Universiteit Utrecht

Master's Thesis

Detecting Prediction Influence Drift In Data Streams

Author:
Tineke Jelsma *(6574696)*

Supervisors:
prof. habil. Georg Krempl
prof. dr. A.P.J.M. Siebes

External supervisor:
Iman Jabor

Faculty of Science
MSc Business Informatics

July 6, 2021

UTRECHT UNIVERSITY

Abstract

Detecting Prediction Influence Drift In Data Streams

Tineke Jelsma

Today, many machine learning models are being deployed to make decisions. One of the assumptions of the models is that the data are stationary over time. If the data change over time due to a dynamic environment, this is called concept drift. However, the predictions of the model could also influence future data. For example, if a model predicts that stock prices will increase, more stocks are bought, and the stock prices will indeed move up. The opposite is also possible: If malware is detected, malware is redesigned in such a way that it will not be detected as malware in the future. These mechanisms are called a *self-fulfilling* and *self-defeating* prophecy.

This thesis aims to distinguish between *prediction influence drift* and *intrinsic drift*. First, a method is developed in which a *feedback loop* exists between the data and the predictions of a machine learning model. Next, a prediction influence detection approach is created that calculates the density trajectories of data before and after classification. Linear regression models are fitted on the trajectories before classification. If the predictions differ from the density trajectories after classification, this indicates the start of drift after classification. If prediction influence exists, the differences between the predictions and trajectories within a true class are expected to be different. The detection approach is evaluated on synthetic data with and without prediction influence, and with and without intrinsic drift. Density trajectories on real-world data are presented and discussed.

The data generator shows promising results because the performance of a model with prediction influence is correctly influenced. The detection approach does not show clear results. The density trajectories are different per influence, but the differences between predicted and calculated densities cannot be distinguished. The trajectories on real-world data are interesting, but due to the lack of true labels, the approach cannot detect any prediction influence.

A first step in detecting prediction influence is taken, but the current approach does not work. The results show a clear influence on the performance of the machine learning models. Both the prediction influence data generator and the density trajectories look promising and could be used in new approaches.

***K*eywords** Prediction Influence · Concept Drift · Self-Fulfilling Influence · Self-Defeating Influence · Data Generation

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Problem Statement

With the rise of data in our economy, it has become much easier for users to build data models. Normally, a model is trained with a large amount of data and is tested with different data. In the real world, often data points do not come all at once, but arrive sequentially over time. This creates a new issue: When a model is trained on time series data, the model will accurately predict the data at that time point but it is not certain that the model will have the same performance on data at a later time point. If the data are stationary over time, meaning that the statistical properties of the data do not change over time, the model will show a stable performance over time. However, there are many situations where the properties will not remain the same, such as trends and seasonal changes. An obvious way to solve this problem is to regularly retrain the model on newer data. Sometimes this is not an optimal solution: Retraining is expensive and the cause of the change is not detected and eventually solved.

Detecting change in data, usually called concept drift or change points, is a big challenge. In current detection approaches, changes in data are assumed to be caused by the environment. The decision-making or detection systems are only receiving data as input without influencing it. However, feedback loops can arise. If the predictions of the system are used to make decisions related to the data, the data may change because of these predictions. In other words, systems can influence their own environment.

Two different types of this prediction influence can be distinguished: Self-fulfilling influence and self-defeating influence. With a self-fulfilling influence, the predictions cause themselves to be true. If the bank denies someone a loan because the person is classified as high risk, this person might use their credit card, which will result in being high risk. If the police classify a certain area as high risk based on crime reporting, the area will be better surveilled. Highly surveilled areas receive more crime reporting, thus this feedback loop will lead to a self-fulfilling prophecy. With a self-defeating influence, the predictions cause themselves to be false. This influence could be present in a malware detection system. A malware detector detects malware, but the goal of malware designers is to be unnoticed and undetected. If a malware designer discovers that their malware is detected, they will change their malware until the detector will not classify it as malware anymore. Self-defeating prophecies are also visible today:

If an algorithm predicts a high amount of Covid-19 cases, governments will change their strategies, which hopefully will lead to a wrong prediction.

Prediction influence can be detected throughout the whole data set, but it is also possible to have prediction influence in a part of the data set. We identify multiple types of prediction influence:

- **Prediction influence in regression**: Predicting Covid-19 cases is a result of a regression model. The model does not predict a label but predicts a value. The prediction influence is therefore applicable on the whole data set. The prediction that is made at a certain point in time is based on a set of features. Measures against the virus are part of the features. So, when the measures change based on the prediction of the Covid-19 cases, the set of features changes as well. With a different set of features, a different prediction is made. If the prediction is true, the measures against Covid-19 have probably stayed the same, while an incorrect prediction is a strong indication that something might have changed.
- **Prediction influence in classification**: We expect that two different types of prediction influences exist:
  - **Prediction influence in a part of the data set**: We expect that in certain data sets, prediction influence only happens in a part of the subset, such as in one label. For example, a malware data set has positives (malware) and negatives (non-malware). Malware designers will change their malware when the malware is detected (self-defeating influence). If malware is not detected as malware, then the designers do not have to change their malware. Likewise, if non-malware is not detected as malware, software engineers will not change anything. It is also possible for the self-fulfilling influence to be present in part of the data set.
  - **Prediction influence in the complete data set**: On the other hand, prediction influence can manifest in the complete data set, in all labels. A real-world example is when a teacher tells their students they will perform well, they are going to perform better than expected, and when a teacher tells their students they will perform badly, they are going to perform worse than expected. A system that grants or rejects loans, might have a similar effect. If a loan helps a company with financial success, the system might give a new loan based on this. On the other hand, a company that did not get a loan, might get in financial trouble, and therefore will be rejected a loan in the future as well. This type of prediction influence is also possible for the self-defeating influence.
- **Prediction influence in recommender systems**: Recommending items to users is becoming a standard practice in social media. We expect that a self-fulfilling influence is visible in recommender systems. A user has a certain profile, and based on the profile the recommender system

recommends certain items. If the user likes the items, the user profile gets updated. This cycle continues and the user profile changes based on the recommendations.

The focus of this research is to detect data drift in a binary data stream and to differentiate self-fulfilling drift, self-defeating drift, and intrinsic drift.

1.1 Research Question

The following research question with sub-questions is defined:

RQ: *To what extent can we distinguish between intrinsic concept drift and prediction influence drift in a binary data stream?*

Several sub-questions are defined that support the main research question:

SRQ1: *How can we detect the presence of drift?*

Before being able to differentiate between two types of drift, a detector should be able to detect any drift. This support question is answered by reviewing many detection approaches for concept drift, data drift, and anomaly detection.

SRQ2: *How can we simulate the interaction between a machine learning model and a data generator?*

Prediction influence is caused by the interaction between a model and a data set, therefore simulating this interaction will lead to prediction influence in the data set. This question will be answered by looking into current papers that create interaction and by creating a data generator that interacts with the predictions of a model.

SRQ3: *How can we detect prediction influence extending an existing concept drift detection method?*

This question can be answered by designing a detection approach that detects prediction influence. A concept drift detection method is picked from the methods in the literature review.

SRQ4: *How can we evaluate the prediction influence detector?*

The detector has to be evaluated to validate the design. The detector is validated by using a synthetically generated data stream (created for SRQ2). The prediction influence detector is applied to different sets of data streams. By comparing the results of the detector on streams with and without synthetically created prediction influence drift, and with or without intrinsic drift, it is possible to evaluate the approach.

SRQ5: *What is the performance of the prediction influence detector on real-world data?*

Finally, the detection approach is used on real-world data, to evaluate whether the data set contains any drift. If the data set contains drift, the detection approach detects whether this is intrinsic drift or prediction influence drift.

Symbol	Meaning
$P()$	Probability function
X	Set of features
y	True label
$\hat{y}$	Predicted label

Table 1.1: Symbols in this thesis

Abbreviation	Meaning
CM	Confusion Matrix
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
b	Temporal bandwidth
l	Track length

Table 1.2: Abbreviations in this thesis

1.2 Notation

Standard notations and their meanings are described in Table 1.1, commonly used abbreviations are described in Table 1.2.

1.3 Implementation

The implementation of the approach has been developed with Python 3 [40] in the package River [29]. The code of the experiments and the results can be found in the GitHub repository[1]. Furthermore, several Python packages are used, such as Matplotlib [19] for visualization, pandas for dataframes [32] and numPy [17] and sciPy [41] for programming.

1.4 Outline

The rest of the thesis is structured as follows: First, the related literature regarding concept drift, anomaly detection, and change point detection is given. Next, the implementation of the data generator, the detector, and the evaluation is explained. Then, the results of the implementation on synthetic data and real-world data are evaluated and compared. Finally, we discuss and summarize the results, conclude, and give some suggestions for future work.

[1] `https://github.com/TinekeJelsma/river-private`

Chapter 2

Related Literature

2.1 General

Detecting change in data over time is a problem which is covered in multiple disciplines. In this literature review, the state-of-the-art approaches in concept drift detectors, anomaly detectors, and change point detectors are discussed. In general, the detectors have two main approaches: offline and online detection. In online detection, the detector processes the data point immediately after it arrives, while in offline detection the detector processes the entire data set at once [13]. The approaches are suitable for different scenarios: An online detector is favourable when a change has to be detected as soon as possible while minimizing false alarms, and an offline detector is favourable when all changes in a sequence have to be identified [13].

In online detection, the data should be in the form of a stream. Babcock et al. [4] define a data stream as follows:

- Data elements in a stream arrive online;
- The model has no control over the order in which data points arrive;
- Data streams are potentially unbound in size;
- Once a data point is processed, it is discarded or archived. It cannot be retrieved easily unless it is explicitly stored in the memory.

In the next section, the definition of concept drift, change points, and anomalies are introduced, and several detectors are described. In every field, the gaps and challenges are described. Then, the related literature on prediction influence is given, and finally, several data generators in changing environments are introduced.

