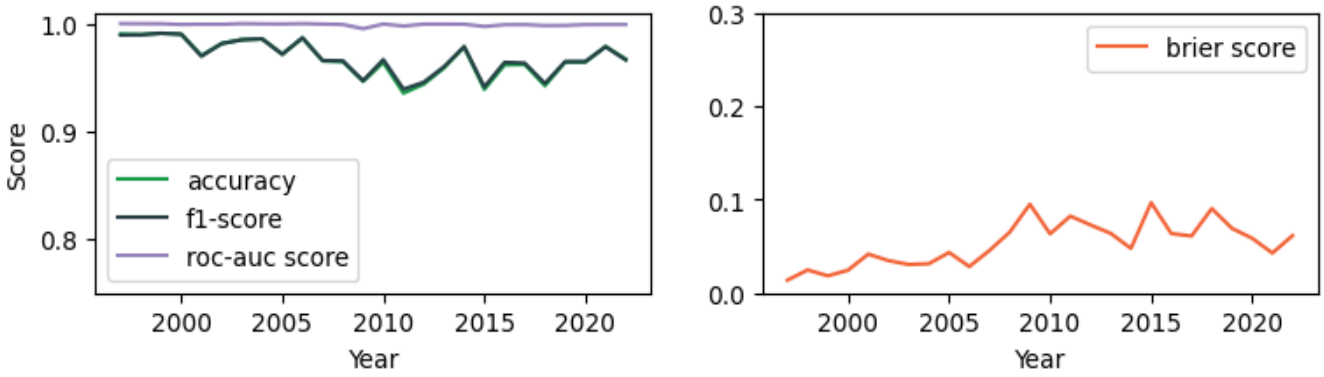
Table 3.2: Best model hyperparameters for SVM, Logistic regression, Random forest and Hoeffding adaptive tree classification models for occupations in Groups A and B.

Group	Model	Validation performance (average accuracy over 5-folds)	Best hyperparameters
Group A	SVM	0.9931	C: 10, max features: 1000, use TFIDF: True, kernel: linear, features: 3-gram
		0.9931	C: 10, max features: 5000, use TFIDF: True, kernel: linear, features: 3-gram
	Logistic regression	0.9907	C: 10, max features: 5000, use TFIDF: True, features: 3-gram
		0.9907	C: 10, max features: 10000, use TFIDF: True, features: 3-gram
	Random forest	0.9842	max features: 5000, number of trees: 100, use TFIDF: True, features: 3-gram
	Hoeffding adaptive tree	0.6347	grace period: 1, delta: 0.1, features: word
Group B	SVM	0.9567	C: 1, max features: 5000, use TFIDF: True, kernel: linear, features: 3-gram
		0.9567	C: 1, max features: 10000, use TFIDF: True, kernel: linear, features: 3-gram
		0.9567	C: 10, max features: 5000, use TFIDF: True, kernel: linear, features: 3-gram
		0.9567	C: 10, max features: 10000, use TFIDF: True, kernel: linear, features: 3-gram
	Logistic regression	0.9567	C: 10, max features: 500, use TFIDF: True, features: 3-gram
	Random forest	0.9459	max features: 500, number of trees: 500, use TFIDF: False, features: 3-gram
	Hoeffding adaptive tree	0.6410	grace period: 1, delta: 0.001, features: word

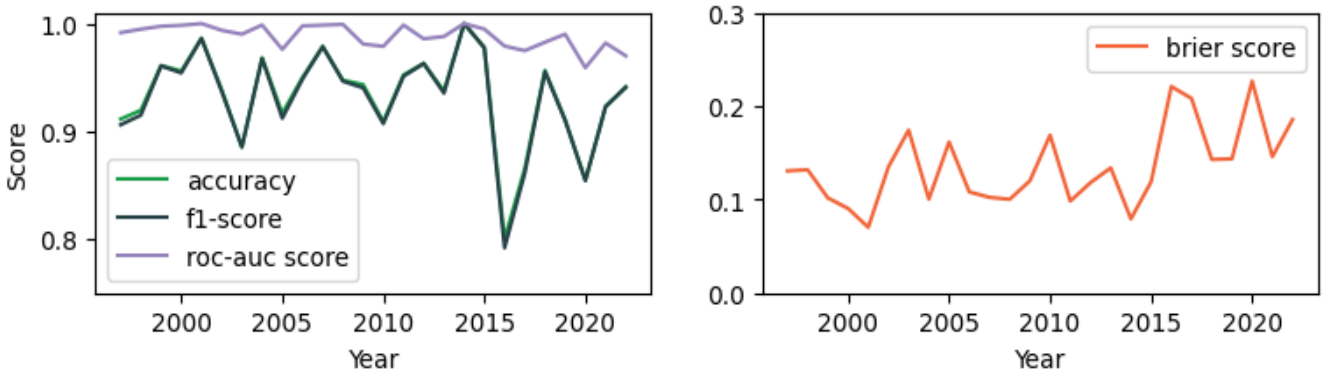
3.3.3. Model degradation

Model degradation refers to the decrease in model performance over time; see Figure 1.3. We are interested in investigating whether this is the case in our two classification models (Groups A and B) over the 27-year time period. Section 3.2 indicates that features have changed significantly for Group A over this time period and descriptive results shown in section 3.1, indicate that at least some features have changed their distributions over this time period for both Groups A and B.

To test model degradation, a fixed reference point (year 1996) was used to train an SVM model, based on the best hyperparameters from model tuning shown in 3.3.1. Occupation classifications were then predicted for all years after this. This represents a scenario where a model is trained and put into a production setting with no retraining. Data used in this, and the following sections includes the 75 percent not used in model tuning. Figure 3.6 shows results using four performance metrics for the two groups of data. Accuracy and F1-scores are quite high, above



(a) Group A



(b) Group B

Fig. 3.6: Performance metrics by year for a classification model using a fixed reference period (1996) for training showing some model degradation. Metrics are shown separately for Group A and B.

90 percent for all time periods in Group A and above 80 percent for all time periods in Group B. In Group A, there appears to be a slight decline in the accuracy and F1-scores, particularly between 2005 and 2010. Group B shows a wider variation in the accuracy and F1-scores and is in general lower performance compared to that seen in Group A. It shows a slight decrease in values over the time period. We also note that the values for the accuracy and F1-scores are very similar to each other. The data in this study is relatively balanced and indicates that these are measuring performance in a similar way.

Figure 3.6 shows that the ROC-AUC measures are high (near 1.0) in both Group A and B. This indicates that the model is quite certain about the classifications it is producing. In Group A, there is no indication of model degradation based on the ROC-AUC performance measure. Group B shows some signs of a decrease in performance, particularly from 2015 onwards, based on the ROC-AUC.

The Brier scores also show some indication of model degradation for both Group A and B. Values closer to zero for the Brier score indicate a more certain and better-performing model. Again, Group B has a larger variation in the values, but both show an upward trend, towards increased uncertainty in the model.

Overall, there is some indication that if an occupation classification model was trained on a single time period and put into production without retraining, the performance would decrease over time.

3.3.4. Adaptive prediction models

Adaptive approaches were tested based on SVM classification models to adapt to drift in the data. Again, hyperparameters established from tuning (see section 3.3.1) were used in the models. Table 3.3 provides the average results for all years of the adaptive approaches, in addition to a fixed reference point as in section 3.3.3, and a non-adaptive accumulative approach, using all data available prior to the testing time point. It shows that the accumulative approach performed, on average, best, based on all performance measures we tested. For the ROC-AUC performance measure, the 5-year sliding window and weighted approaches performed equally well in Group A compared to the accumulative approach.

Table 3.3: Average performance metrics over all years (1997 to 2022) for adaptive, fixed and accumulative approaches. Metrics are given separately for Groups A and B.

Group	Method	Accuracy	F1-score	ROC	Brier score
Group A	Fixed	0.9684	0.9691	0.9991	0.0528
	1-year window	0.9815	0.9813	0.9994	0.0500
	5-year window	0.9866	0.9865	**0.9997**	0.0240
	Accumulated	**0.9878**	**0.9878**	**0.9997**	**0.0193**
	Weighted	0.9875	0.9874	**0.9997**	0.0275
	Matched	0.9851	0.9849	0.9996	0.0349
Group B	Fixed	0.9328	0.9311	0.9885	0.1354
	1-year window	0.8829	0.8726	0.9787	0.2468
	5-year window	0.9536	0.9524	0.9909	0.0928
	Accumulated	**0.9600**	**0.9593**	**0.9939**	**0.0685**
	Weighted	0.9550	0.9538	0.9934	0.1060
	Matched	0.9372	0.9354	0.9892	0.1325

To investigate how well the model was performing for each of the occupation classes, the F1-scores were calculated within occupations and are shown in Table 3.4. We can see that while the accumulative approach performed best in most situations overall, it was not always the best method based on the individual classes. The novel matching approach performed best in occupation class 6130 (mixed crop and animal producers). Note that in this occupation, the RV scores in Figure 3.4 showed a recurrent drift pattern. A weighted approach performed slightly better than the other approaches in occupation for early childhood teachers (2342) and for RNMS (2224). Interestingly, these are two classes that showed some slow drift patterns in Figure 3.4 heatmaps. Within Group B occupations, the accumulative approach performed best in all occupation classes.

Figure 3.7 shows the F1-scores throughout the 27-year period. A 3-year running average was used to smooth the values to visualize trends. Results for all performance metrics by year are shown in Appendix D. In general, accuracy, F1-scores and Brier scores show a similar pattern to each other. The ROC-AUC scores were generally

Table 3.4: Average F1-scores over all years (1997 to 2022) for adaptive, fixed and accumulative approaches. Metrics are given separately for each occupation class in Groups A and B.

Group	Method	2224	2342	4322	6130
	Fixed	0.9667	0.9878	0.9289	0.9153
	1-year window	0.9627	0.9871	0.9565	0.9819
Group	5-year window	0.9759	0.9908	0.9628	0.9881
A	Accumulated	0.9783	0.9914	**0.9672**	0.9852
	Weighted	**0.9810**	**0.9923**	0.9601	0.9848
	Matched	0.9714	0.9873	0.9605	**0.9925**
		3151	3152	3153	5111
	Fixed	0.9334	0.9462	0.9222	0.8800
	1-year window	0.8868	0.9295	0.6850	0.7843
Group	5-year window	0.9477	0.9584	0.9361	0.9516
B	Accumulated	**0.9533**	**0.9653**	**0.9502**	**0.9555**
	Weighted	0.9514	0.9602	0.9486	0.9475
	Matched	0.9368	0.9522	0.9056	0.8980

higher and with less variation than the other measures. F1-score was chosen for Figure 3.7 as a good representative performance measure for investigating individual occupation trends.

In Group A, we see that all models performed well for the early childhood teachers (2342) throughout the time period. For the RNMS occupation (2224), the 1-year window showed some variation, performing not as well as the other in some of the time periods, particularly between 2006 and 2009. For production clerks, we see an increase in F1-scores for all approaches in the early years from 1997 to 2000. Based on the observation and corpus sizes shown in Figure 3.4 we can observe that there were very few observations in 1997 which may be contributing to the low F1-scores in the beginning. One incorrectly predicted observation will have a large impact on the overall F1-score for the class in this case. We see an increase in all approaches in this class between 1997 and 2000, indicating it is not just the increased amount of data in the training set improving the model (the fixed reference model also increased over this time). The fixed reference period model has a decrease in performance from around 2005. The RV values for this class appear to increase in the later years, perhaps indicating some drift (Figure 3.4). The mixed crop and animal producers (6130) occupation shows a drop in the F1-scores for many of the approaches between 2010 and 2018. Comparing this with the RV values shown in Figure 3.4, we see that this occurs at around the same time as the number of observations drops and the RV values also decrease. The matching approach appears to have the highest F1-scores during this time of change.

In Group B, the F1-scores appear to be more variable over the time period compared to Group A. In all groups, the 1-year window approach performed worse the the others at most time periods. This perhaps reflects the small number of observations and instability in the model when only a single year is used for the training. The accumulative, weighted, and 5-year window methods all appear to improve in model performance over the early years; 1997 to 2000. These approaches use increasing amounts of training data and indicate that using only 1 year of training data

is under-fitting the model. The fixed reference period approach did not perform better than the other approaches, however, does not appear to drop substantially in any of the occupation classes in B. This indicates that not much drift is occurring in the data. Finally, the F1-scores for the matching approach are not better than the other approaches over the time period in Group B.

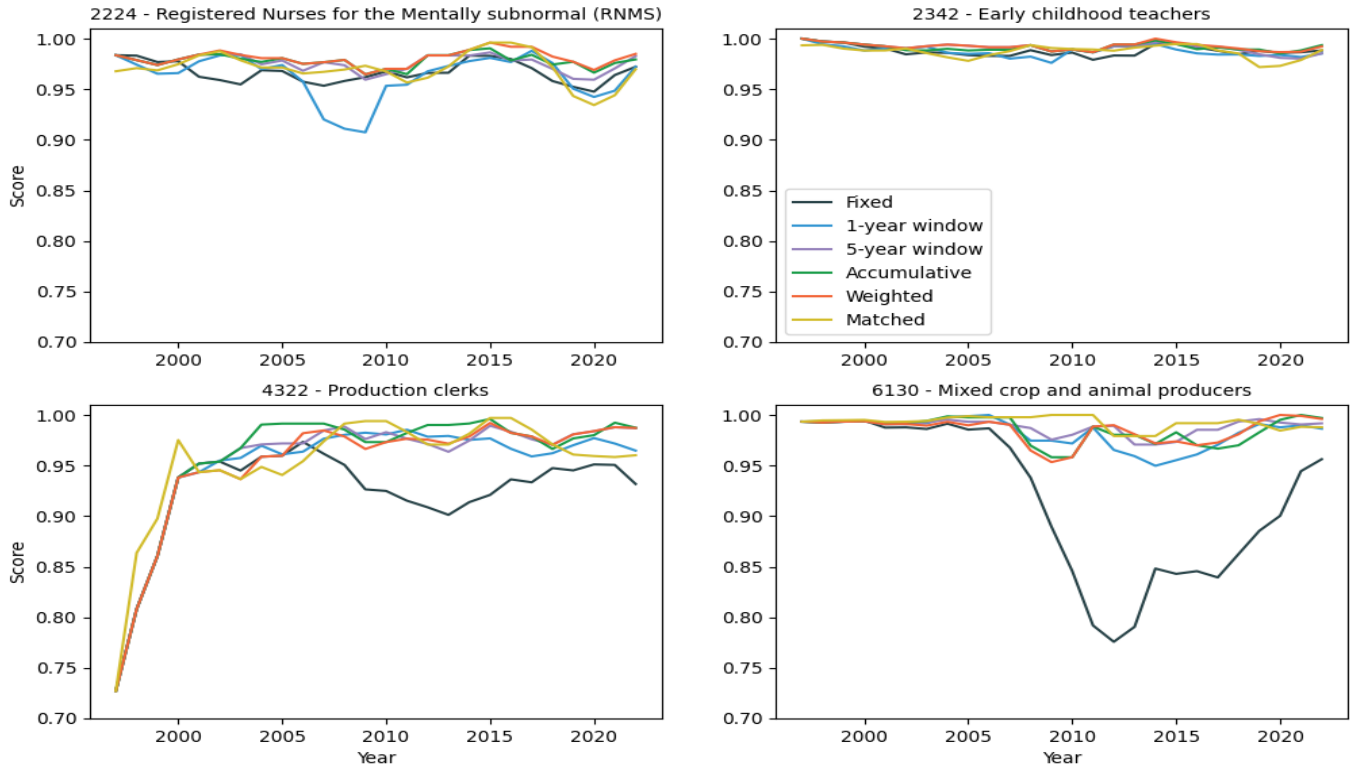
Confusion matrices are shown in Figure 3.8, for the best-performing model using an accumulative approach. We see that the model can classify the occupations in both Group A and B with a high degree of accuracy. Group B shows slightly higher levels of misclassification compared to Group A, but still above 90 accuracy within all of the occupation classes. Among Group A, production clerks had the highest levels of misclassifications with around 2 percent being classified as mixed crop and animal producers. Among Group B, occupation 3153 (aircraft pilots and related associate professionals) was misclassified as 3152 (ships deck officers and pilots) in around 7 percent of predictions.

Average run times for the prediction models are shown in Table 3.5. Run times were averaged over 10 repetitions of the modeling process, except for our novel matching algorithm which was run once. This was only run once due to the very long processing time. We see that run times for Group A are approximately double those for Group B. This is mainly because of the difference in the number of observations in the groups (9843 observations compared to 2216 observations). The run times using the full accumulated training data and with weighting were similar and the slowest of the official package algorithms. The new matching algorithm was the slowest by a large margin due to the slowness of the matching procedure for establishing the training data.

Table 3.5: Average run times for adaptive and standard modeling for Groups A and B. Timings are in seconds and are the average of 10 runs except for the matched algorithm which shows results for 1 run.

Run time	Group	
	A	B
Fixed Reference	3.8425	1.9046
1-year window	2.5827	1.5367
5-year window	4.7346	2.2643
Accumulated	10.3552	3.9014
Weighted	10.1811	3.6160
Matched	15675.61	890.62

Finally, we are interested in how our models and approaches will scale up to larger amounts of data and classes. Table 3.6 shows performance results for the prediction approaches using combined data with 8 occupation classes (Group A and B combined). The targeted matching approach was dropped here due to the extensive time it took to run the algorithm. All other approaches were tested.



(a) Group A

(b) Group B

Fig. 3.7: F1-scores by year and occupation for Group A and B. F1-scores are smoothed using a 3-year running average.

(a) Group A

(b) Group B

Fig. 3.8: Cumulative confusion matrix for the accumulative approach to training data.

Table 3.6: Performance metrics for a prediction model for occupation with 8 classes.

	Accuracy	F1-score	ROC	Brier score
Fixed	0.9694	0.9693	0.9989	0.0497
1-year window	0.9630	0.9618	0.9984	0.0824
5-year window	0.9787	0.9786	0.9992	0.0374
Accumulated	0.9857	0.9857	0.9996	0.0238
Weighting	0.9802	0.9801	0.9995	0.0349

Chapter 4

Discussion

In this thesis, we have explored model drift detection and adaption techniques in a text classification example. We have investigated occupation classification data from the Norwegian Labour Force Survey (NLFS) and seen indications of underlying changes in occupation descriptions. Both slow drift and recurrent drift patterns were observed in some occupations, while others showed no signs of change. A decrease in the performance of the classification models was seen in some cases. Both standard adaptive algorithms and a new targeted matching approach were tested on two classification models. We saw that several of the approaches were able to adapt to the effects of changes seen in our data, however, a simple accumulative method worked best overall.

This discussion chapter summarises the results from our analysis in the context of our research questions, provides an outline of the limitations to the scope of our findings, and provides context to the results with ideas for future directions.

4.1. SUMMARY OF RESULTS

This study has investigated four objectives, described as research questions in section 1.5. These are summarised in the following sections.

4.1.1. Is drift occurring in the chosen occupation groups?

The first objective was to investigate whether drift was occurring in our 8 chosen occupation classes. We first used descriptive techniques to look at the frequency of the most common words and visualized this using Figures of the frequencies through time (Figures 3.1 and 3.2). Some changes were observed, for example, the frequency of ”førskolelær” decreased while ”pedagogisk” increased in the early childhood teacher occupation. Among descriptions in the aircraft pilot occupation class (3153) the occurrence of ”flyg” and ”flyv” decreased while ”pilot” increased. These both indicate that some drift is occurring in the features of our data. In other classes, such as

production clerks (4322), the word frequencies for the most common words have remained relatively stable over the 27-year period and do not support that drift is occurring.

We used Kolomogornov-Smirnov (KS) drift detection as a more formal test for the presence of drift. This is a method implemented in several Python packages and an established technique for use in this setting. Using a fixed reference period of 1996, a significant difference was seen in most years after 2001 for Group A. This is interesting as 2001 was the first year where occupation was compulsory for companies to report in the *AA-register*, a data source available to those performing the classifications at an individual level. This supports the idea of upstream drift at this time point. Results from comparing features only with the previous year indicated only 2020 and 2021 were different in Group A. Again, there were changes to the register data available to coders in 2021 and again supports the idea of upstream drift at this time point. In Group B, no significant drift was detected using the KS method.

RV values calculated and plotted using heatmaps allowed us to investigate changes within the 8 chosen occupations (see Figures 3.4 and 3.5). Signs of drift patterns were seen within some of the classes. For example, mixed crop and animal producers (6130) showed a recurrent drift pattern where quite sudden changes occurred in around 2009 and then reverted back in around 2014. Gradual drift can be seen in the occupation for early childhood teachers (2342) and to some extent RNMS (2224) shown by weaker RV values at time points further from each other. Among the Group B occupations, we found little evidence of drift using KS or RV values. One challenge within Group B was the lower observation values as seen in Figures 3.4 and 3.5. The RV values appear to be lower when the observation numbers are lower. This may limit its value in drift detection settings. This may only be useful as a drift indicator with sufficiently large data. How much data is needed should be determined experimentally, and is an area of future study.

4.1.2. How does drift impact the performance of the occupation classification models?

We saw some evidence of drift in the description of occupations in Group A. Classification models were tuned for Groups A and B separately and model performance was assessed over the 27-year period in different ways. Using a fixed reference point (1996) we saw that a model trained only on this appeared to degrade slightly according to F1-score, accuracy and Brier score measures. The degradation was only slight though, around 5 percent in Group A over the time period for accuracy and F1-scores (see Figure 3.6). While we did not detect a drift using KS among the Group B occupations, there appeared to be some model degradation, with all performance metrics appearing to decrease over the time period. Again, the amount of the performance decrease was only slight; 5 to 10 percent. Considering the long time period (27 years), the decrease in model performance is perhaps less than we expected. Using only 1 year of data, the model accuracy was still over 90 percent in Group A, 27 years later. This is perhaps a reflection on how different the four occupation classes are in Group A, and how different their features are from one another. Even with more similar occupations in Group B, the model only appeared to suffer slightly and was still able to distinguish the classes, most of the time.

4.1.3. How well do standard adaptive approaches including fixed-window, weighting, and Hoeffding adaptive tree, mitigate drift in our occupation classification problem?

We tested several standard algorithms and approaches to learning under drift, with overall results shown in Table 3.3. All approaches tested performed better than using a fixed reference period in Group A. Using a 1-year window performed slightly worse than a fixed reference point for Group B. This group had fewer observations and using 1 year of training data (either with a fixed reference point, 1996, or the most recent period) appears to be too little training data and is likely under-fitting the model. Using a 5-year window improved the performance, but not as much as using all data up until the test point, referred to as an accumulative approach. We saw that this accumulative approach worked best overall, in both Groups A and B, but did not necessarily perform best when we saw individual occupation results (see Table 3.4). Among Group A results, weighting the training data appeared to improve the performance for two of the occupations (RNMS and early childhood teachers). This is interesting as these were the two occupations where a slow drift pattern was observed based on RV values (see Figure 3.4). This indicates that weighting the training data may be the best approach when there is known slow drift in the data.

The high performance of the accumulative approach in Group B, overall and in all occupations (see Table 3.4) supports the idea that not much drift has occurred in this group over the time period. In stable environments, more training data will generally lead to a better-performing model by providing more information on the underlying processes. The accumulative approach utilizes the most training data with no time consideration.

The Hoeffding adaptive tree algorithm was not shown in the results tables as tuning showed accuracy scores well below the levels seen by the other base methods (Table 3.2). While we tuned some of the main hyperparameters including grace period, and delta value, we were not able to find a way to increase the performance to a level where it could compete against the other approaches. Further tuning and testing could lead to a better-performing model, however, due to time limitations in this thesis, was not tested further.

4.1.4. Can targeted matching create better training data to mitigate drift effects in occupation classification models compared to standard approaches?

A targeted matching algorithm was implemented to create training data similar to the observations to be predicted. We used a matching approach based on 3-gram character features and used an SVM model for prediction. Overall, the targeted matching didn't perform as well as accumulative, weighted, and 5-year window methods (see Table 3.3). The method used training data which was twice the size of the prediction data (2 similar observations were found per observation to predict). This was perhaps too little data and increasing this amount may have perhaps led to better overall results.

Interestingly, when looking at the individual class performances (Table 3.4), the targeted matching performed slightly better than all other methods for the occupation class mixed crop and animal producers (6130). This class showed an interesting pattern where the occupation descriptions changed and/or observation numbers dropped in the

middle of the time period (see Figure 3.4). It produced a recurrent pattern where the change reverted back to a similar pattern to that seen earlier in the time period. It is easy to see that in this case, specifically targeting the training data to find similar observations in an earlier time period could be beneficial. There is therefore some indication that for data with recurrent drift, targeting matching could be a useful approach.