2.2 Concept Drift

2.2.1 Definitions

In a dynamic world, the patterns and relations in data often change over time. In machine learning, the goal of a model is to predict the target variable $y \in dom(Y)$ given a set of features $X = \{X_1, ..., X_n\}$. The model is trained with a

training set, where both the input features and the target variable per instance are known. After training, the model is tested with new instances, where the target variable is not known at the time of prediction. The probability that a class belongs to a certain set of features according to the Bayesion Decision Theory [14] is:

$$p(y \mid X) = \frac{p(y)p(X \mid y)}{p(X)} \tag{2.1}$$

where $p(X) = \sum_{y=1}^{c} p(y)p(X \mid y)$. The model only performs well if $p(y \mid X)$ does not change. Changes happen for various reasons: changing interests, changing populations, or adversary activities [49]. Specifically, a changing relationship between the input data and the target variable is called concept drift [16]. The formal way to describe a concept is [16]:

$$Concept = P(X, y) \tag{2.2}$$

Concept drift is:

$$P_t(X, Y) \neq P_u(X, Y), \tag{2.3}$$

where t and u are different time points [47]. Similarly, a concept drift between two time periods $[t, u]$ and $[v, w]$ can be defined as [47]:

$$P_{[t,u]}(X, Y) \neq P_{[v,w]}(X, Y) \tag{2.4}$$

Two types of concept drift are distinguished:

- *Real concept drift (class drift)*: A change in posterior distribution of the output given the input $P(Y \mid X)$ changes while $P(X)$ may stay unchanged [16]; Formally, this occurs when $P_t(Y \mid X) \neq P_u(Y \mid X)$.
- *Virtual drift (Covariate drift)*: The distribution of the features P(X) changes while the posterior distribution $P(Y \mid X)$ is unaffected [16]. Formally, this occurs when $P_t(X) \neq P_u(X) \wedge P_t(Y \mid X) = P_u(Y \mid X)$.

In a paper of 2016, Webb defines multiple quantitative measures of concept drift [45]:

- *Magnitude of a drift*: The distance of the concepts between the start and the end of the concept drift;
- *Duration of a drift*: The time between the start and the end of the concept drift;
- *Path length of a drift*: This combines the magnitude and duration of a drift;
- *Rate of a drift*: This quantifies the speed of the distribution change. The average rate is the path length divided by the duration.

Brzeziński and Stefanowski [10] categorise concept drift in three forms: Sudden drift, gradual drift, and recurring drift. Sudden drift occurs when the distribution is completely replaced by a different distribution at a high rate, gradual drift occurs at a slower rate. Recurring drift occurs when previous

distributions recur. Webb et al. [45] state that these terms have an informal definition. They argue that if abrupt drift can occur with a small magnitude and this abrupt drift is flowed by periods of drift, it is difficult to distinguish periods of repeated abrupt drift and other kinds of drift. Webb et al. created a taxonomy of different concept drifts and corresponding functions in their paper.

2.2.2 Approaches

Lu et al. [24] distinguish three main categories: Error-rate-based methods, data-distribution-based methods, and methods that use multiple hypothesis tests [24]. In the following paragraphs, the categories are shortly explained, and some well-known detection approaches are shortly touched upon. A more extensive comparison and summary of drift detectors can be found in [24].

Error-rate-based methods use the error rate of the model based on the true labels. The most well-known methods are DDM [16], EDDM [5], and STEPD [31]. DDM detects concept drifts by analyzing the error rate and the standard deviation. The error rate is the probability of making an incorrect prediction. If the distribution of the data is stationary, the error rate should decrease over time. If the error rate increases over time, it is assumed that the data distribution is changed. EDDM is similar, but it tracks the distance between two consecutive errors. If the error rate decreases over time, errors are less common, and the distance between two errors increases. When the distance decreases over time, this indicates that drift is present. STEPD compares the accuracy of two equally sized time windows, recent and older. It is expected that the accuracy is the same when the data is stationary. If there is a significant difference between the windows, drift is detected.

In 2018, Barros and Santos conducted a large-scale comparison of concept drift detectors [6]. The comparison was limited to error-rate-based methods. They compared the detectors in terms of accuracy in abrupt and gradual concept drift. In data sets with abrupt drifts, they measured the precision, recall, and the Matthews Coefficient metric. They compared different data set generators with different sizes of the concept drifts. Overall, no single drift detector is the best in all situations.

Distribution-based methods are used for unsupervised models. They calculate the distribution change of the features directly. If labels are present, the labels can be used as a feature or used to split the data set based on their labels. Distribution change is measured by calculating the distance. The distance is calculated with the total variation [23], or the Hellinger distance [45]. These algorithms can accurately identify the time of the drift, but the computational costs are higher than the costs of error-rate-based methods. The first formal method is proposed by Kifer et al. [23]. Kifer et al. created a family of distances: Relativized Discrepancy (RD). A different distribution-based algorithm is the Information-Theoretic Approach (ITA) [12], which uses kdqTree to partition the data into bins, then uses the KL divergence to calculate the difference in density per bin. The hypothesis testing is done via bootstrapping. A

complete list of distribution-based detection algorithms can be found in [24]. The last category is the multiple hypothesis drift detection. This category applies similar techniques to the previous categories. This category has two subcategories: parallel multiple hypothesis tests and hierarchical multiple hypothesis tests. The first subcategory uses multiple test statistics parallelly, then executes multiple hypothesis tests parallelly. The second subcategory first applies a test statistic and hypothesis test to detect drift, then applies a test statistic and hypothesis test to validate. The first algorithm belonging to the first subcategory is the Just-In-Time (JIT) adaptive classifier [2]. JIT detects the mean change in the features interested by learning systems. Another implementation is the Linear Four Rate (LFR) drift detection [43]. This detector tracks the changes in the True Positive (TP), True Negative (TN), Positive Predicted Value (PPV), and the Negative Predicted Value (NPV) rate. An advantage of LFR compared to traditional error rate detectors such as DDM is that LFR detects drift in more situations. DDM fails to detect a drift unless the sum of FP and FN changes, LFR detects a drift even though the sum of FP and FN does not change. An example of the hierarchical multiple hypothesis tests is the Hierarchical Linear Four Rate (HLFR) detector [43]. This approach uses the LFR detection approach in the detection layer, and validation layer of the HLFR is a zero-one loss.

A different way to categorise the concept drift detectors is proposed by Hu et al. [18]. The methods are divided into two categories: performance-based and data distribution-based approaches. The first category monitors metrics, such as accuracy, precision, recall, and F-measure. This category is comparable with Lu's error-rate-based category and supervised methods belong to this category. The distribution-based category uses metrics such as location, density, and range. The methods can be either supervised or unsupervised. This category is comparable to Lu's last two categories. The supervised drift detectors can be further divided into four categories: Statistical methods, windows-based methods, block-based ensembles, and incremental-based ensembles [44]. Well-known detectors in the statistical methods category are DDM and EDDM. Windows-based methods compare certain metrics, often accuracy, of two different time frames. One popular example of this is Adaptive Sliding Window (ADWIN) algorithm [8]. When two time windows show different averages, drift is assumed. The Hoeffding bound statistic determines whether the sub-windows are wide enough and whether the averages are different enough. These requirements take a lot of computational time and memory, therefore ADWIN2 was introduced, which is currently a state-of-the-art method. The main difference with ADWIN is that ADWIN2 removes an entire bucket while ADWIN deletes one element of the time window when a change is detected [9]. Block-based ensemble methods are methods with a group of classifiers. Similar to the windows-based methods, these methods process blocks of data. The first algorithm in this category is the Streaming Ensemble Algorithm [35]. It constructs classifiers based on the blocks. The classifiers are added to the ensemble. When the ensemble is full, the worst-performing classifier is removed. When concept drift occurs, the algorithm quickly discards classifiers that are trained on outdated data. Accuracy Weighted Ensemble (AWE) [42]

is similar to SEA but uses a version of the mean square error to select the best classifiers. Accuracy Updated Ensemble (AUE) [10] improves upon AWE by updating learning models directly rather than per chunk. Incremental-based ensembles process elements sequentially. The first algorithm in this category is the Dynamic Weighted Majority (DWM), where each classifier has been given a weight. The prediction of an instance is based on the prediction of each classifier and its weight. The classifiers that correctly classified an instance, get an increased weight, and classifiers with an incorrect prediction get a reduced weight. When a classifier has a weight below a certain threshold, it gets replaced by a new classifier. Another example is the Learn++ algorithm family, which uses batches of data and weighted majority voting. The first Learn++ algorithm starts with a single classifier [34]. Prediction errors are used to create a weighted distribution of all samples. A second classifier is trained with selected examples from the weighted distribution. These steps are executed until there are k classifiers. However, a disadvantage of this method is that many outdated classifiers can outvote classifiers that are trained on newer data. Additionally, it never forgets old information, which makes it hard to detect concept drift. Newer algorithms in the family, like Learn++.NSE, try to tackle this problem. This algorithm assigns more weight to newer classifiers. A complete overview of supervised drift detectors can be found in [44].

2.2.3 Gap

Through an extensive review of existing literature, Wares et al. [44] identified several shortcomings: The state-of-the-art drift detection methods are outdated, there is a lack of benchmark data sets for evaluation, metrics for the evaluation are flawed, and the inability of drift detectors to cope with additional data anomalies [44]. Hu et al. [18] state that concept drift detection follows the No Free Lunch Theorem: It is difficult to have a universal best approach. Supervised methods are more accurate but are not cost-efficient because they require labelled data. Unsupervised methods do not have labelling costs, but they are not able to detect all different kinds of concept drift. They conclude in their survey that current detection methods lack dynamic parameter tuning, where parameters change depending on the concept drift. By combining dynamic parameter tuning and improving unsupervised learning, the trade-off between performance and cost can be reduced [46].

2.3 Change Point Detection

2.3.1 Definition

Change points are abrupt variations in time series data. Time series data are sequences of data over time. Formally, a time series data stream can be defined as: $S = x_1, ..., x_i, ...$ [37]. Unlike concept drift detectors, change point detectors detect changes in the data distribution, without taking labels into account (change in $P(X)$). Change point detectors are often based on statistical decision theory. A change point detector cannot detect all change: If $P(Y \mid X)$

changes over time, but $P(X)$ remains unchanged, a change point detector will not be able to detect this.