4.2. STUDY LIMITATIONS

In this thesis, we have studied drift patterns in two small classification models for predicting occupation codes from text data. The two sets of data represent a case study, that has allowed us to look in more detail at the relationships of the features, model classification patterns and misclassifications. The groups were chosen to represent two settings, one with occupations that were quite different from one another and likely to give a model that can easily distinguish between the classes, and a case where the occupations were more similar, and where a model might struggle more. The occupations in the case study were chosen from the 48 occupations where we had a full-time series for the years 1996 to 2022. While we chose the groups for the case study that were more frequent, there was still a limited number of observations. The smallest class (aircraft pilots) had only 268 observations over the time period which equates to only around 10 observations per year on average. This limitation can make the models and predictions somewhat limited in performance, even more so as we split the data for model tuning and testing.

Using only 4 occupations in each of the classification models is also a limitation in this study. In a real production setting, there are 407 occupation codes in the STYRK-08 standard for a model to predict. Our approaches in this thesis are therefore limited in scope and would need to be tested and validated further in the wider context of classification. Our approach using RV coefficients and heatmaps to visualize drift would not be feasible in a production setting with 407 occupations, without adaption. One possible extension to our approach would be to combine methods, using a drift detection algorithm first and combining with RV and heatmap plots to gain more insight into what type of drift is occurring.

The text data was processed in a number of ways: removing numbers, stop words and stemming. We used standard, out-of-the-box tools for this, including pre-made lists of stop words. These were partially adapted, by excluding one word we believed could be useful for classification in our context (”dykke”- dive) and adding one that we saw frequently (”as” - company). However, we did not analyze the effect of removing these stop words or whether better adaptions could be used. Likewise, we used a stemming algorithm that was written for a more general setting. We saw that it reduced the number of features in our data (see Figure 2.3), however, it also reduced words to incorrect roots, in the case of ”lærer” (teacher). While this is not necessarily a problem for a prediction model, further investigations would be interesting to look into the effects of this. We saw that stemming didn’t help against spelling errors where, for example, ”gårdbruk” and ”gårdsbruk” (farming) were identified as two separate features. This may be one of the reasons why the model using 3-character grams performed better as it is more robust against these errors. We only tested a very limited range of feature extractions (word and character 3-grams) and further

testing of this would be interesting to optimize the model further. Additionally, we know that some respondents reply to the NLFS in English. While we didn't see many examples in our data, pre-processing including stop words and stemming assumed text was Norwegian. Again, further analysis of this would be interesting to determine if a multi-language approach would be beneficial to the prediction models.

We tested drift detection using Kolmogorov-Smirnov (KS) tests, however, there are limitations to this approach. Its use of a Bonferroni adjustment for multiple testing of features makes it a conservative approach and may be prone to incorrectly missing detection points (prone to type II errors). We used a default setting for the confidence level (p-value = 0.05) but it would be interesting to test this further. Alternative approaches for drift detection would have been interesting to compare with, such as the ADWIN algorithm. Due to difficulties in installing valid packages and time constraints, it was not included in the comparisons. This is a popular algorithm for both detection and adaptive modelling and it would be good for future work to include this approach.

Model tuning was performed on a subset of data without consideration of time or model adaption type. This is a limitation of our study as when running the drift adaption models, much smaller subsets of training data were used. While we used a small set of data (only 25 percent) to reflect some aspects of the smaller data sizes, the lack of time considerations limits it. Hyperparameter values may have been optimized differently compared to if we had considered time and adaptive aspects during the tuning. For example, we found that using a maximum of 5000 and 1000 features for the models was optimal, but when running models based on yearly data, there may not even be this amount of features. Running model tuning by splitting the data into time segments, for example, 5-year groups, and predicting the next time period could be a way to improve the model tuning in this setting. Additionally, considering the adaptive method and further tuning of different options for the weighting approach may have improved the performance of the models.

4.3. CONTEXT OF THE RESULTS AND FUTURE WORK

In National Statistical Institutes (NSIs), there is often time and resources spent on implementing new methods such as ML classification models. Less focus is placed on monitoring ML models that are put into production, risking decreased quality with model degradation over time. The 2021 UNECE report on ML for official statistics [2] acknowledges the need for monitoring after deploying, however, nothing beyond retraining is mentioned as a solution. This may be costly and unnecessary, depending on the performance consequences of the drift. Lu et. al. [5] divide learning under drift into 3 parts: detection, understanding and adaption. We have used a holistic approach, exploring all three tasks associated with model drift in a small case study. This is the first time this has been done in an NSI setting that we know of. Our work therefore provides a guide to methods that can be used in classification tasks within Statistics Norway and other NSIs.

Our study has shown that simple monitoring using drift detection, such as KS tests, is possible to implement in the current IT platform at Statistics Norway. While this is a standard package algorithm, there is a lack of examples

of the performance of drift detection in multi-class text classification [14]. Our study has shown an example of this with results indicating detection at time points where changes have occurred in upstream processes. This may indicate that this is a good method to use for detecting sudden, yet perhaps small changes. Slow gradual changes are often more difficult to detect. Using a set reference point to compare against showed that the KS method may be able to detect gradual changes when they become big enough. However, using this method by comparing each consecutive year with the previous is not likely to pick up on gradual changes over longer periods of time.

We tested using RV values in the context of model drift as a new method for detecting and explaining drift. It has the advantage over many other methods in that it is well suited to multi-variate data. This is particularly important in the context of text classification, where there are often many features. We calculated values within classes, giving us a method to further explain drift patterns. The understanding part of drift mitigation is often overlooked, with most focus in the literature on detection and adaption [9, 12, 10]. Using RV values within classes, we were able to better understand drift dynamics, specifically where and when drift was occurring. However, it doesn't provide us with how extensive the model will be affected. The heatmaps we used to visualize RV values showed quite significant changes in the feature data (see Figure 3.4). This only appeared to reflect a small amount on the model performances (see Figure 3.6). It also seems to be quite sensitive to observation numbers; fewer observations generally resulted in low correlations in the data years. Further testing and development of this tool in drift problems is needed to explore this further.

We have shown that using RV values is a viable new way of helping understand drift patterns. However, we see that the levels of the RV, vary greatly. A low correlation didn't necessarily mean that drift was occurring as it was often consistent over the whole time period (see Figure 3.5). It would therefore be difficult to use it as a drift detection tool, as a fixed threshold value can't be set for all scenarios. Additionally, a challenge in using RV values for drift monitoring is how to scale up to a larger number of classes. We have chosen to look in detail at eight occupations. Scaling this method up to 407 occupations using visual inspection would not be feasible in a production setting. In this case, perhaps subsets of classes could be run using more traditional drift detection methods first, followed by calculating RV values within classes if a drift is detected, to better understand the mechanics behind it. We have also not shown a direct link between drops in RV values and drops in model performance. The KS test showed the first drift detection in 2001 in Group A (Table 3.1) which can also be seen reflected in a slight performance drop in the fixed reference model (3.6). The heatmap however doesn't show any discernible pattern at this time point (3.4). Similarly to the KS method, they can not distinguish between virtual drift and real drift. Rather than monitoring the input data distributions, an alternative approach to drift detection is to monitor the probabilities/certainties that are produced by the models [2]. This may help to avoid false virtual drift detection where the model is still performing well. The idea is to avoid picking up on isolated changes to the feature distributions ($p(\mathbf{x})$) but changes to how certain the model is. This may have a closer link to real concept drift and would be an interesting extension to test in the future.

When a system is stable with little change, models generally benefit from increased training data. Whereas in dynamic environments, there is a benefit to a forgetting mechanism to adapt to changes. This trade-off, described by Grossberg [26] as the stability-plasticity trade-off is important in the context of adapting to drift. In this thesis, we saw an example where stability was the most important aspect (Group B), where more data, irrespective of time, produced the best model performance. We also observed an example where adaptive approaches may be useful (based on class performance scores in Group A). While an accumulative approach performed best overall, within occupation classes where drift was noted, adaptive approaches that had a forgetting mechanism performed better.

Our novel targeted matching approach wasn't the best performer but showed promising signs in our occupation class with a recurrent pattern. The approach we took was inspired by Mallick et. al.'s [4] *Matchmaker* algorithm, however, we implemented a more targeted approach, matching at an observation level. In contrast, the *Matchmaker* algorithm matches the most similar batch based on covariate and concept drift rankings [4]. Their implementation, while less targeted, allows for fast and online implementation with good performance outcomes under drift conditions. Classification tasks at NSIs are often not at the same large scale as in other problems and computational time is often less of an issue based on smaller data sets. However, our implementation of targeted matching was very slow and would require speeding up for this to be a feasible option to use in the future.

Generally, our classification models performed well and were easily able to discern the occupation classes. While some earlier pilot studies using ML for occupation classification have been performed at Statistics Norway, this is the first to use a more systematic approach, using cross-validation for hyperparameter tuning and comparing several methods. Performance measures including accuracy and F1-scores were high for both groups of data (above 95 percent) and show promise for future implementation. While we have investigated the problem as a multi-class problem, future work should focus on the effects of scaling up the problem to all (407) classes. Our initial test of combining both Group A and B into an 8-class problem did not appear to reduce the performance much. However, including further classes will introduce additional challenges such as dealing with classes with very few observations. It is important to note that the goal of a full classification model is not necessarily to achieve the level of accuracy and performance seen in this case study. A double-coding study of occupation in 2018 and 2019 showed that two separate manual coders only achieved 75 percent identical coding at the highest detail level (4 digits) [76]. This indicates that even if a model only achieves 75 percent accuracy, it still may be a useful tool for implementation. Furthermore, Muller [53], outlined the implementation of such predictive models in a semi-assisted way using a Human-in-the-loop framework. Adaptive algorithms could also be implemented in this way, using manual coding for difficult classifications or in periods where there is known drift occurring.

New methods for adaptive learning are being developed at high rates. While we have mostly only investigated shallow, more traditional base ML models, adaptive approaches based on deep Neural Networks are being developed. For example, Xu and Wang's dynamic extreme learning machine uses an adaptive approach where the number of nodes and hidden layers is adaptively changed under drift conditions [77]. There has also been attention given

to transfer learning in drift settings for both drift detection [78] and adaptive classification [79] settings. Transfer learning builds on the idea that a rich data source must be adapted to a new problem, which is very similar to the problem of model drift. These developments are exciting extensions that would be interesting to test in future studies.

Chapter 5

Conclusion

In this thesis, we have shown drift occurring in a text classification problem using occupation classification as a case study. Using a holistic approach, detecting, understanding and adaptive methods were tested to investigate drift patterns. We saw that some drift had occurred during the time period within some of the occupation classes. However, taking a simple approach and using all data available appeared to perform well, above more complex adaptive algorithms in this setting. This emphasizes that despite drift being detected, extensive, adaptive approaches may not be necessary in all contexts. We found some evidence for weighting adaption algorithms performing better within classes where gradual drift appeared to be occurring. Additionally, we showed that creating matched training data appeared to perform well in the class where there was a local recurrent pattern. Further work on extending the models beyond the limited 8 classes used in this thesis will provide a more extensive and realistic picture for future implementation. NSIs, such as Statistics Norway, have an extensive amount of rich data, often with long time series. With increased automation of classification tasks using ML methods, monitoring of model performance should be implemented. Adaption methods, like the ones tested in this thesis, should be tested and considered further in cases where drift (both upstream and conceptual) is known or detected.

References

[1] European Statistical System. European Statistics Code of Practice. Luxembourg: European Union; 2018.

[2] UNECE. Machine Learning for Official Statistics. Geneva: United Nations; 2021.

[3] Yu E, Song Y, Zhang G, Lu J. Learn-to-adapt: Concept drift adaptation for hybrid multiple streams. Neurocomputing. 2022;496:121-30.

[4] Mallick A, Hsieh K, Arzani B, Joshi G. Matchmaker: Data drift mitigation in machine learning for large-scale systems. Proceedings of Machine Learning and Systems. 2022;4:77-94.

[5] Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G. Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering. 2018;31(12):2346-63.

[6] Gemaque RN, Costa AFJ, Giusti R, dos Santos EM. An overview of unsupervised drift detection methods. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2020 11;10.

[7] Storkey AJ. When Training and Test Sets are Different: Characterising Learning Transfer. In: Quiñonero-Candela J, Sugiyama M, Schwaighofer A, Lawrence ND, editors. Dataset shift in machine learning. Mit Press; 2008. p. 3-28.

[8] Becker A, Becker J. Dataset shift assessment measures in monitoring predictive models. Procedia Computer Science. 2021;192:3391-402.

[9] Bayram F, Ahmed BS, Kassler A. From concept drift to model degradation: An overview on performance-aware drift detectors. Knowledge-Based Systems. 2022:108632.

[10] Ditzler G, Roveri M, Alippi C, Polikar R. Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine. 2015;10(4):12-25.