2.3.2 Approaches

Again, the methods can be divided into online and offline approaches. The difference between these approaches is the goal: In online methods, the goal is to detect change points as soon as possible, minimizing the rate of false alarms. In offline detection, the goal is to identify all change points, while avoiding false positives and detecting true change points [13].

The simplest methods to detect change points are control charts, CUSUM algorithms, and likelihood ratio tests [7]. A control chart uses the mean and variance to create an upper and lower bound to detect outliers. CUSUM stands for cumulative sum and is a control chart. In likelihood ratio methods, the probability distributions of data before and after a possible change point are analyzed. If the distributions are significantly different, the point is marked as a change point. The difference is measured via the logarithm of the likelihood ratio between two time intervals [21].

In 2020, Truong et al. [38] published a paper reviewing offline change point detection methods. They used three selection criteria for the review: Cost function (a low cost means no change points), search method, and constraint (the penalty for complexity). The search methods are split into optimal methods and approximate methods. The first category includes the Opt and Pelt algorithm, the second category window-sliding, binary segmentation, and bottom-up segmentation. In the Opt algorithm, the number of change points is fixed, and it recursively solves sub-problems. When the amount of change points is high, algorithm PELT (Pruned Exact Linear Time) is a more suitable approach. This algorithm uses linear functions of the number of change points as linear penalties. Window sliding is a fast alternative to the optimal methods. It computes the difference between two time periods. In binary segmentation, the algorithm splits the data in the point where a change point is detected and repeats this until the number of expected breakpoints is met. Lastly, the bottom-up approach is the counterpart of binary segmentation. It splits the data into many small sub-signals and merges them until a fixed number of change points remain.

In a survey for change point detection, Aminikhanghahi and Cook [3] split methods into supervised and unsupervised methods. Supervised methods are divided into three subcategories: multi class classifiers, binary classifiers and virtual classifiers. Unsupervised methods are divided into likelihood ratio, subspace model, probabilistic methods, kernel-based methods, graph-based methods, and clustering. Often, supervised methods perform better than unsupervised methods. However, supervised data depend on the quality and quantity of training data, which is often not high in real-world data. A disadvantage of the unsupervised methods is that they are only suitable for stationary data, which makes them unusable for data with concept drift.

Aminikhanghahi and Cook [3] also classify the categories on whether they are parametric or non-parametric. Non-parametric methods gain power in computational costs and are less expensive than parametric costs when the dimensions of the data increase. Overall, they are more robust than parametric methods, because those rely on the choice of parameters.

2.3.3 Gap

The survey mentions some challenges for future work in the change point detection methods [3]: In many real-world applications, it is important that change points are detected as soon as possible. This problem requires online algorithms or smaller window sizes. Smaller window sizes create higher costs and detect more local changes. Another problem is the robustness of the algorithms: In general, it is assumed that non-parametric algorithms are more robust than parametric algorithms, but there exists no formal analysis. Additionally, most detection approaches focus on detecting the presence of change. In real-world applications not only knowing if and when change is happening but also knowing what changes and why it changes, is important. This requires the development of more accurate change analysis.

In most methods, a certain threshold value is applied to determine if a change is significant. Selecting the optimal value for this threshold is often hard, developing a statistical method to find significant change points would increase reliability.

Finally, concept drift and change point detection are very closely related. One difference between these disciplines is that concept drift uses the label in data sets to detect a change in the posterior distribution of data, while change point detection only investigates the distribution of the features. Another difference is that many algorithms for change point detection only work on stationary data, while concept drift happens in non-stationary data. By bringing these two disciplines together, change points in non-stationary data can be detected.

2.4 Anomaly Detection

2.4.1 Definitions

Anomalies or outliers are data points that do not conform to the normal behaviour [11]. Examples of anomalies can be credit card fraud or a breakdown of a system. Anomaly detection is related to noise detection, but noise is removed before analysis, while anomalies do not have to be removed. It is also closely related to novelty detection, where the main difference is that novelties will be incorporated in the model and anomalies will not. Chandola et al. [11] performed a survey of anomaly detection methods, in which they distinguish three different kinds of anomalies:

- Point anomalies: If an individual data instance is anomalous, this is considered a point anomaly. This is the simplest form of an anomaly;

- Contextual anomalies: If a data instance is anomalous in a certain context, but not in general, this is a contextual anomaly. An example is the temperature throughout the year: An observation of 30 degrees Celsius is possible in the summer but would be an anomaly in the winter. It is an anomaly in a certain context;
- Collective anomalies: If a set of related data instances is anomalous concerning the entire data set, this is called a collective anomaly.

2.4.2 Approaches

Anomaly detection methods can be divided into three categories [11]: supervised, semi-supervised, and unsupervised. In supervised methods, a predictive model is built for normal vs anomaly classes. In semi-supervised detector, it is assumed that normal data have labels and anomalies do not have labels. When no labels are available, unsupervised methods are used. These methods assume that normal data are more frequent than anomaly data (if this is not true, there is a high false alarm rate).

In a survey in 2018, Xu et al. [48] compared several state-of-the-art anomaly detectors. They categorise the methods in neighbour-based detection, sub-space detection, ensemble-based detection, and mixed-type detection. Neighbour-based detection identifies outliers by using neighbourhood information. The anomaly score is based on the average distance, weighted distanced, or the local outlier factor (LOF). These methods are independent of data distributions and are good in detecting isolated points. Due to reliance on distance measures, these detection methods do not work well in high-dimensional data. This can be solved by ranking the neighbours. Determining the right number of neighbours remains an issue in this category.

Subspace-based detection assumes that in a high-dimensional data set, the outliers can be detected by a subset of features. These methods aim to find anomalies by going through several sets of subspaces in an ordered sequence. Subspaces with an abnormally low density are considered anomalies. A challenge in these detection methods is to choose the right subspaces or base learners.

Ensemble-based detection summarizes the anomaly scores of different learning techniques or subspaces and selects the best after ranking. Feature bagging trains multiple models on different feature subsets and combines the results to decide which one is the best. Subsampling methods use training points without replacement. Random subsampling can be used to obtain the nearest neighbours for each object and get a local density.

Mixed-Type detection is used for data sets that have both numerical and categorical features. The previous categories are only suitable for numerical data. One example of such a detection is a frequent pattern-based anomaly detector. Potential anomalies are data points with infrequent patterns.

2.4.3 Gap

Ahmed et al. [1] describe multiple challenges in the discipline:

- There is a lack of effective general-purpose anomaly detection methods;
- Noise is hard to distinguish from anomalies;
- Labelled data is often not available;
- If concept drift is present, the anomaly detector may work now but not in the future;
- Abnormal activities often try to appear as normal activities (fraud).

Xu et al. [48] mention that (effectively) evaluating detection methods is hard because anomalies are rare. Another necessity is to introduce an effective distance metric that is suitable for high-dimensional and mixed-type data. Finally, for many types of anomaly detectors, parameters have to be chosen. For example, the number of neighbours in neighbour-based anomaly detection algorithms, or the number of subspaces in subspace-based methods and base learners in the ensemble-based method.

2.5 Prediction Influence

No literature is found on *prediction influence*. Prediction influence is the influence of predictions of a machine learning model on new data points. Some research fields are closely related to prediction influence. In the next paragraphs, fairness in AI, filter bubbles, and feedback loops (in recommender systems) are discussed.

In machine learning, fairness is described as "*the absence of any prejudices or favoritism towards an individual or a group based on inherent or acquired characteristics*" [27]. Unfairness in machine learning models is often described with models that accept or reject loan applications. If the model is based on data with discrimination, the model will systematically be biased against people with certain characteristics, such as race and gender. This leads to unfairly denying loans to certain groups [15]. If the bias against minorities increases over time, thus being bigger than in the original data set, this is prediction influence. More specifically, this example shows the self-fulfilling prophecy: If a group is unfairly denied a loan, then in the future the algorithm will use the information (being denied a loan in the past) to again deny a loan. Another well-known example of an unfair self-fulfilling prophecy is in the classroom: If the teacher expects high results from students, the students will perform better, and if the teacher expects low results, the students perform worse. A different example of a self-fulfilling prophecy is presented by Lum and Isaac [25]: Predictive policing software recommended to increase police presence in areas where crime data is reported. The simulation also showed that when police are present, people report more crime. In this example, black neighbourhoods ended up in a vicious circle, where people reported more crime due to higher police presents, and due to more reports, the police increased their presence.

Unfairness in algorithms either comes from preexisting bias in the data or bias emerges from the algorithm itself. Prediction influence does not have to have a preexisting bias.

Quite similar is the filter bubble in recommender systems. Pariser [33] describes the filter bubble as a mechanism where algorithms give content to users based on their interests and do not give any other content. In the context of news, this means that a user gets news stories that reinforce a certain opinion. As a consequence, the user will not get exposed to any news articles that are not aligned with his/her current opinion [26].

The effect of iterative feedback loops on algorithms has been simulated by Sun et al. [36]. They simulate a recommender system, where both the input for the user as well as the input of the algorithm are biased. The data is equally split between likes and dislikes and the bias is tested with three different forms: filter, active learning, and random. They find that when the human interaction probability is set to 1, the bias affects the algorithm performance. Moreover, by using the iterated filter bias, less data is shown that is relevant and the gap between liked and disliked items increases.

In a paper by Khritankov [22], a feedback loop was created in a housing price system. A model is trained on part of the original data. At each step, a user takes a prediction from the model and decides on a housing price. This new price is added to the end of the data stream (and the first item is removed so that the size of the data set remains the same). After a while, the model is retrained on the newer data set. For each round, the R^2 and mean absolute error (MAE) are known. The results show that if users use and adhere to the predictions, throughout the rounds, MAE decreases.

2.6 Data Generators With Concept Drift

While the detection of concept drift, change points, or prediction influence is most useful in real-world data sets, these data sets are often limited when testing a framework. It is not known when drift or change occurs, which type of change this is, and if this change is an anomaly or not. It is possible to simulate changes on real-world data, such as dropping classes, adding data, or tweaking class labels [30]. The results are not great, and therefore, synthetic data sets are generated that include drift. There are several synthetic data sets available, such as SEA concepts, STAGGER, and Mixed. In the following paragraphs, some data generation frameworks are presented.