[11] Moreno-Torres JG, Raeder T, Alaiz-Rodríguez R, Chawla NV, Herrera F. A unifying view on dataset shift in classification. Pattern Recognition. 2012;45:521-30.

[12] Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A. A survey on concept drift adaptation. ACM computing surveys (CSUR). 2014;46(4):1-37.

[13] Feldhans R, Wilke A, Heindorf S, Shaker MH, Hammer B, Ngonga Ngomo AC, et al. Drift detection in text data with document embeddings. In: Intelligent Data Engineering and Automated Learning–IDEAL 2021: 22nd International Conference, IDEAL 2021, Manchester, UK, November 25–27, 2021, Proceedings 22. Springer; 2021. p. 107-18.

[14] Barros RSM, Santos SGTC. A large-scale comparison of concept drift detectors. Information Sciences. 2018 7;451-452:348-70.

[15] Sarnelle J, Sanchez A, Capo R, Haas J, Polikar R. Quantifying the limited and gradual concept drift assumption. In: 2015 International Joint Conference on Neural Networks (IJCNN). IEEE; 2015. p. 1-8.

[16] Elwell R, Polikar R. Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks. 2011;22(10):1517-31.

[17] Kolmogorov A. Sulla determinazione empirica di una legge di distribuzione. Giornale dell'Istituto Italiano degli Attuari. 1933;(4):83-91.

[18] Smirnov N. Table for Estimating the Goodness of Fit of Empirical Distributions. The Annals of Mathematical Statistics. 1948;19(2):279 281. Available from: https://doi.org/10.1214/aoms/1177730256.

[19] Van Looveren A, Klaise J, Vacanti G, Cobb O, Scillitoe A, Samoilescu R, et al.. Alibi Detect: Algorithms for outlier, adversarial and drift detection (version 0.10.3); 2019. Accessed: 2022-08-17. https://github.com/SeldonIO/alibi-detect.

[20] dos Reis DM, Flach P, Matwin S, Batista G. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 1545-54.

[21] Taplin R, Hunt C. The population accuracy index: A new measure of population stability for model monitoring. Risks. 2019;7(2):53.

[22] Bifet A, Gavalda R. Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. SIAM; 2007. p. 443-8.

[23] Bu L, Alippi C, Zhao D. A pdf-free change detection test based on density difference estimation. IEEE transactions on neural networks and learning systems. 2016;29(2):324-34.

[24] Josse J, Pagès J, Husson F. Testing the significance of the RV coefficient. Computational Statistics & Data Analysis. 2008;53(1):82-91.

[25] Gama J, Castillo G. Learning with local drift detection. In: International conference on advanced data mining and applications. Springer; 2006. p. 42-55.

[26] Grossberg S. Nonlinear neural networks: Principles, mechanisms, and architectures. Neural networks. 1988;1(1):17-61.

[27] Luo H, Paal SG. A locally weighted machine learning model for generalized prediction of drift capacity in seismic vulnerability assessments. Computer-Aided Civil and Infrastructure Engineering. 2019;34(11):935-50.

[28] Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F, Pfharinger B, et al. Adaptive random forests for evolving data stream classification. Machine Learning. 2017;106:1469-95.

[29] Littlestone N, Warmuth MK. The weighted majority algorithm. Information and computation. 1994;108(2):212-61.

[30] Kolter JZ, Maloof MA. Dynamic weighted majority: An ensemble method for drifting concepts. The Journal of Machine Learning Research. 2007;8:2755-90.

[31] Domingos P, Hulten G. Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining; 2000. p. 71-80.

[32] Hulten G, Spencer L, Domingos P. Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining; 2001. p. 97-106.

[33] Bifet A, GavaldR̀. Adaptive learning from evolving data streams. In: Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8. Springer; 2009. p. 249-60.

[34] Bifet A, Gavaldà R, Holmes G, Pfahringer B. Machine Learning for Data Streams with Practical Examples in MOA. MIT Press; 2018. https://moa.cms.waikato.ac.nz/book/.

[35] Thomsen I, Rideng A. Oversikt over arbeidet med ny utvalgsplan. Oslo-Kongsvinger: Statistics Norway; 1974.

[36] Jentoft S. The Norwegian Labour Force Survey sampling design: A revision of the sampling plan for the 2021 NLFS. Kongsvinger-Oslo: Statistics Norway; 2022.

[37] European Parliment of the Council. Regulation (EU) 2019/1700 of the European Parliament and of the Council of 10 October 2019 establishing a common framework for European statistics relating to persons and households, based on data at individual level collected from samples, amending Regulations (EC) No 808/2004, (EC) No 452/2008 and (EC) No 1338/2008 of the European Parliament and of the Council, and repealing Regulation (EC) No 1177/2003 of the European Parliament and of the Council and Council Regulation (EC) No 577/98. Official Journal of the European Union. 2019;62(L261I).

[38] Statistics Norway. Standard for yrkesklassifisering (STYRK-08). Oslo-Kongsvinger: Statistics Norway; 2011. Available from: https://www.ssb.no/a/publikasjoner/pdf/notat_201117/notat_201117.pdf.

[39] International Labour Office. International Standard Classification of Occupations: Structure, group definitions and correspondence tables. Geneva; 2012. Available from: https://www.ilo.org/public/english/bureau/stat/isco/docs/publication08.pdf.

[40] Håland I, Næsheim HN. Måling av langsiktige endringer i yrkersstrukturen: Om ulike datakilder og yrkesstandarder. Oslo-Kongsvinger: Statistics Norway; 2016.

[41] Bø TP, Håland I. Dokumentasjon av Arbeidskraftundersøkelsen (AKU) etter omleggingen i 2006. Oslo-Kongsvinger: Statistics Norway; 2015.

[42] Statistics Norway. Arbeidskraftundersøkelsen 2001: Labour Force Survey 2001. Oslo-Kongsvinger: Statistics Norway; 2003.

[43] Horgen EH, Lien HH, Sandvik O, Sundt CS. Dokumentasjon av Arbeidskraftundersøkelsen etter omleggingen i 2021. Oslo-Kongsvinger: Statistics Norway; 2023.

[44] Johnsen MB, Konci E, Røv V. Nye variabler i a-ordningen: Ansettelsesform og årsak til sluttdato. Undersøkelse av datakvaliteten. Kongsvinger-Oslo: Statistics Norway; 2022.

[45] Aukrust I, Aurdal PS, Bråthen M, Køber T. Registerbasert sysselsettingsstatistikk. Oslo-Kongsvinger: Statistics Norway; 2010. Available from: https://www.ssb.no/a/publikasjoner/pdf/notat_201008/notat_201008.pdf.

[46] Villund O. Kvalitet på yrke i registerbasert statistikk: Resultater og videre utfordringer. Oslo-Kongsvinger: Statistics Norway; 2005. Available from: https://www.ssb.no/a/publikasjoner/pdf/notat_200514/notat_200514.pdf.

[47] Jianqiang Z, Xiaolin G. Comparison research on text pre-processing methods on twitter sentiment analysis. IEEE access. 2017;5:2870-9.

[48] R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria; 2023. Available from: https://www.R-project.org/.

[49] Feinerer I, Hornik K, Meyer D. Text Mining Infrastructure in R. Journal of Statistical Software. 2008 March;25(5):1-54.

[50] HaCohen-Kerner Y, Miller D, Yigal Y. The influence of preprocessing on text classification using a bag-of-words representation. PloS one. 2020;15(5):e0232525.

[51] A Norwegian stop word list; 2005. Accessed: 02.10-2023. http://snowball.tartarus.org/algorithms/norwegian/stop.txt.

[52] Porter MF. An algorithm for suffix stripping. Program. 1980;14(3):130-7.

[53] Müller DM. Classification of Consumer Goods into 5-digit COICOP 2018 Codes. NMBU; 2021.

[54] Jentoft S, Toth B, Müller D. From Manual to Machine: challenges in machine learning for COICOP coding. Nordic Statistical Meeting; 2022. Available from: https://www.nsm2022.is/s/FROM-MANUAL-TO-MACHINE-CHALLENGES-IN-MACHINE-LEARNING-FOR-COICOP-CODING.pdf.

[55] Ramos J. Using tf-idf to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning. vol. 242. Citeseer; 2003. p. 29-48.

[56] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information processing & management. 1988;24(5):513-23.

[57] Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, et al. API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning; 2013. p. 108-22.

[58] Raschka S, Mirjalili V. Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd; 2019.

[59] Lopes RHC. In: Lovric M, editor. Kolmogorov-Smirnov Test. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 718-20. Available from: https://doi.org/10.1007/978-3-642-04898-2_326.

[60] Stephens MA. Use of the Kolmogorov–Smirnov, Cramer–Von Mises and related statistics without extensive tables. Journal of the Royal Statistical Society Series B: Statistical Methodology. 1970;32(1):115-22.

[61] VanderWeele TJ, Mathur MB. Some desirable properties of the Bonferroni correction: is the Bonferroni correction really so bad? American journal of epidemiology. 2019;188(3):617-8.

[62] Escoufier Y. Le traitement des variables vectorielles. Biometrics. 1973:751-60.

[63] Tomic O, Graff T, Liland KH, Næs T. hoggorm: a python library for explorative multivariate statistics. The Journal of Open Source Software. 2019;4(39). Available from: http://joss.theoj.org/papers/10.21105/joss.00980.

[64] Hastie T, Tibshirani R, Friedman JH, Friedman JH. The elements of statistical learning: data mining, inference, and prediction. vol. 2. Springer; 2009.

[65] Kotsiantis SB. Decision trees: a recent overview. Artificial Intelligence Review. 2013;39:261-83.

[66] Breiman L. Random forests. Machine learning. 2001;45:5-32.

[67] Breiman L. Bagging predictors. Machine learning. 1996;24:123-40.

[68] Vapnik VN. An overview of statistical learning theory. IEEE transactions on neural networks. 1999;10(5):988-99.

[69] Burges CJ. A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery. 1998;2(2):121-67.

[70] Bolton C, et al. Logistic regression and its application in credit scoring. University of Pretoria; 2010.

[71] Stoltzfus JC. Logistic regression: a brief primer. Academic emergency medicine. 2011;18(10):1099-104.

[72] Ratcliff JW, Metzener D, et al. Pattern matching: The gestalt approach. Dr Dobb's Journal. 1988;13(7):46.

[73] Ilyankou I. Comparison of Jaro-Winkler and Ratcliff/Obershelp algorithms in spell check. IB Extended Essay Computer Science. 2014;1(2):3.

[74] Montiel J, Halford M, Mastelini SM, Bolmier G, Sourty R, Vaysse R, et al. River: machine learning for streaming data in python. The Journal of Machine Learning Research. 2021;22(1):4945-52.

[75] Brier GW. Verification of forecasts expressed in terms of probability. Monthly weather review. 1950;78(1):1-3.

[76] With ML. Yrkeskoding i EU-SILC; 2020. Unpublished.

[77] Xu S, Wang J. Dynamic extreme learning machine for data stream classification. Neurocomputing. 2017;238:433-49.

[78] Wang P, Jin N, Davies D, Woo WL. Model-centric transfer learning framework for concept drift detection. Knowledge-Based Systems. 2023:110705.

[79] García DE, DeCastro-García N, Castañeda ALM. An effectiveness analysis of transfer learning for the concept drift problem in malware detection. Expert Systems with Applications. 2023;212:118724.

Appendix A

List of stop words removed from text

Table A.1: List of stop words removed in text processing

Stop words used in text preprosessing					
og	i	jeg	det	at	en
et	den	til	er	som	på
de	med	han	av	ikke	ikkje
der	så	var	meg	seg	men
ett	har	om	vi	min	mitt
ha	hadde	hun	nå	over	da
ved	fra	du	ut	sin	dem
oss	opp	man	kan	hans	hvor
eller	hva	skal	selv	sjøl	her
alle	vil	bli	ble	blei	blitt
kunne	inn	når	være	kom	noen
noe	ville	dere	som	deres	kun
ja	etter	ned	skulle	denne	for
deg	si	sine	sitt	mot	å
meget	hvorfor	dette	disse	uten	hvordan
ingen	din	ditt	blir	samme	hvilken
hvilke	sånn	inni	mellom	vår	hver
hvem	vors	hvis	både	bare	enn
fordi	før	mange	også	slik	vært
være	båe	begge	siden	dykkar	dei
deira	deires	deim	di	då	eg
ein	eit	eitt	elles	honom	hjå
ho	hoe	henne	hennar	hennes	hoss
hossen	ikkje	ingi	inkje	korleis	korso
kva	kvar	kvarhelst	kven	kvi	kvifor
me	medan	mi	mine	mykje	no
nokon	noka	nokor	noko	nokre	si
sia	sidan	so	somt	somme	um
upp	vere	vore	verte	vort	varte
vart	as				

Appendix B

Python code for targeted matching

The following code provides the functions used for creating training data for the targeted matching algorithm.

```
from difflib import SequenceMatcher
from heapq import nlargest

def get_close_matches_indexes(text, possibilities, n=2, cutoff=0.6):
    """
    This function uses SequenceMatcher from difflib to find the index of the n
    most similar compared to a single text string.

    Parameters:
    - text (str): String containing the features for one observation in the
    prediction set.
    - possibilities (pd.Series): List of items to be compared with the text as
    a pandas series.
    - n (int, optional): Number of closest matches to return. Defaults to 2.
    - cutoff (float, optional): The minimum similarity ratio required for a match.
    Defaults to 0.6.

    Returns:
    list: A list containing the indexes of the best n matches.
    """
    result = []
    s = SequenceMatcher()
```

```
    # Set the second sequence as the text
    s.set_seq2(text)

    # Iterate through each possibility and compare it with the features
    for id, x in enumerate(possibilities):
        s.set_seq1(x)

        # Check if all three similarity ratios meet the cutoff
        if s.real_quick_ratio() >= cutoff and \
           s.quick_ratio() >= cutoff and \
           s.ratio() >= cutoff:
            result.append((s.ratio(), id))

    # Move the best scorers to the top of the list
    result = nlargest(n, result)

    # Extract the indexes of the best n matches
    return [x for score, x in result]


def create_training(X_tren, X_pred, y_tren, n=2):
    """
    Create training data by matching the strings in X_pred with corresponding
    values in X_tren.

    Parameters:
    - X_tren (pd.Series): Training data features. String variable.
    - X_pred (pd.Series): Prediction data features to be matched with X_tren.
    String variable.
    - y_tren (pd.Series): Training data labels corresponding to X_tren.
    - n (int, optional): Number of closest matches to consider when matching.
    Defaults to 2.
```

```
    Returns:
    tuple: A tuple containing matched X_tren features and corresponding y_tren
    labels.
    """
    # Establish empty lists for results
    X_tren_matched = []
    y_tren_matched = []

    # Iterate through each string in X_pred
    for i in range(len(X_pred)):
        # Get the best n matches in X_tren for the current prediction
        match_strings = get_close_matches_indexes(X_pred.iloc[i], X_tren,
            n=n, cutoff=0)

        # Check if there are any matches
        if len(match_strings) > 0:
            # Accumulate the matched features and labels
            X_tren_matched += X_tren.iloc[match_strings].tolist()
            y_tren_matched += y_tren.iloc[match_strings].tolist()

    return X_tren_matched, y_tren_matched
```

Appendix C

Model tuning results

Table C.1: Model tuning results for SVM tuning for Group A

Max features	C	Use idf	Kernel	Features	Mean test score
10000	10	True	linear	3gram	0.9931
5000	10	True	linear	3gram	0.9931
5000	1	True	linear	3gram	0.9927
10000	1	True	linear	3gram	0.9927
10000	100	True	linear	3gram	0.9923
5000	100	True	linear	3gram	0.9923
5000	10	False	rbf	3gram	0.9894
5000	100	False	rbf	3gram	0.9894
10000	10	False	rbf	3gram	0.9894
10000	100	False	rbf	3gram	0.9894
1000	1	True	linear	3gram	0.9894
10000	1	False	linear	3gram	0.9890
1000	100	True	rbf	3gram	0.9890
1000	10	True	rbf	3gram	0.9890
5000	1	False	linear	3gram	0.9890
5000	1	False	rbf	3gram	0.9886
1000	10	False	rbf	3gram	0.9886
10000	1	False	rbf	3gram	0.9886
1000	100	False	rbf	3gram	0.9886
1000	1	True	rbf	3gram	0.9886

Max features	C	Use TFIDF	Kernel	Features	Mean test score
1000	1	False	linear	3gram	0.9882
1000	0.1	True	linear	3gram	0.9882
5000	100	False	linear	3gram	0.9878
10000	10	False	linear	3gram	0.9878
10000	1	True	rbf	3gram	0.9878
5000	10	True	rbf	3gram	0.9878
10000	100	True	rbf	3gram	0.9878
5000	1	True	rbf	3gram	0.9878
5000	10	False	linear	3gram	0.9878
10000	100	False	linear	3gram	0.9878
10000	10	True	rbf	3gram	0.9878
5000	100	True	rbf	3gram	0.9878
10000	0.1	True	linear	3gram	0.9874
5000	0.1	True	linear	3gram	0.9874
1000	10	True	linear	3gram	0.9874
1000	100	True	linear	3gram	0.9874
500	100	False	rbf	3gram	0.9874
1000	1	False	rbf	3gram	0.9874
500	10	False	rbf	3gram	0.9874
1000	10	False	linear	3gram	0.9870
1000	100	False	linear	3gram	0.9870
500	0.1	True	linear	3gram	0.9866
10000	0.1	False	linear	3gram	0.9862
1000	0.1	False	linear	3gram	0.9862
5000	0.1	False	linear	3gram	0.9862
500	1	True	rbf	3gram	0.9858
500	100	True	rbf	3gram	0.9858
500	10	True	rbf	3gram	0.9858
500	1	False	rbf	3gram	0.9858
500	1	True	linear	3gram	0.9854
500	10	True	linear	3gram	0.9846
500	1	False	linear	3gram	0.9846
500	100	True	linear	3gram	0.9842

Max features	C	Use TFIDF	Kernel	Features	Mean test score
500	10	False	linear	3gram	0.9837
500	100	False	linear	3gram	0.9837
500	0.1	False	linear	3gram	0.9833
10000	1	True	linear	word	0.9825
5000	1	True	linear	word	0.9825
1000	10	True	rbf	word	0.9817
1000	100	True	rbf	word	0.9817
5000	10	True	linear	word	0.9813
10000	100	True	linear	word	0.9813
10000	10	True	linear	word	0.9813
5000	100	True	linear	word	0.9813
1000	10	True	linear	word	0.9805
10000	10	True	rbf	word	0.9785
10000	100	True	rbf	word	0.9785
5000	10	True	rbf	word	0.9785
5000	100	True	rbf	word	0.9785
1000	1	True	rbf	word	0.9785
5000	1	False	linear	word	0.9785
10000	1	False	linear	word	0.9785
500	100	True	rbf	word	0.9781
500	10	True	rbf	word	0.9781
500	1	True	rbf	word	0.9781
500	1	True	linear	word	0.9781
500	10	False	rbf	word	0.9781
500	100	False	rbf	word	0.9781
1000	10	False	linear	word	0.9781
500	10	True	linear	word	0.9781
1000	1	True	linear	word	0.9777
500	0.1	False	rbf	3gram	0.9772
1000	100	True	linear	word	0.9772
5000	10	False	linear	word	0.9772
10000	10	False	linear	word	0.9772
1000	1	False	linear	word	0.9768

Max features	C	Use TFIDF	Kernel	Features	Mean test score
5000	100	False	linear	word	0.9768
10000	100	False	linear	word	0.9768
500	1	False	linear	word	0.9764
500	1	False	rbf	word	0.9764
1000	0.1	False	rbf	3gram	0.9760
1000	100	False	rbf	word	0.9760
1000	10	False	rbf	word	0.9760
500	100	True	linear	word	0.9760
500	0.01	False	linear	3gram	0.9756
500	0.01	True	linear	3gram	0.9756
1000	0.01	False	linear	3gram	0.9752
500	10	False	linear	word	0.9752
5000	1	True	rbf	word	0.9748
10000	1	True	rbf	word	0.9748
500	0.1	True	linear	word	0.9744
1000	1	False	rbf	word	0.9740
1000	100	False	linear	word	0.9740
5000	0.01	False	linear	3gram	0.9732
1000	0.1	True	linear	word	0.9732
10000	0.01	False	linear	3gram	0.9732
5000	100	False	rbf	word	0.9728
10000	100	False	rbf	word	0.9728
10000	10	False	rbf	word	0.9728
5000	10	False	rbf	word	0.9728
500	100	False	linear	word	0.9728
1000	0.01	True	linear	3gram	0.9724
10000	0.1	False	rbf	3gram	0.9716
5000	0.1	False	rbf	3gram	0.9716
500	0.1	True	rbf	3gram	0.9707
10000	1	False	rbf	word	0.9699
5000	1	False	rbf	word	0.9699
5000	0.1	True	linear	word	0.9695
10000	0.1	True	linear	word	0.9695

Max features	C	Use TFIDF	Kernel	Features	Mean test score
1000	0.1	False	linear	word	0.9687
500	0.1	False	linear	word	0.9687
10000	0.1	False	linear	word	0.9675
5000	0.1	False	linear	word	0.9675
1000	0.1	True	rbf	3gram	0.9642
10000	0.01	True	linear	3gram	0.9618
5000	0.01	True	linear	3gram	0.9618
500	0.01	False	linear	word	0.9240
500	0.01	True	linear	word	0.9204
1000	0.01	False	linear	word	0.9175
10000	0.01	False	linear	word	0.9102
5000	0.01	False	linear	word	0.9102
500	0.1	False	rbf	word	0.9086
1000	0.01	True	linear	word	0.9009
1000	0.1	False	rbf	word	0.8996
5000	0.1	True	rbf	3gram	0.8850
10000	0.1	True	rbf	3gram	0.8850
5000	0.1	False	rbf	word	0.8797
10000	0.1	False	rbf	word	0.8797
5000	0.01	True	linear	word	0.8610
10000	0.01	True	linear	word	0.8610
500	0.1	True	rbf	word	0.8574
1000	0.1	True	rbf	word	0.8074
10000	0.1	True	rbf	word	0.7481
5000	0.1	True	rbf	word	0.7481
500	0.01	False	rbf	3gram	0.7237
1000	0.01	False	rbf	3gram	0.7156
500	0.01	True	rbf	3gram	0.7107
5000	0.01	False	rbf	3gram	0.7095
10000	0.01	False	rbf	3gram	0.7095
500	0.01	False	rbf	word	0.6623
1000	0.01	False	rbf	word	0.6558
5000	0.01	False	rbf	word	0.6469

Max features	C	Use TFIDF	Kernel	Features	Mean test score
10000	0.01	False	rbf	word	0.6469
1000	0.01	True	rbf	3gram	0.6331
500	0.01	True	rbf	word	0.5160
1000	0.01	True	rbf	word	0.4547
10000	0.01	True	rbf	3gram	0.4189
5000	0.01	True	rbf	3gram	0.4189
10000	0.01	True	rbf	word	0.3881
5000	0.01	True	rbf	word	0.3881

Table C.2: Model tuning results for SVM tuning for Group B

Max features	C	Use TFIDF	Kernel	Features	Mean test score
5000	1	True	linear	3gram	0.9567
10000	1	True	linear	3gram	0.9567
10000	10	True	linear	3gram	0.9567
5000	10	True	linear	3gram	0.9567
500	10	False	linear	3gram	0.9549
1000	1	True	linear	3gram	0.9549
500	100	False	linear	3gram	0.9531
1000	100	False	linear	3gram	0.9531
500	1	True	linear	3gram	0.9531
10000	100	True	linear	3gram	0.9531
5000	100	True	linear	3gram	0.9531
500	10	True	linear	3gram	0.9513
1000	10	False	linear	3gram	0.9513
10000	10	False	linear	3gram	0.9513
500	1	True	rbf	3gram	0.9513
1000	1	False	linear	3gram	0.9513
500	10	True	rbf	3gram	0.9513
500	100	True	rbf	3gram	0.9513
500	1	False	linear	3gram	0.9513
5000	10	False	linear	3gram	0.9513
5000	1	False	linear	3gram	0.9513
10000	1	False	linear	3gram	0.9513
5000	100	False	linear	3gram	0.9513
10000	100	False	linear	3gram	0.9513
500	100	True	linear	3gram	0.9495
1000	1	True	rbf	3gram	0.9495
1000	0.1	True	linear	3gram	0.9477
500	0.1	True	linear	3gram	0.9477
1000	10	True	linear	3gram	0.9477
1000	100	False	rbf	3gram	0.9477
10000	10	False	rbf	3gram	0.9477
10000	100	False	rbf	3gram	0.9477