Narasimhamurthy and Kuncheva [30] proposed a framework for simulating changing environments, including STAGGER concepts and the moving hyperplane drift. The distribution of source i is based on the class-conditional probability function $p_i(\mathbf{x} \mid \omega_j)$, where $\mathbf{x} \in \Re^n$ is a set of features and $\Omega = \{\omega_1, \ldots, \omega_c\}$ is the set of class labels, and prior probabilities $p_i(\omega_j), j = 1, ..., c$ for source $i = 1, ..., K$. At any time point, one or more sources are active. Function $v_i(t) \in [0, 1]$ is the influence of a source i at time t. The sum of the influence of all sources is always equal to 1. This means

that the distribution at time t is characterised by $P(\omega_j, t) = \sum_{i=1}^{K} v_i(t) P_i(\omega_j)$ and $p(\mathbf{x} \mid \omega_j, t) = \sum_{i=1}^{K} v_i(t) p_i(\mathbf{x} \mid \omega_j)$. The distribution D at time t can be defined as $D^{(t)} = \{v_1(t), \ldots, v_K(t)\}$. Via this framework, gradual drift can be created by using two data sources. One concept fades gradually while the other source takes over. The moving hyperplane can be simulated by setting the amount of data sources equal to the amount of time instances, and at every time t, $v_i(t) = 1$, and all other $v_j(t) = 0$ where $j \neq i$. The consecutive data sources have to be closely related. In [30], this is illustrated with several scatterplots. STAGGER concepts can also be generated, by creating three data sources, which are called 'target concepts'. The influence of the concepts is set to 1, for a specific amount of time. At the borders, one data source is immediately replaced by the next one. Lastly, recurring trends are possible by setting $D^{(t)} = D^{(t+T)}$, where T is the period of repetition.

Minku et al. [28] created a framework that generates data sets for STAGGER boolean concepts, SEA moving hyperplane concepts, circle drift, sine drift, moving hyperplane drift, and boolean drift. In their framework, irrelevant features, class noise, different levels of severity of the drift and different levels of speed of the drift can be added to the data. The training set is composed of $2N$ samples, where N is the number of time steps before the drift starts. The first N examples are generated with the old concept, then $drifting\_time$ amount of samples are used to generate the drift, the rest of the samples $(2N - N - drifting\_time)$ are generated according to the new concept. Minku et al. [28] argue that test data should have a different distribution compared to the training set. They divided the test set in two partitions, $0.25N$ composed of the old concept and $0.25N$ composed of the new concept. In this way, the model can be tested before the drift with the first partitions. After the drift starts, only instances of the partition based on the new concept are used, even when the drift is not completed. In this way, it is possible to check the performance of the model on the old and new concept.

Part II

Approach

Chapter 3

Approach

In this chapter, we explain the approach that is developed to answer the research questions stated in Section 1.1. The approach can be divided in three parts:

1. Data Generator
2. Prediction Influence Detector
3. Evaluation

In the next section, the approach for each component is described and the implementation details are discussed.

3.1 Data Generator

The data generator has to interact with the predictions of the classifier. This means that there should be a certain feedback loop between the classification model and the data generator. The classification model should request a new sample from the data generator and the data generator gets the prediction of the classification model to update itself in such a way that a prediction influence is incorporated. The steps of the data generator for the two main strategies, a self-fulfilling influence and a self-defeating influence, are explained. For both strategies, the classification model starts with a data stream without any preexisting prediction influence, because no predictions have been made. If the data stream consists of i concepts, the probability of choosing an instance from one of the concepts is equal to $1/i$. Similarly to Narasimhamurthy and Kuncheva's framework [30], each concept has its own function $v_i(t) \in [0, 1]$. If in the data stream i class labels for positive instances and j class labels for negative instances exist, then $\sum_{z=1}^{i+j} v_z(t) = 1$ at any time t. If the data is balanced, then the sum of the functions of the positive instances should be 0.5, and the sum of the functions of the negative instances should be 0.5. When new instances are classified, there are two different outcomes when applying a self-fulfilling influence:

- True Positive and True Negative: If an instance from concept i is correctly classified, the probability of choosing a new instance of the concept where the instance originated from, becomes higher. In other words, $v_i(t)$ increases.

- False Positive and False Negative: If an instance from concept i is incorrectly classified, the probability of choosing a new instance of the concept where the instance originated from, becomes lower: $v_i(t)$ decreases.

This is the purest form of self-fulfilling influence. In certain cases, the self-fulfilling influence might only be present in positive instances. Then the influence only applies to TP instances and FN instances. If the influence only occurs in negative instances, only the probability of the concepts with TN instances or FP instances changes based on predictions.

The consequence of a self-defeating influence are:

- True Positive and True Negative: If an instance from concept i is classified correctly, the probability of choosing a new instance from that concept is decreasing: $v_i(t)$ decreases.
- False Positive and False Negative: If an instance from concept i is classified incorrectly, the probability of choosing a new instance from that concept is increasing: $v_i(t)$ increases.

Similar to the self-fulfilling influence, the self-defeating influence does not have to occur in both negative and positive instances. If the influence occurs in negative instances, the concepts which have TN and FP instances will change. If the influence is only in positive instances, the probability of choosing concepts with TP and FN instances will change.

By increasing and decreasing the influence $v_i(t)$, we can simulate a gradual shift in concept densities. It should also be possible to include intrinsic drift because the world changes every day. In the implementation, intrinsic drift can be added in the input streams.

3.1.1 Implementation

The data generator is implemented in a local copy of the source code of the Python Package River [29], more specifically in `~\River\datasets\synth\prediction_influence_stream.py`. The data generator requires several input parameters, summarized in Table 3.1. The pseudo-code of a simplified version (without any update delay or weight update in batches) is shown in Algorithm A.1, which shows how a new sample in the stream is picked. Furthermore, Algorithm A.2 and Algorithm A.3 show how the weight of the streams is updated, taking into account that true labels can have a latency relative to the moment of classification.

3.2 Prediction Influence Detector

Concept drift detectors check whether the distribution of $P(Y \mid X)$ changes over time. The prediction influence detector should also check whether the distribution of the TP instances within the positive instances changes over time:

Parameter name	Explanation
streams	The data generator combines multiple streams (representing a concept). The individual streams have parameters, such as n_classes, n_features, and class_weights.
weight_correct	Multiplier for the weight of the stream that has a correctly classified instance.
weight_incorrect	Multiplier for the weight of the stream that has an incorrectly classified instance.
weight_update	Weights can be updated online (weight_update = 1) or in batches (weight_update >1). Default is set to 1.
update_delay	Feedback of instance can be immediately applied (update_delay = 1) or applied at a later time point. Default is set to 1.

TABLE 3.1: Input parameters for the data generator

$$P_u(X, Y_{pred} = + \mid Y_{true} = +) \neq P_t(X, Y_{pred} = + \mid Y_{true} = +) \tag{3.1}$$

where u and t are different time points in a data stream, and whether the distribution of the FN instances change over time:

$$P_u(X, Y_{pred} = - \mid Y_{true} = +) \neq P_t(X, Y_{pred} = - \mid Y_{true} = +) \tag{3.2}$$

The same applies to the distribution of the negative instances:

$$P_u(X, Y_{pred} = - \mid Y_{true} = -) \neq P_t(X, Y_{pred} = - \mid Y_{true} = -) \tag{3.3}$$

and

$$P_u(X, Y_{pred} = + \mid Y_{true} = -) \neq P_t(X, Y_{pred} = + \mid Y_{true} = -) \tag{3.4}$$

Our approach is based on the probability density change. A change in density over time is a good indicator of concept drift. Prediction influence is a form of concept drift, thus possibly a suitable way of detecting the influence. The detection approach will be evaluated on the data generator as well as a real-world data set. The real-world data set does not provide true labels, thus an error-rate-based approach is excluded.

3.2.1 Density-Based Approach

In our approach, the density of instances per true and predicted label per month is calculated in periods before and after the month of classification. In a data stream with a self-fulfilling influence, the density value of correctly classified instances in a month increases after classification, because similar instances become more frequent in the future. When a self-defeating influence is present,

fewer similar instances will arrive, thus the density value for correctly classified instances decreases after the moment of classification. The probability density function of a temporal period is based on instances with the same true label as the label of the instances that are fitted on the function. The density trajectories within a true class (TP and FN, and TN and FP) are expected to show a difference. A difference in the density trajectories is an indication of prediction influence.

The visualization of the approach is shown in Figure 3.1. The approach works with an online data stream, thus instances arrive over time. Depending on the length of the density trajectory, track length l, the method starts calculating density trajectories from month $l + 1$. If all instances including the true labels of that month have arrived, the instances are divided in TP, TN, FP, and FN. The instances are then scored on the density probability function of the past months. The temporal bandwidth b is at least one month but can be extended to multiple months if necessary. If the features are continuous variables, the probability density function is a Kernel Density Estimator (KDE). An example of scoring the instances is shown in Figure 3.2. In this example, the current month is 06-2010. The TP instances are scored on a KDE of one month ago, 05-2010, which is created on instances with a positive label. This process repeats itself l times. The current month is a month in the future for earlier months, thus two KDEs are created on the positive and negative instances of 06-2010. Again, four subsets of the instances of earlier months are created and scored on the KDE with the same true label. For example, if the TP of month 03-2010 are scored on the KDE created with 06-2010, the score is relatively three months after classification. Eventually, instances have a trajectory from $-l$ till l. A simplified version of a table is shown in Table 3.2. The pseudo-code of how densities are calculated is in Algorithm A.4. This algorithm is a very simplified version of the real implementation. The actual code can be found in the Github repository.

If the temporal bandwidth b is bigger than one month, the months receive a sample weight. The most recent month has a sample weight of 1, the second most recent month a sample weight of $1 - 1/b$, then $1 - 1/b * 2$, et cetera. All instances within a month have the same sample weight. A visualization of the approach with a bandwidth of two months is shown in Figure 3.3. The summary of the input parameters is shown in Table 3.3

When the density trajectories are calculated for at least one month, the trajectories can be evaluated. On every row (instance), a linear regression model is fitted based on the scores from $-l$ till 0. The linear regression is used to predict values from 1 till l. If the data stream does not have intrinsic drift nor prediction influence drift, the linear regression is expected to have a slope close to zero. The difference between predicted densities and densities in the table should also be close to zero. If a data stream contains intrinsic drift, but no prediction influence drift, the linear regression slope is bigger, but the difference between the calculated densities and the predicted densities is close to zero. If the stream has no intrinsic drift but prediction influence drift, the linear regression slope should be close to zero, but the densities

Instances	-track_length	...	-1	0	1	...	track_length
01-06-2010	...	...	KDE.fit(05-2010)	KDE.fit(06-2010)	KDE.fit(07-2010)	...	...
02-06-2010	...	...	KDE.fit(05-2010)	KDE.fit(06-2010)	KDE.fit(07-2010)	...	...
...	...	...	...	...	...	...	...
01-07-2010	...	...	KDE.fit(06-2010)	KDE.fit(07-2010)	KDE.fit(08-2010)	...	...

TABLE 3.2: Example of table where temporal bandwidth = 1

should differ from the predicted densities after classification. Finally, if the data stream contains both intrinsic drift and prediction influence drift, the linear regression line should have a bigger slope, as well as a visible difference between the predicted densities after classification and the calculated densities in the previous step. The linear regression is used to account for intrinsic drift. The pseudo-code for this part is shown in Algorithm A.5.

In the final step, a statistical test is used to check whether the differences between the predicted and calculated densities after classification within a true label (TP and FN, and TN and FP) are significantly different. The Wilcoxon Rank Sum Test is used, which is a non-parametric test to compare outcomes between two independent groups. If the p-value is below 0.05, the hypothesis that the two groups are equal is rejected. The pseudo-code of this part of the approach is shown in Algorithm A.6.

3.3 Evaluation

We evaluate the approach by detecting drift in different settings:

- No prediction influence drift and no intrinsic drift;
- No prediction influence drift with intrinsic drift;
- Self-fulfilling influence drift without intrinsic drift;
- Self-fulfilling influence drift with intrinsic drift;
- Self-defeating influence drift without intrinsic drift;
- Self-defeating influence drift with intrinsic drift.

Throughout the whole data stream, intrinsic drift and prediction influence drift are continuously present. It is not possible to include both a self-fulfilling and a self-defeating influence in one setting. The prediction influence drift is present in both negative and positive instances. After we evaluate the approach on synthetic data, the approach is applied on the Lending Club Data (see Chapter 4.2). The last step of the approach using the Wilcoxon Rank Sum Test cannot be executed, because the Lending Club Data only provides positively and negatively predicted samples. A classification model can be created based on the data, which creates the confusion matrix. However, for this master's thesis, it is out of scope.

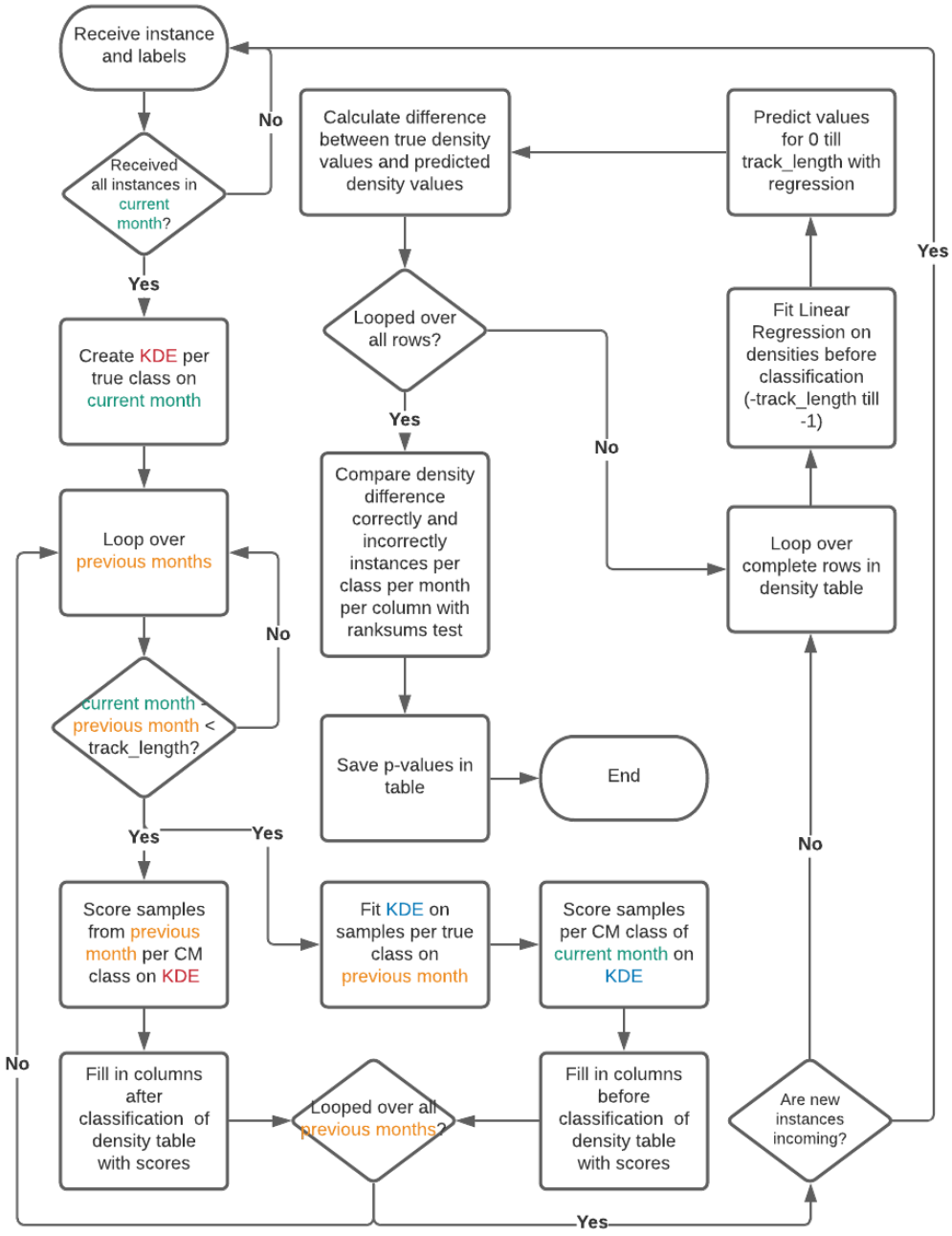


FIGURE 3.1: Simplified flow chart of density-based approach

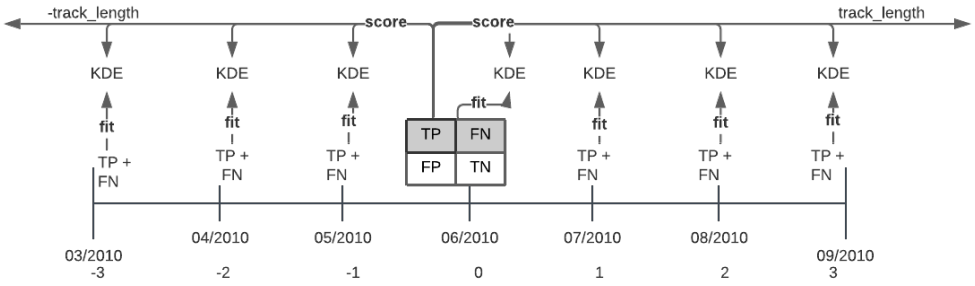


FIGURE 3.2: Visualization of creating the density trajectories where temporal bandwidth = 1

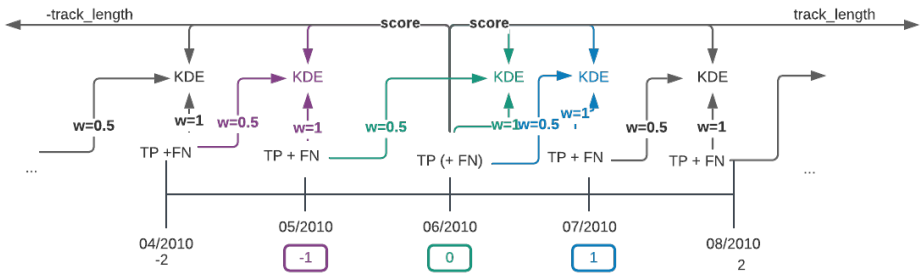


FIGURE 3.3: Visualization of creating the density trajectories where temporal bandwidth = 2

Parameter name	Explanation
temporal bandwidth	This value is used to determine how many samples should be used to fit a KDE.
track_length	Every instance is scored on KDEs for length of this value

Table 3.3: Input parameters for the density-based detection approach

3.4 Implementation

The code is written and implemented in the python package River [29]. The code for the data generator is in `~\River\datasets\synth\prediction_influence_stream.py`, the detection approach is written in `~\River\distribution\tracker.py`. The code is coming together in `~\River\evaluate\prediction_influence.py`. The scripts of the experiments are in `~\River\experiments`.

Part III

Experiments and results

Chapter 4

Experiment

4.1 Synthetic data stream experiment

In Chapter 3.1 the data generation approach is explained. This data generator is used to generate the synthetic data streams for the validation phase of the experiment. Multiple methods for stream generators are available in River, in our experiment we chose the RandomRBFGenerator. The Radial Basis Function generator is introduced by Bifet et al. [9]. The data instances are created on a fixed number of random centroids. Each centroid has a position, one single standard deviation, a (fixed) class label, and a weight. Centroids with higher weights are selected more often when generating new instances. A new instance is generated by selecting a centroid at random and offset from the central point in a random direction. Intrinsic drift can be generated by moving the centroid with a constant speed. In River, this is a parameter with a value between 0 and 1. The RandomRBFGenerator can only generate numeric features. An overview of the input parameters is given in Table 4.1.