Max features	C	Use TFIDF	Kernel	Features	Mean test score
5000	100	False	rbf	3gram	0.9477
5000	10	False	rbf	3gram	0.9477
1000	10	False	rbf	3gram	0.9477
500	1	False	rbf	3gram	0.9477
5000	100	True	rbf	3gram	0.9459
10000	100	True	rbf	3gram	0.9459
5000	10	True	rbf	3gram	0.9459
10000	10	True	rbf	3gram	0.9459
1000	100	True	rbf	3gram	0.9459
1000	10	True	rbf	3gram	0.9459
1000	100	True	linear	3gram	0.9459
1000	1	False	rbf	3gram	0.9459
500	10	False	rbf	3gram	0.9441
500	100	False	rbf	3gram	0.9441
5000	0.1	True	linear	3gram	0.9405
10000	0.1	True	linear	3gram	0.9405
10000	1	True	rbf	3gram	0.9405
5000	1	True	rbf	3gram	0.9405
5000	1	False	rbf	3gram	0.9387
10000	1	False	rbf	3gram	0.9387
500	0.1	False	linear	3gram	0.9351
1000	0.1	False	linear	3gram	0.9333
5000	0.1	False	linear	3gram	0.9297
10000	0.1	False	linear	3gram	0.9297
500	1	True	linear	word	0.9206
1000	1	False	linear	word	0.9189
5000	1	False	linear	word	0.9189
10000	1	False	linear	word	0.9189
5000	1	True	linear	word	0.9171
1000	1	True	linear	word	0.9171
10000	1	True	linear	word	0.9171
10000	100	True	linear	word	0.9170
5000	100	True	linear	word	0.9170

Max features	C	Use TFIDF	Kernel	Features	Mean test score
1000	100	True	linear	word	0.9170
500	1	False	linear	word	0.9170
10000	10	True	linear	word	0.9152
5000	10	True	linear	word	0.9152
1000	10	True	linear	word	0.9152
1000	10	False	linear	word	0.9134
5000	10	False	linear	word	0.9134
10000	10	False	linear	word	0.9134
1000	10	True	rbf	word	0.9116
1000	100	True	rbf	word	0.9116
5000	10	True	rbf	word	0.9116
5000	100	True	rbf	word	0.9116
10000	10	True	rbf	word	0.9116
10000	100	True	rbf	word	0.9116
10000	100	False	linear	word	0.9080
5000	100	False	linear	word	0.9080
1000	100	False	linear	word	0.9080
500	100	True	rbf	word	0.9080
500	10	True	rbf	word	0.9080
500	10	False	linear	word	0.9044
500	100	False	rbf	word	0.9026
500	10	False	rbf	word	0.9026
500	10	True	linear	word	0.9026
5000	100	False	rbf	word	0.9008
1000	10	False	rbf	word	0.9008
1000	100	False	rbf	word	0.9008
10000	10	False	rbf	word	0.9008
5000	10	False	rbf	word	0.9008
10000	100	False	rbf	word	0.9008
500	0.1	True	linear	word	0.8990
500	1	True	rbf	word	0.8972
500	0.1	False	linear	word	0.8972
500	1	False	rbf	word	0.8936

Max features	C	Use TFIDF	Kernel	Features	Mean test score
5000	1	False	rbf	word	0.8936
10000	1	False	rbf	word	0.8936
1000	1	False	rbf	word	0.8936
1000	0.1	True	linear	word	0.8936
5000	0.1	True	linear	word	0.8936
10000	0.1	True	linear	word	0.8936
5000	0.1	False	linear	word	0.8918
1000	0.1	False	linear	word	0.8918
500	100	False	linear	word	0.8918
10000	0.1	False	linear	word	0.8918
10000	1	True	rbf	word	0.8900
1000	1	True	rbf	word	0.8900
5000	1	True	rbf	word	0.8900
500	100	True	linear	word	0.8900
500	0.01	False	linear	3gram	0.7906
1000	0.01	False	linear	3gram	0.7743
5000	0.01	False	linear	3gram	0.7671
10000	0.01	False	linear	3gram	0.7671
500	0.01	True	linear	3gram	0.7636
1000	0.01	True	linear	3gram	0.7112
500	0.01	False	linear	word	0.6968
500	0.1	False	rbf	3gram	0.6949
10000	0.01	True	linear	3gram	0.6841
5000	0.01	True	linear	3gram	0.6841
1000	0.01	False	linear	word	0.6805
5000	0.01	False	linear	word	0.6805
10000	0.01	False	linear	word	0.6805
1000	0.1	False	rbf	3gram	0.6589
500	0.01	True	linear	word	0.6445
10000	0.1	False	rbf	3gram	0.6083
5000	0.1	False	rbf	3gram	0.6083
5000	0.01	True	linear	word	0.6065
1000	0.01	True	linear	word	0.6065

Max features	C	Use TFIDF	Kernel	Features	Mean test score
10000	0.01	True	linear	word	0.6065
500	0.1	False	rbf	word	0.4928
500	0.1	True	rbf	3gram	0.4874
1000	0.1	False	rbf	word	0.4549
10000	0.1	False	rbf	word	0.4549
5000	0.1	False	rbf	word	0.4549
1000	0.1	True	rbf	3gram	0.4278
500	0.1	True	rbf	word	0.4278
5000	0.1	True	rbf	3gram	0.4260
10000	0.1	True	rbf	3gram	0.4260
1000	0.1	True	rbf	word	0.4242
500	0.01	True	rbf	word	0.4242
500	0.01	False	rbf	word	0.4242
5000	0.01	False	rbf	word	0.4242
10000	0.01	True	rbf	word	0.4242
10000	0.01	False	rbf	word	0.4242
10000	0.1	True	rbf	word	0.4242
5000	0.1	True	rbf	word	0.4242
5000	0.01	True	rbf	word	0.4242
1000	0.01	True	rbf	word	0.4242
1000	0.01	False	rbf	word	0.4242
500	0.01	True	rbf	3gram	0.4242
500	0.01	False	rbf	3gram	0.4242
5000	0.01	True	rbf	3gram	0.4242
5000	0.01	False	rbf	3gram	0.4242
10000	0.01	True	rbf	3gram	0.4242
10000	0.01	False	rbf	3gram	0.4242
1000	0.01	False	rbf	3gram	0.4242
1000	0.01	True	rbf	3gram	0.4242

Table C.3: Model tuning results from logistic regression for Group A

Max features	C	Use TFIDF	Features	Mean test score
5000	10	True	3gram	0.990655
10000	10	True	3gram	0.990655
10000	10	False	3gram	0.989435
5000	10	False	3gram	0.989435
1000	10	True	3gram	0.989029
1000	10	False	3gram	0.988216
1000	1	True	3gram	0.987403
500	1	True	3gram	0.986997
5000	1	True	3gram	0.986997
10000	1	True	3gram	0.986997
10000	1	False	3gram	0.986996
5000	1	False	3gram	0.986996
1000	1	False	3gram	0.986184
500	10	True	3gram	0.986183
500	10	False	3gram	0.984966
500	1	False	3gram	0.984559
500	10	False	word	0.980087
5000	10	True	word	0.978868
10000	10	True	word	0.978868
500	10	True	word	0.977651
1000	10	False	word	0.977650
1000	10	True	word	0.976838
500	0.1	True	3gram	0.976024
5000	10	False	word	0.976023
10000	10	False	word	0.976023
500	0.1	False	3gram	0.975619
5000	0.1	False	3gram	0.975212
10000	0.1	False	3gram	0.975212
1000	0.1	False	3gram	0.975212
500	1	True	word	0.973993
1000	0.1	True	3gram	0.972773
1000	1	True	word	0.971555

Max features	C	Use TFIDF	Features	Mean test score
5000	1	True	word	0.969524
10000	1	True	word	0.969524
500	1	False	word	0.968711
1000	1	False	word	0.967899
10000	1	False	word	0.965460
5000	1	False	word	0.965460
5000	0.1	True	3gram	0.965458
10000	0.1	True	3gram	0.965458
500	0.1	True	word	0.927262
500	0.1	False	word	0.926854
1000	0.1	False	word	0.923196
10000	0.1	False	word	0.919136
5000	0.1	False	word	0.919136
1000	0.1	True	word	0.917509
10000	0.1	True	word	0.884195
5000	0.1	True	word	0.884195
500	0.01	False	3gram	0.754570
500	0.01	True	3gram	0.746445
1000	0.01	False	3gram	0.744413
1000	0.01	True	3gram	0.744413
10000	0.01	False	3gram	0.744413
5000	0.01	False	3gram	0.744413
5000	0.01	True	3gram	0.742380
10000	0.01	True	3gram	0.742380
500	0.01	True	word	0.728567
1000	0.01	True	word	0.727754
5000	0.01	True	word	0.726535
10000	0.01	True	word	0.726535
500	0.01	False	word	0.722878
1000	0.01	False	word	0.722065
10000	0.01	False	word	0.720032
5000	0.01	False	word	0.720032

Table C.4: Model tuning results from logistic regression for Group B

Max features	C	Use TFIDF	Features	Mean test score
500	10	True	3gram	0.956740
5000	10	True	3gram	0.953137
1000	10	True	3gram	0.953137
10000	10	True	3gram	0.953137
500	10	False	3gram	0.951319
1000	10	False	3gram	0.949517
10000	10	False	3gram	0.949517
5000	10	False	3gram	0.949517
500	1	True	3gram	0.945913
1000	1	True	3gram	0.942310
5000	1	True	3gram	0.938706
10000	1	True	3gram	0.938706
500	1	False	3gram	0.938657
1000	1	False	3gram	0.933251
10000	1	False	3gram	0.927846
5000	1	False	3gram	0.927846
1000	10	True	word	0.918853
5000	10	True	word	0.918853
10000	10	True	word	0.918853
500	10	True	word	0.918821
5000	10	False	word	0.915250
10000	10	False	word	0.915250
1000	10	False	word	0.915250
500	10	False	word	0.913399
500	1	True	word	0.899017
1000	1	True	word	0.897199
10000	1	True	word	0.897199
5000	1	True	word	0.897199
500	1	False	word	0.895381
10000	1	False	word	0.893579
5000	1	False	word	0.893579
1000	1	False	word	0.893579

Max features	C	Use TFIDF	Features	Mean test score
500	0.1	False	3gram	0.803260
1000	0.1	False	3gram	0.785127
10000	0.1	False	3gram	0.783325
5000	0.1	False	3gram	0.783325
500	0.1	True	3gram	0.779787
1000	0.1	True	3gram	0.743702
500	0.1	False	word	0.736478
5000	0.1	True	3gram	0.718395
10000	0.1	True	3gram	0.718395
5000	0.1	False	word	0.716609
10000	0.1	False	word	0.716609
1000	0.1	False	word	0.716609
500	0.1	True	word	0.689566
10000	0.1	True	word	0.673333
1000	0.1	True	word	0.673333
5000	0.1	True	word	0.673333
500	0.01	False	3gram	0.521704
1000	0.01	False	3gram	0.4838
10000	0.01	False	3gram	0.458509
5000	0.01	False	3gram	0.458509
500	0.01	False	word	0.433202
500	0.01	True	3gram	0.4314
5000	0.01	True	3gram	0.425995
1000	0.01	True	3gram	0.425995
10000	0.01	True	3gram	0.425995
10000	0.01	False	word	0.424193
5000	0.01	True	word	0.424193
10000	0.01	True	word	0.424193
1000	0.01	True	word	0.424193
500	0.01	True	word	0.424193
1000	0.01	False	word	0.424193
5000	0.01	False	word	0.424193

Table C.5: Model tuning results from Random forest for group A

Max features	n estimators	Use TFIDF	Features	Mean test score
5000	100	True	3gram	0.984153
5000	50	True	3gram	0.983746
500	500	False	3gram	0.983340
500	100	False	3gram	0.983339
500	500	True	3gram	0.983339
10000	500	False	3gram	0.982931
500	100	True	3gram	0.982528
5000	50	False	3gram	0.982527
1000	100	False	3gram	0.982527
10000	500	True	3gram	0.982525
500	50	True	3gram	0.982120
1000	500	False	3gram	0.982120
500	50	False	3gram	0.982120
1000	500	True	3gram	0.982119
10000	50	False	3gram	0.981714
5000	100	False	3gram	0.981713
10000	100	True	3gram	0.981713
5000	500	True	3gram	0.981307
1000	100	True	3gram	0.981306
10000	100	False	3gram	0.980901
1000	50	False	3gram	0.980494
5000	500	False	3gram	0.980493
1000	50	True	3gram	0.980088
10000	50	True	3gram	0.980088
10000	50	True	word	0.971962
1000	100	True	word	0.971146
5000	100	False	word	0.970744
5000	50	True	word	0.970741
5000	500	False	word	0.970338
10000	500	True	word	0.969931
10000	100	False	word	0.969118
5000	500	True	word	0.969118