In the experiment, the data generator uses ten RandomRBFGenerator streams, five streams with class label 0, and five streams with class label 1. The model's seed and the sample's seed are different for every stream. Every stream is based on one centroid. For the experiment, both the self-fulfilling and the self-defeating influence are generated. In both settings, the data generator is tested with and without intrinsic drift. The baseline is a data generator without an influence (stream weight multipliers are set to 1). For all experiments, the streams have two classes and one feature. Together the streams produce 65 instances per day, every month has around 2000 instances. The maximum instances per experiment are set to 71.175 instances, meaning that 36 months are covered. The track length is set to 12 months, thus the maximum output is 12 months with density trajectories if all months contain enough samples in the confusion matrix classes (leaving out the first and last twelve months). The classification model that is used is Gaussian Naive Bayes, which updates itself regularly (online). In Table 4.2, the other parameters for the experiment are shown. The weights for the self-fulfilling approach are closer to 1 because the successful streams are becoming stronger over time and do not correct over time. In the self-defeating approach, the classification model will correct itself after an unsuccessful stream gets a higher weight, thus the multiplier weights are stronger. Next to the classification model, we also use the RandomSampler

Input Parameter	Explanation
seed_model	Model's seed to generate centroids
seed_sample	Sample's seed
n_classes	Number of class labels to generate
n_features	Number of numerical features to generate
n_centroids	Number of centroids to generate
change_speed	The concept drift speed
n_drift_centroids	Number of centroids that will drift

Table 4.1: Input parameters for RandomRBFGenerator [29]

	change_speed	n_drift_centroids	weight_incorrect	weight_correct
Self-fulfilling, no intrinsic drift	0	0	0.9999	1.0001
Self-fulfilling, intrinsic drift	0.75	1	0.9999	1.0001
Self-defeating, no intrinsic drift	0	0	1.001	0.999
Self-defeating, intrinsic drift	0.75	1	1.001	0.999

Table 4.2: Parameters for the data generator

from River, in which the desired distribution is balanced. This will prevent the classifiers in the experiments with self-fulfilling influence to focus on only one class.

4.2 Real-world data set experiment

In the second phase of the experiment, the artificially generated data stream is replaced with a real-world data set. A data set with loans is an interesting topic for the prediction influence detector because people might get rejected a loan based on previously rejected applications. The data are collected by The Lending Club (`www.lendingclub.com`). This organisation is a platform where individuals can apply for a loan. The Lending Club evaluates applications and either accepts or rejects the loans. The data between 2007 and 2017 are available on `www.kaggle.com` [1]. The data are split into a rejected loan data set and an accepted loan data set. The data set with accepted loans contains $\sim$ 1.6 million applications and 150 features, while the rejected loan data set contains over 16 million loans, but only contains nine features. Turiel and Aste [39] cleaned these data and combined them in one data set. They removed the current loans, resulting in around 800.000 accepted loans and 14.2 million rejected loans. The features are reduced to debt to income (DTI) ratio, employment length, loan amount, and the purpose for the loan (binary features). A detailed description of the preprocessing and cleaning can be found in [39], and the data set can be found in [20] and on their GitHub repository [2]. A summary of the features is shown in Table 4.4. We resampled the data set to make it smaller, and removed the binary purpose features because the detection approach is focused on continuous numerical variables. Every month contains

[1] `https://www.kaggle.com/wordsforthewise/lending-club`
[2] `https://github.com/jeremyDT/P2P-lending-with-AI`

5.000 samples, starting from the month 2012-01. The proportion between the accepted and instances and rejected instances is kept the same. Finally, this results in a total of 345000 instances, of which 313602 rejected instances and 31398 accepted instances. An overview of the instances per month is shown in Table 4.3.

While the data generator allows us to have samples for the TP, TN, FP, and FN, the Lending Club data set only has the predicted class (predicted accepted and predicted rejected).

month	rejected	accepted
2012-01	4364	636
2012-02	4227	773
2012-03	4394	606
2012-04	4421	579
2012-05	4410	590
2012-06	4345	655
2012-07	4277	723
2012-08	4123	877
2012-09	4165	835
2012-10	4267	733
2012-11	4237	763
2012-12	4242	758
2013-01	4282	718
2013-02	4104	896
2013-03	4176	824
2013-04	4270	730
2013-05	4250	750
2013-06	4303	697
2013-07	4385	615
2013-08	4281	719
2013-09	4323	677
2013-10	4408	592
2013-11	3394	1606
2013-12	4264	736
2014-01	4431	569
2014-02	4355	645
2014-03	4455	545
2014-04	4418	582
2014-05	4450	550
2014-06	4502	498
2014-07	4397	603
2014-08	4468	532
2014-09	4714	286
2014-10	4250	750
2014-11	4350	650
2014-12	4769	231

month	rejected	accepted
2015-01	4516	484
2015-02	4577	423
2015-03	4595	405
2015-04	4526	474
2015-05	4606	394
2015-06	4641	359
2015-07	4550	450
2015-08	4660	340
2015-09	4702	298
2015-10	4601	399
2015-11	4699	301
2015-12	4673	327
2016-01	4782	218
2016-02	4748	252
2016-03	4748	252
2016-04	4831	169
2016-05	4809	191
2016-06	4820	180
2016-07	4868	132
2016-08	4865	135
2016-09	4898	102
2016-10	4900	100
2016-11	4909	91
2016-12	4911	89
2017-01	4940	60
2017-02	4949	51
2017-03	4959	41
2017-04	4956	44
2017-05	4970	30
2017-06	4968	32
2017-07	4979	21
2017-08	4980	20
2017-09	4995	5

Table 4.3: Rejected and accepted instances per month in the Lending Club Data set

Feature name	Type	Explanation
dti	Continuous numerical variable	Debt to income ratio of applicant
emp_length	Continuous numerical variable	Length of employment in months
issue_d	Date	Time stamp
loan_amnt	Continuous numerical variable	Loan amount of the loan requested
rejected	Binary variable (0,1)	If loan is rejected, rejected = 1, accepted = 0

TABLE 4.4: Features Lending Club data set

Chapter 5

Results

5.1 Results of the synthetic data experiment

The results section is divided into six different paragraphs for the different experiments. Per experiment a density trajectory, the corresponding density difference and, if applicable, stream weight development for one month are shown. For all months, the p-values for the Wilcoxon Rank Sum Test per true class are shown in tables. The full results are available in the appendix.

The overall accuracy of the Gaussian Naive Bayes model over months per experiment is shown in Figure 5.1. Both experiments with self-fulfilling influence show an increase of accuracy over time. The classification model of the experiment with a self-fulfilling influence without intrinsic drift performs best with an accuracy of around 68%, the model of the experiment with self-fulfilling influence and intrinsic drift follows a similar trend but ends at around 66%. Below these models, the accuracy of the models of the experiments without any prediction influence is shown. Both models show a stable accuracy over time, the model without intrinsic drift at 63%, and the model with intrinsic drift at 60%. At the bottom, the accuracy of the models of the experiment with a self-defeating influence is visible. Both models are showing an unstable development, sometimes performing better and sometimes performing worse. Overall, the model with intrinsic drift performs worse than the model without intrinsic drift. The accuracy of the models ends a bit above 50%.

5.1.1 No influence without intrinsic drift

The instances start from 1-1-2010, thus the density trajectories start after twelve months: 2011-1. The density trajectories of all months are shown in Appendix B.1, and the corresponding differences between predicted and calculated density values are in Appendix B.2. The p-values of Wilcoxon Rank Sum Test of the negative (TN and FP) instances per month are shown in Table 5.1, and the p-values of Wilcoxon Rank Sum Test of the positive (TP and FN) instances per month are shown in Table 5.2. In Figure 5.3, the density trajectory for month 2011-1 and the corresponding differences for the months after the classification are shown. The density trajectories in Figure 5.3 are relatively stable. The linear regression models (the dotted lines) have a small slope. The TN line

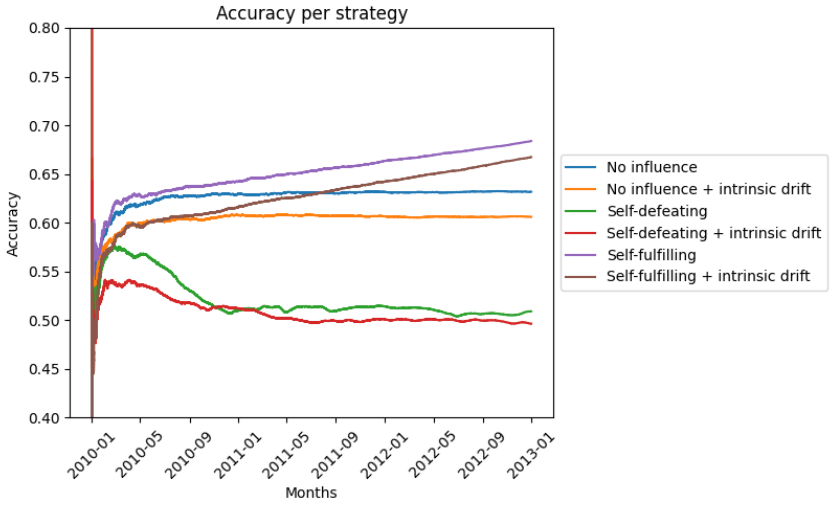
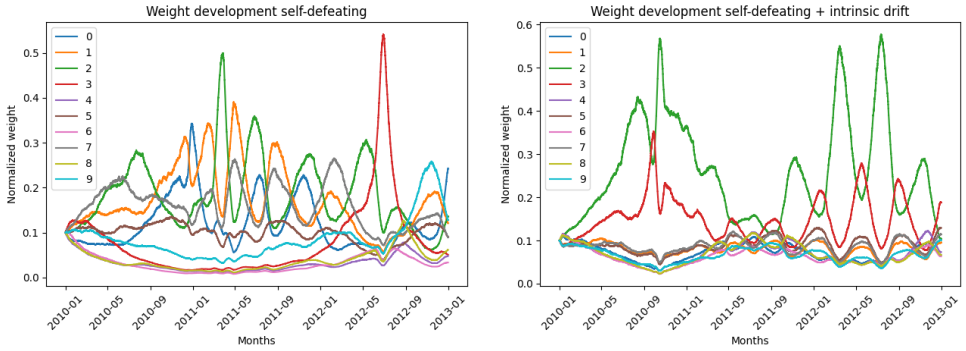
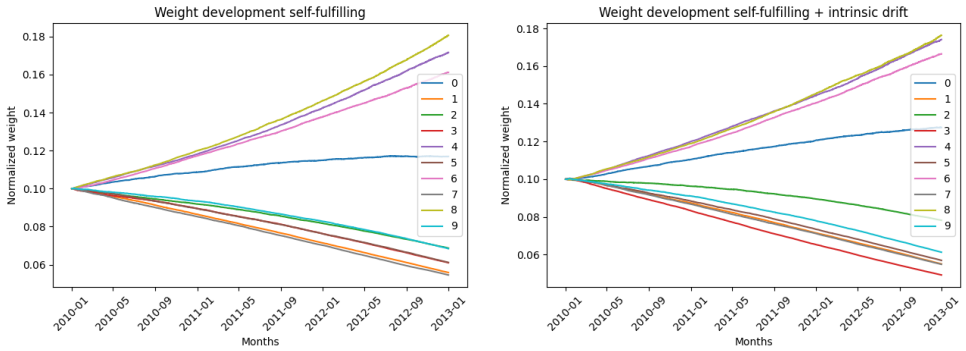


FIGURE 5.1: Accuracy over time for all experiments



(A) Self-defeating influence without intrinsic drift

(B) Self-defeating influence with intrinsic drift



(C) Self-fulfilling influence without intrinsic drift

(D) Self-fulfilling influence with intrinsic drift

FIGURE 5.2: Stream weight development in the different experiments

has the highest probability density, and the FP the lowest probability density. The classification model has adapted well to the streams that deliver negative instances. This has led to many TN instances and few FP instances. The differences between the prediction and the density trajectory are small: The

values range between 0.05 and -0.05. The corresponding p-values are shown in the first row of Table 5.1 and 5.2. For the negative instances, the p-values are all significant. For the positive instances, only the eighth month after classification shows an insignificant p-value. Overall, the tables show mostly significant p-values.

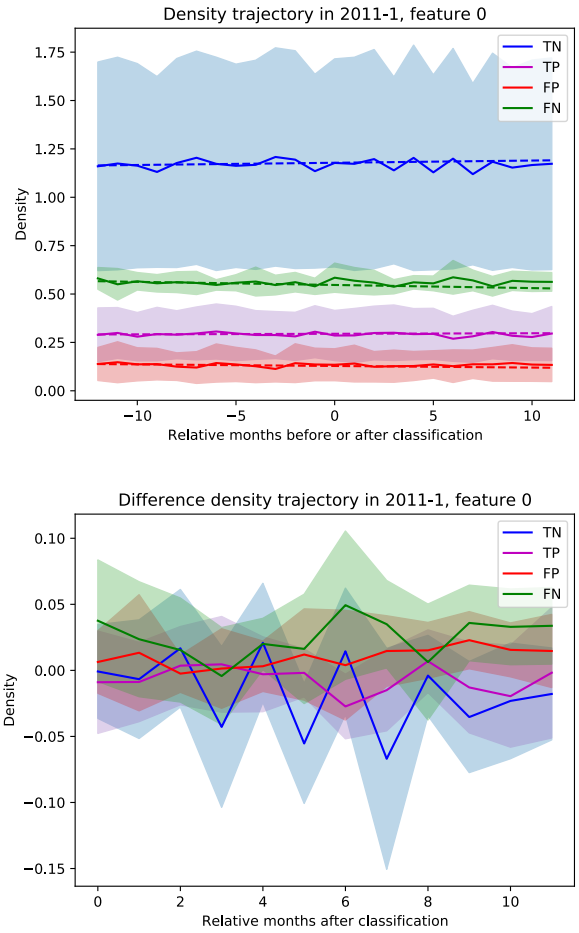


FIGURE 5.3: Density trajectory for month 2011-1, no influence without intrinsic drift

5.1.2 No influence with intrinsic drift

The p-values for the positive instances are shown in Table 5.4 and the p-values for negative instances are in Table 5.3. The density trajectories are shown in Appendix B.3, and the graphs for the differences between predicted and real density values are shown in Appendix B.4. In Figure 5.4 the density trajectory of month 2011-1 and its corresponding differences are shown. Similar to the results without intrinsic drift, all lines are relatively stable and the linear regression models have a small slope. Overall, the size per category is better distributed. The TN instances have the highest probability density, followed by the FN. The density values of the TP and FP instances are close together. The differences range from 0.03 to -0.02. The differences are more stable over time compared to the results of the experiment without intrinsic drift. The p-values that belong

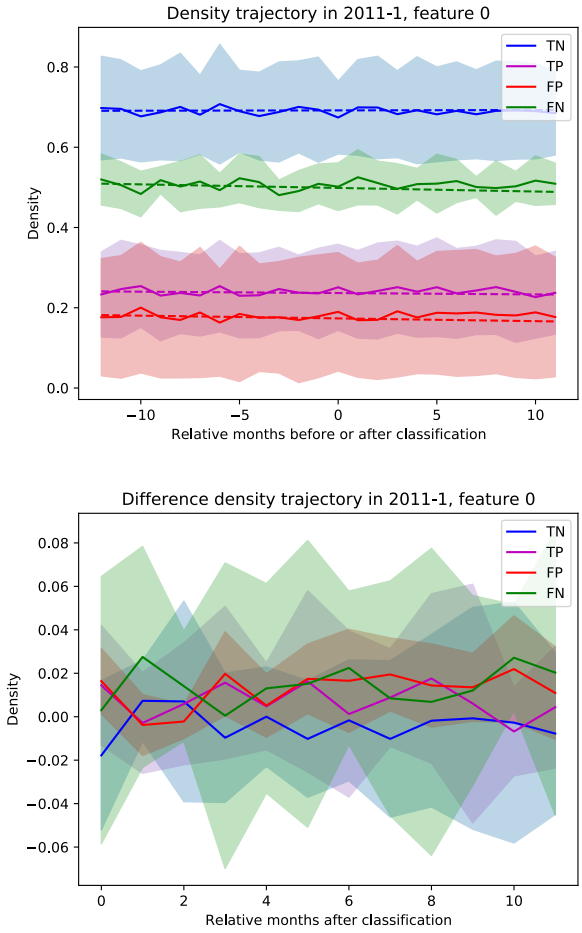


FIGURE 5.4: Density trajectory for month 2011-1, no influence with intrinsic drift

TABLE 5.1: P-values for negative instances, no influence without intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.018**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**0.024**	**<.001**	**<.001**	**<.001**	**<.001**
2011-3	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.013**
2011-4	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-5	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.405	**<.001**	**0.004**	**<.001**	**<.001**	0.392	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.006**	**<.001**	**<.001**	0.248	**<.001**	**0.016**
2011-8	**<.001**	**<.001**	**<.001**	**0.002**	0.484	**<.001**	**<.001**	**<.001**	**<.001**	0.334	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**
2011-10	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.3	**<.001**	**<.001**	**<.001**
2011-11	**<.001**	**<.001**	**0.026**	**<.001**	**<.001**	**0.001**	**<.001**	**0.002**	**<.001**	**<.001**	**<.001**	0.174
2011-12	**<.001**	0.291	**<.001**	**<.001**	0.696	**<.001**	**0.004**	**<.001**	**<.001**	**<.001**	0.258	**<.001**

TABLE 5.2: P-values for positive instances, no influence without intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.724	**<.001**	**<.001**	**<.001**
2011-2	0.326	**<.001**	**<.001**	**0.016**	**<.001**	**<.001**	**<.001**	**<.001**	0.33	**0.001**	**<.001**	**<.001**
2011-3	**<.001**	**<.001**	0.149	**<.001**	**<.001**	0.141	**<.001**	**0.01**	0.119	**0.012**	0.079	0.57
2011-4	**<.001**	**0.036**	**<.001**	**<.001**	**0.044**	**<.001**	0.054	**0.032**	**0.006**	**0.001**	**<.001**	0.067
2011-5	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.003**	**<.001**	**0.001**	**<.001**	**<.001**
2011-6	**0.002**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.002**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.003**	**<.001**	**<.001**	**<.001**	**<.001**
2011-8	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.383	**<.001**	**<.001**	**<.001**	**<.001**
2011-9	**<.001**	**0.004**	0.621	**<.001**	0.12	**<.001**	**<.001**	0.697	**<.001**	**<.001**	**<.001**	0.077
2011-10	**<.001**	**<.001**	**0.015**	**<.001**	0.9	0.127	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-11	**0.02**	0.813	**0.024**	**<.001**	0.745	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.012**	**<.001**
2011-12	**<.001**	0.091	0.869	0.619	0.891	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**

TABLE 5.3: P-values for negative instances, no influence with intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	0.309	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.075	**<.001**	0.465	0.326	**<.001**	0.442	**<.001**
2011-3	**0.004**	**<.001**	0.776	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-4	**<.001**	**0.016**	**<.001**	**<.001**	**<.001**	**<.001**	**0.005**	**<.001**	**0.002**	**<.001**	**<.001**	0.924
2011-5	**<.001**	**<.001**	0.41	**<.001**	**<.001**	**0.023**	0.254	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	**<.001**	**<.001**	**<.001**	**0.02**	**0.013**	**<.001**	**0.002**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**
2011-7	0.145	**<.001**	**<.001**	**<.001**	0.176	**<.001**	**<.001**	0.093	**<.001**	**<.001**	**<.001**	**<.001**
2011-8	**<.001**	0.087	0.735	**<.001**	0.412	**<.001**	**<.001**	**<.001**	**<.001**	0.704	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	0.073	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-10	**0.007**	0.674	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-11	0.078	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-12	**0.008**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**

to month 2011-1 are shown in the first row in Table 5.3 and in Table 5.4. For both positive and negative instances, only the fourth month has an insignificant p-value. The majority of the values are significant for both classes.

5.1.3 Self-defeating influence without intrinsic drift

The results of this experiment are shown in Appendix B.5 and B.6, and p-values for negative instances in Table 5.5 and the p-values for positive instances in Table 5.6. The stream weight development is shown in Figure 5.2a. The even numbers represent the streams that produce negative instances, the odd numbers produce positive values. Till 2010-08, the weight of the streams move either up or down, and the weight of stream 2 already moves towards a peak. From 2010-09 till 2012-02, stream 1 and stream 2 are alternating in peaks. From 2012-01 till 2012-06, stream 3 changes from a small weight to the biggest peak. In the final months, the weight of stream 9 reaches a new peak.