Max features	n estimators	Use TFIDF	Features	Mean test score
1000	500	True	word	0.969114
10000	500	False	word	0.968712
5000	100	True	word	0.968711
1000	500	False	word	0.968709
500	100	True	word	0.968708
10000	50	False	word	0.968305
500	500	True	word	0.968303
10000	100	True	word	0.967899
500	50	False	word	0.967086
500	500	False	word	0.967085
500	50	True	word	0.967081
1000	100	False	word	0.965864
1000	50	False	word	0.965458
5000	50	False	word	0.965054
1000	50	True	word	0.965052
500	100	False	word	0.963833

Table C.6: Model tuning results from Random forest tuning for Group B

Max features	n estimators	Use TFIDF	Features	Mean test score
500	500	False	3gram	0.945897
1000	50	False	3gram	0.945880
1000	500	True	3gram	0.944095
500	100	True	3gram	0.944062
5000	100	False	3gram	0.942293
500	50	True	3gram	0.942277
1000	50	True	3gram	0.940475
500	500	True	3gram	0.940475
1000	500	False	3gram	0.940475
500	100	False	3gram	0.938673
5000	500	True	3gram	0.936888
10000	100	True	3gram	0.936888
5000	500	False	3gram	0.936888
10000	500	False	3gram	0.936888
10000	500	True	3gram	0.936888
10000	50	False	3gram	0.936871
1000	100	True	3gram	0.935102
5000	50	True	3gram	0.935070
10000	100	False	3gram	0.935070
5000	50	False	3gram	0.935053
10000	50	True	3gram	0.933268
1000	100	False	3gram	0.933235
500	50	False	3gram	0.931450
5000	100	True	3gram	0.929648
1000	100	True	word	0.895364
10000	100	True	word	0.893579
5000	100	True	word	0.891777
1000	100	False	word	0.891761
5000	100	False	word	0.891761
10000	500	True	word	0.889959
10000	50	True	word	0.888174
500	100	True	word	0.888157

Max features	n estimators	Use TFIDF	Features	Mean test score
10000	100	False	word	0.888157
10000	500	False	word	0.888157
1000	500	False	word	0.888157
1000	500	True	word	0.888157
5000	50	False	word	0.888157
5000	500	False	word	0.888157
500	500	True	word	0.888141
1000	50	False	word	0.886355
5000	500	True	word	0.886355
1000	50	True	word	0.884570
5000	50	True	word	0.882752
10000	50	False	word	0.882752
500	500	False	word	0.882719
500	50	False	word	0.880934
500	100	False	word	0.875495
500	50	True	word	0.875495

Table C.7: Model tuning results from Hoeffding adaptive tree tuning for Group A

Grace period	Delta	Features	Mean test score
1	0.1	word	0.634721
1	0.9	word	0.631469
1	1e-07	word	0.602207
1	0.001	word	0.600588
10	1e-07	word	0.583915
10	0.1	word	0.572935
10	0.001	word	0.570908
200	0.001	word	0.570095
200	1e-07	3gram	0.570085
200	1e-07	word	0.568061
10	1e-07	3gram	0.567250
50	0.001	3gram	0.565621
50	1e-07	3gram	0.564405
200	0.001	3gram	0.563993
10	0.001	3gram	0.562374
10	0.1	3gram	0.553029
50	0.1	3gram	0.552221
200	0.1	word	0.546927
200	0.1	3gram	0.544907
200	0.9	3gram	0.500997
50	0.9	3gram	0.463638
10	0.9	3gram	0.444548
10	0.9	word	0.414103
200	0.9	word	0.385603

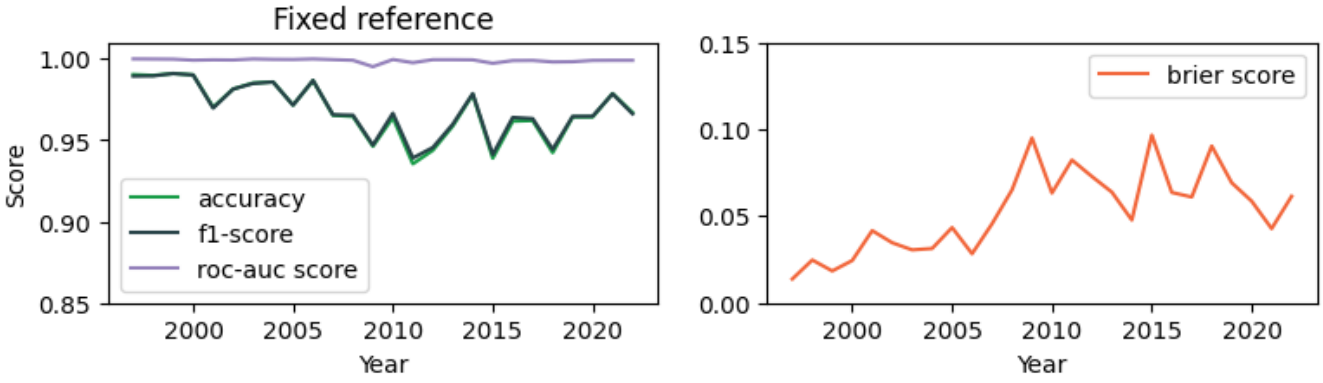
Table C.8: Model tuning results for Hoeffding adaptive tree tuning for Group B

Grace period	Delta	Features	Mean test score
1	0.001	word	0.640950
1	1e-07	word	0.639066
10	0.1	word	0.624701
10	0.001	word	0.619328
10	1e-07	word	0.601245
200	0.1	word	0.592236
200	0.001	word	0.586798
1	0.1	word	0.576020
200	1e-07	word	0.574201
10	0.1	3gram	0.509124
50	0.1	3gram	0.505471
200	0.001	3gram	0.501867
10	0.001	3gram	0.500131
200	0.1	3gram	0.500066
50	1e-07	3gram	0.500066
10	1e-07	3gram	0.500049
200	1e-07	3gram	0.498247
50	0.001	3gram	0.494660
10	0.9	word	0.447797
1	0.9	word	0.440262
50	0.9	3gram	0.436806
10	0.9	3gram	0.424193
200	0.9	3gram	0.386355
200	0.9	word	0.355217

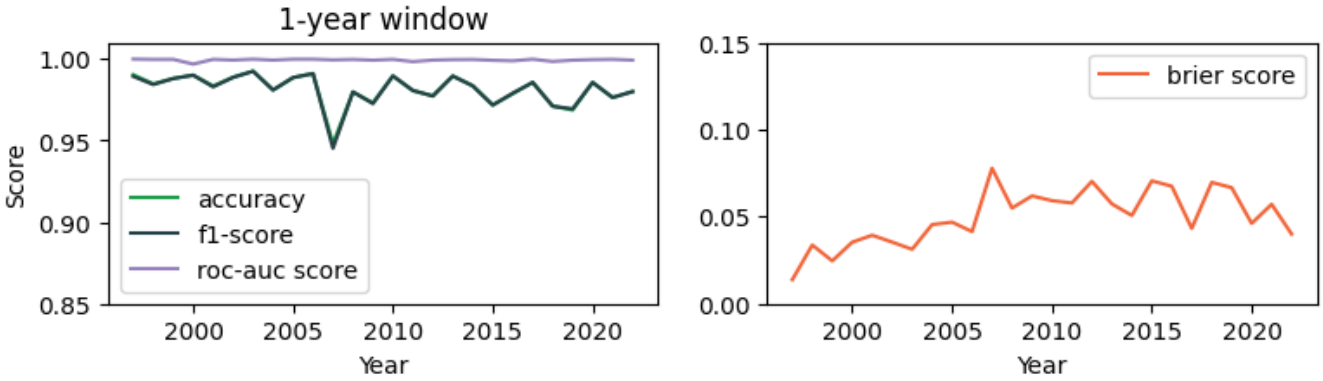
Appendix D

Performance metrics

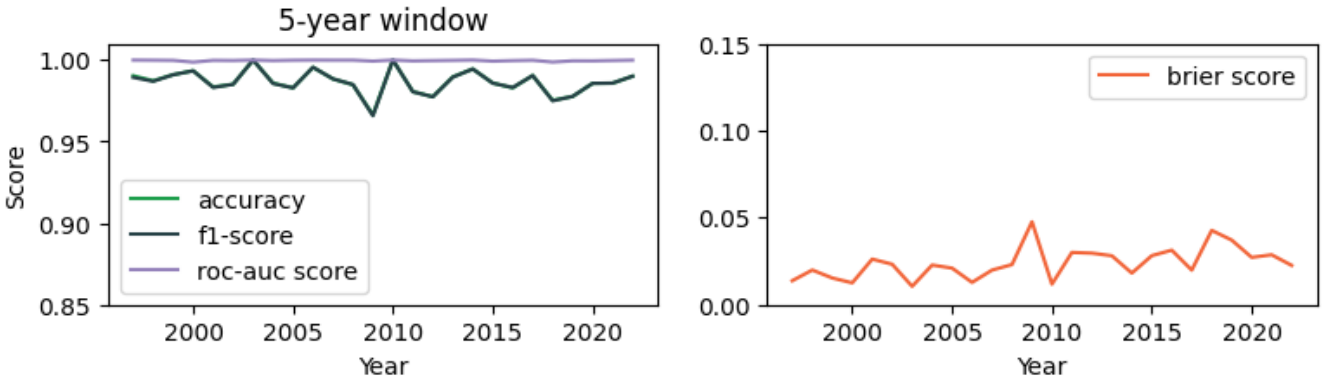
The following figures show performance metrics for the 6 modeling approaches.