Whereas the results of the experiments without prediction influence are very similar over the months, the density trajectories of this experiment show a variance of results amongst the months. In Figure 5.5, the results of month 2011-10 are shown. The density trajectories of the predicted negative class are relatively stable from -12 to 0. The classification model is reasonably well in predicting instances that come from streams 0,1 and 2. When stream 3 increases its weight, the density trajectories start to deviate from the linear

Table 5.4: P-values for positive instances, no influence with intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	0.568	**0.042**	**<.001**	**<.001**	**<.001**	**0.004**	**<.001**	0.879
2011-2	**<.001**	**<.001**	**<.001**	0.647	**0.001**	**<.001**	**0.025**	**<.001**	**0.001**	**<.001**	**0.001**	**<.001**
2011-3	0.911	**<.001**	**<.001**	**<.001**	0.101	**<.001**	**<.001**	**<.001**	**<.001**	0.146	**0.018**	**<.001**
2011-4	**<.001**	**0.019**	**0.004**	**<.001**	**<.001**	**<.001**	0.915	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-5	0.094	0.753	**<.001**	**<.001**	**<.001**	**0.007**	**<.001**	**<.001**	**<.001**	**<.001**	**0.004**	0.629
2011-6	**0.044**	**<.001**	**<.001**	**<.001**	0.889	**<.001**	**<.001**	**<.001**	**<.001**	0.083	0.892	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	0.259	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	0.058	**<.001**	**<.001**
2011-8	0.249	**<.001**	0.255	**<.001**	**<.001**	**<.001**	**<.001**	**0.002**	0.38	**<.001**	**<.001**	**<.001**
2011-9	**<.001**	0.244	**<.001**	**<.001**	**<.001**	**<.001**	**0.004**	**<.001**	0.26	**<.001**	**<.001**	**<.001**
2011-10	**<.001**	**<.001**	**0.002**	**<.001**	**<.001**	**0.044**	**<.001**	**0.019**	**<.001**	**<.001**	**<.001**	0.183
2011-11	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.31	**<.001**	**0.035**	**0.02**
2011-12	**0.002**	**<.001**	**<.001**	0.089	**<.001**	**<.001**	**<.001**	0.275	**<.001**	0.531	**0.032**	**<.001**

Table 5.5: P-values for negative values, self-defeating influence without intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	0.544	**<.001**	**<.001**	0.633	**0.013**	0.108	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-3	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-4	0.525	**0.005**	0.064	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-5	**<.001**	0.558	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	0.147	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-7	0.49	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-8	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-10	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-11	**0.014**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-12	**<.001**	**<.001**	**<.001**	**0.01**	**<.001**	**<.001**	**<.001**	**<.001**	0.193	**<.001**	**<.001**	**<.001**

regression models. The probability densities of TN and TN decrease, while the probability densities of FN and FP increase. At month 11 after classification, the density trajectories get closer to the linear regression lines. This happens when the weight of stream 3 decreases again. The corresponding p-values are in Table 5.5 and in Table 5.6. In month 2011-10, all p-values for both classes are significant. Overall, the p-values for the positive instances are all significant except for two. The negative instances have some more insignificant values. Compared to the results of the experiments without prediction influence, these tables contain more significant values.

5.1.4 Self-defeating influence with intrinsic drift

The results of this experiment are shown in Appendix B.7 and B.8. The p-values for negative instances in Table 5.7 and for positive instances in Table 5.8. The weight development of the streams is shown in 5.2b. Compared to the results of the experiment without intrinsic drift, this result shows mainly two streams that are alternating in peaks. Stream 2 quickly starts to increase in weight, around 2010-09 the weight drops, allowing the weight of stream 3 to increase. Both weights decrease and from 2011-05 till 2011-09, the weight of all the streams is similar. From 2011-10 to 2013-01, the weights of streams 2 and 3 are oscillating again. If stream 2 is in a local minimum, stream 3 is in a local optimum. While all months show interesting developments, the month 2011-4

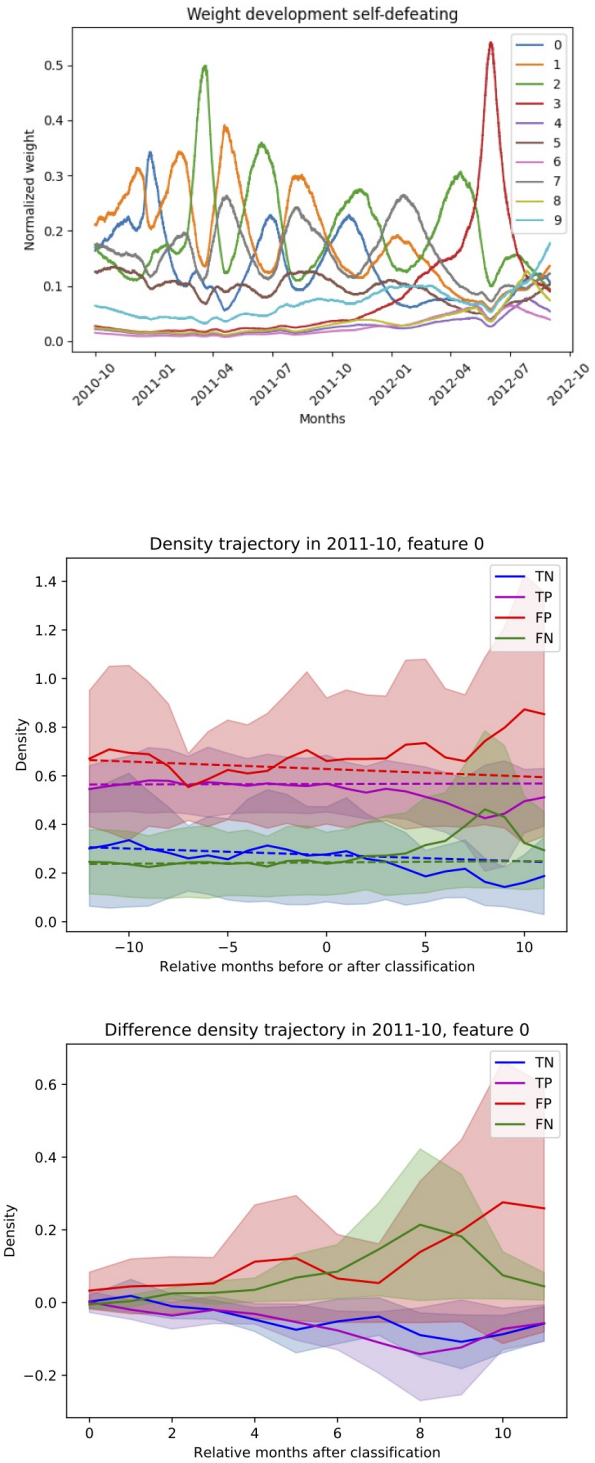


FIGURE 5.5: Density trajectory for month 2011-10, self-defeating without intrinsic drift

is chosen to be described in depth. The density trajectory, the difference, and weight development are shown in Figure 5.6. The density trajectory of the FN is very stable, and the linear regression model has a small slope. The TN and FP follow a similar trend: The instances in the month 2011-4 score quite well

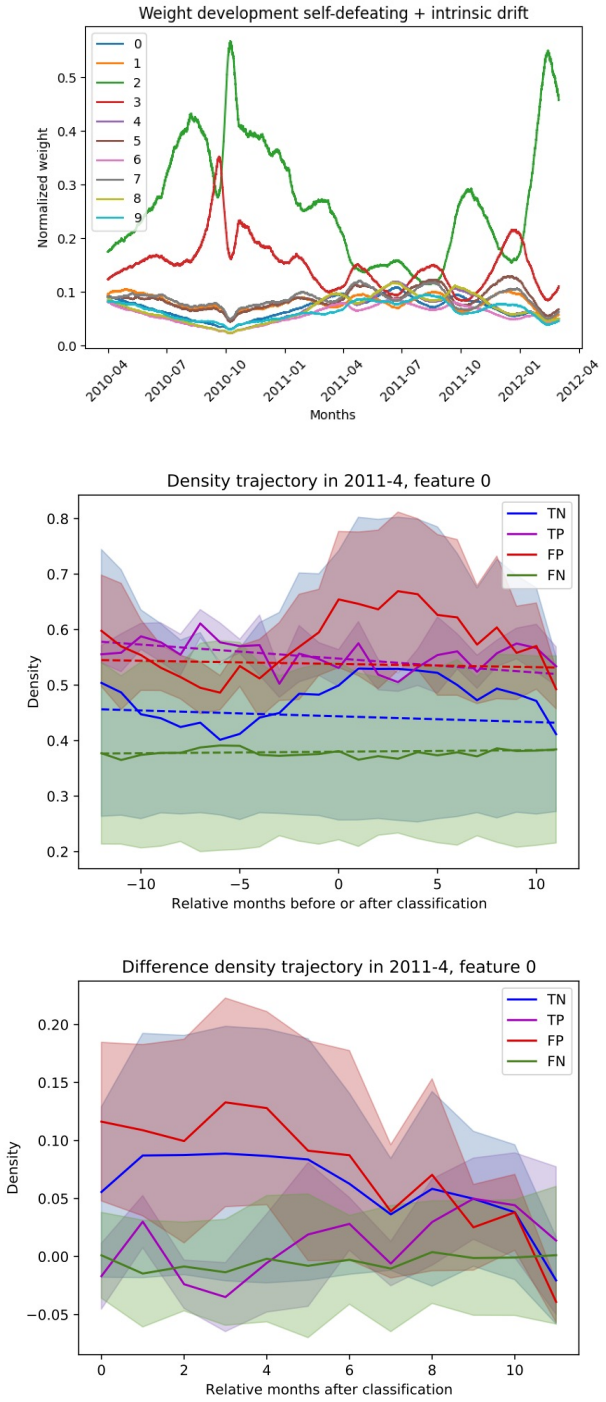


FIGURE 5.6: Density trajectory for month 2011-4, self-defeating with intrinsic drift

twelve months before classification, then when the weight of streams 2 and 3 increase, the probability densities decrease. When the weights of the streams decrease, the probability densities increase. And finally, from the month 2011-9 this pattern occurs again. Several p-values for this month are significant. Overall, compared