(a) Fixed reference

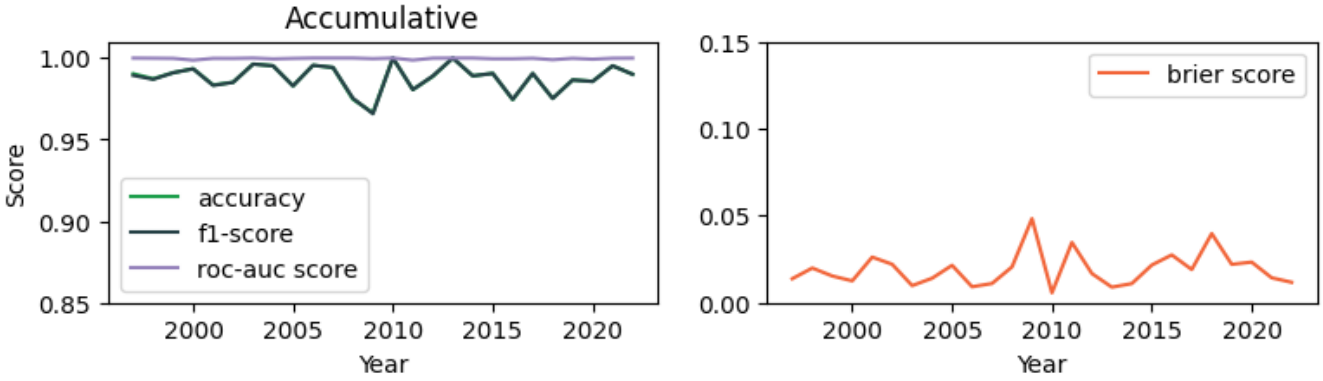


(b) 1-year window

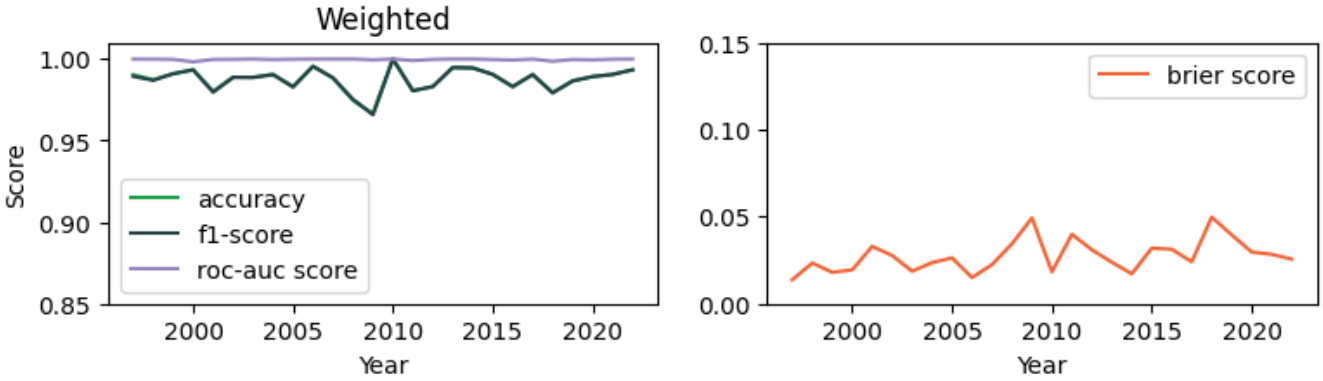


(c) 5-year window

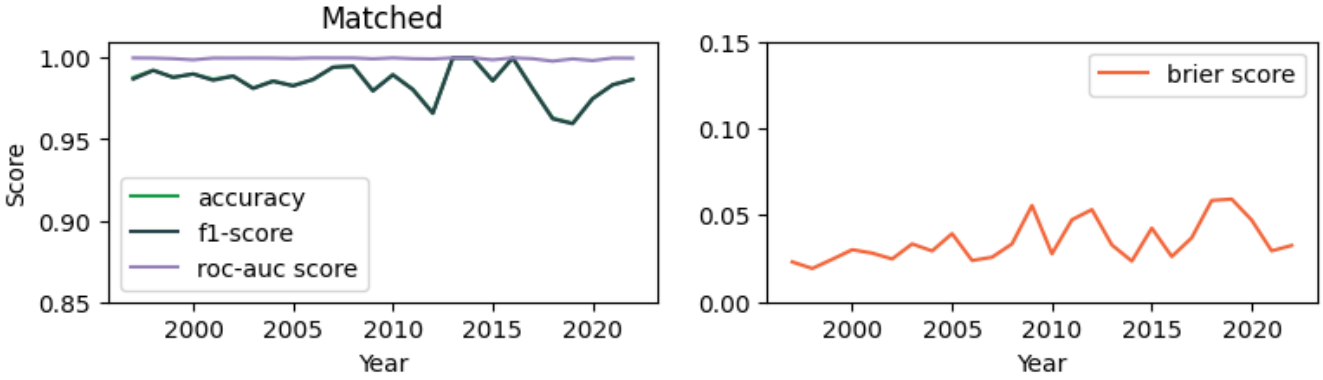
Fig. D.1: Comparison of performance metrics for Group A showing fixed-reference and sliding-window approaches.



(a) Accumulative approach

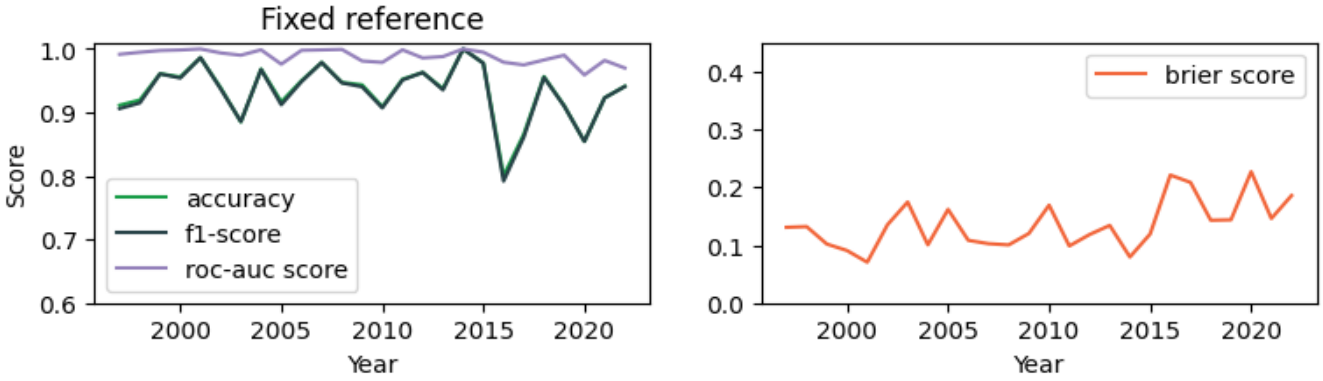


(b) weighted approach

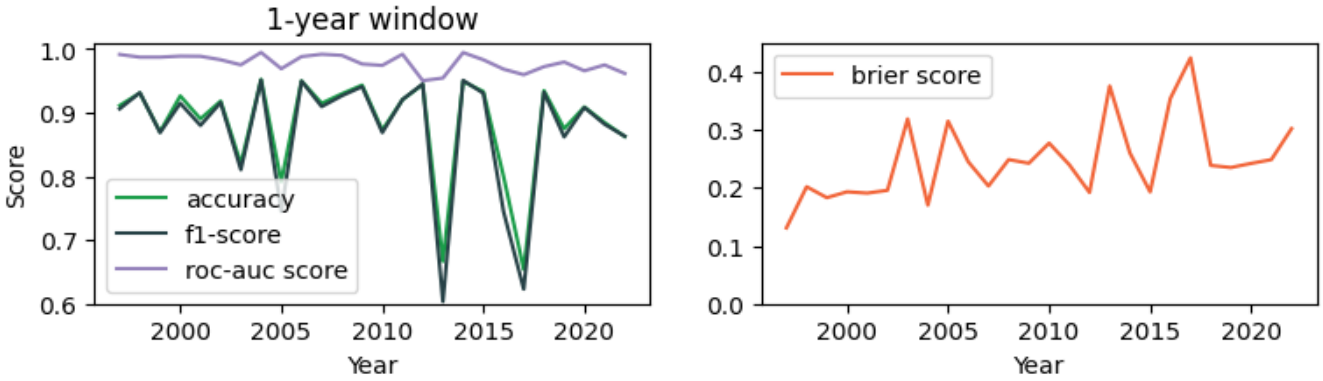


(c) Matched approach

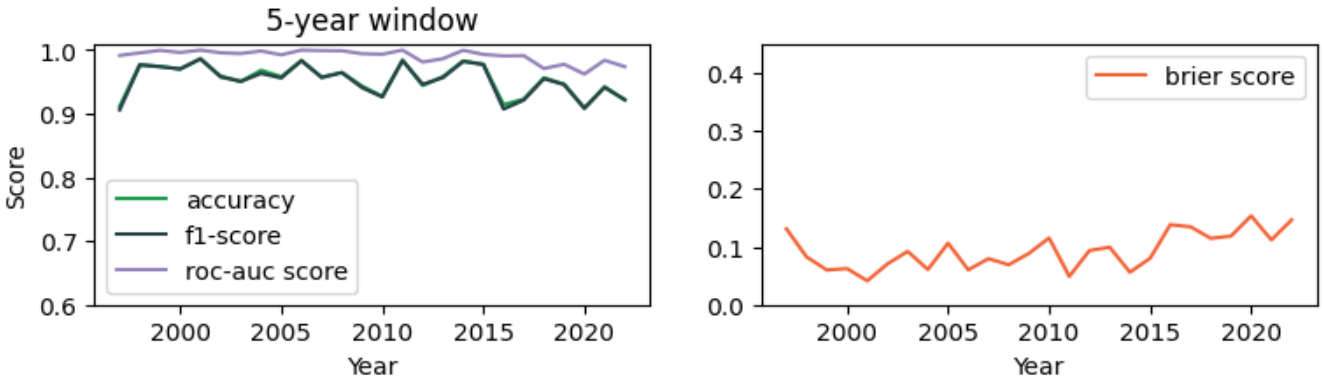
Fig. D.1: Comparison of performance metrics for Group A showing accumulative, weighted and matched approaches.



(d) Fixed reference

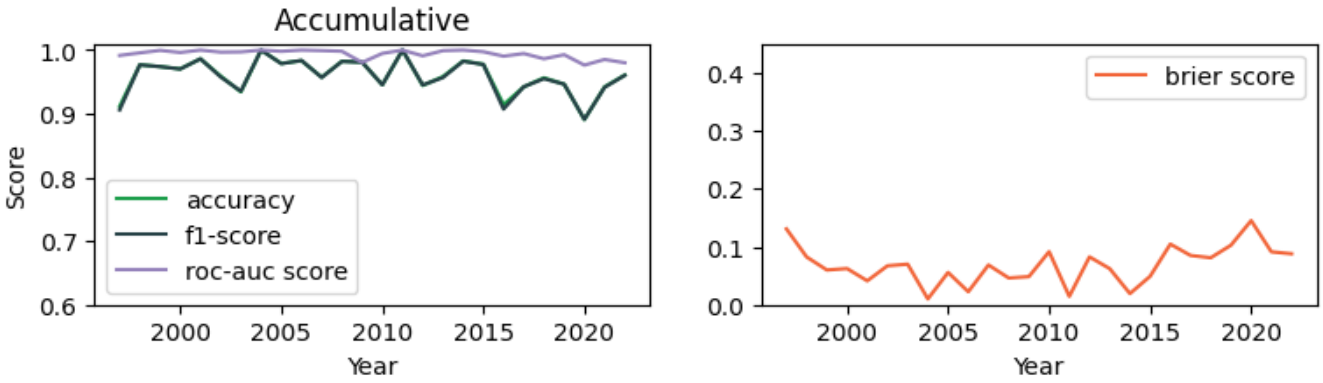


(e) 1-year window

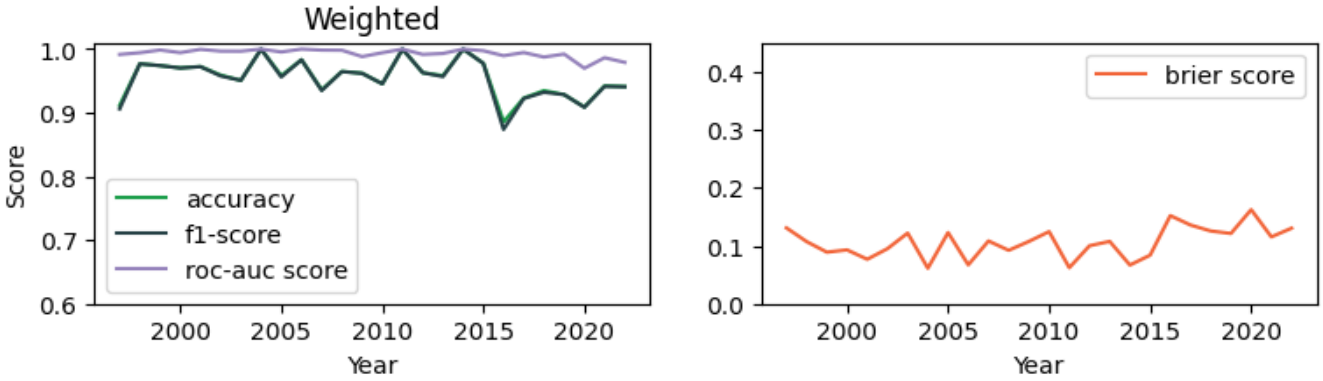


(f) 5-year window

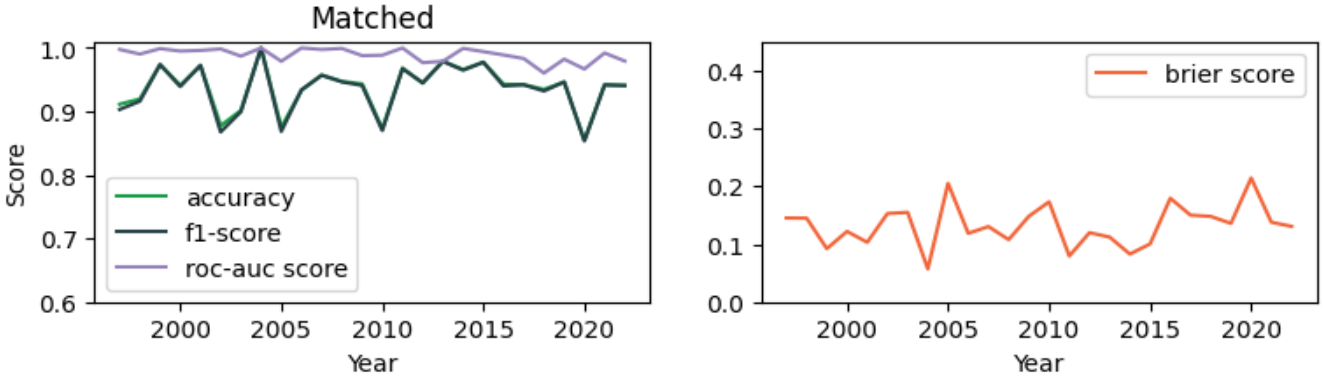
Fig. D.2: Comparison of performance metrics for Group B showing fixed-reference and sliding-window approaches.



(a) Accumulative approach

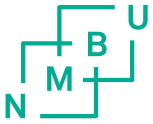


(b) weighted approach



(c) Matched approach

Fig. D.2: Comparison of performance metrics for Group B showing accumulative, weighted and matched approaches.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway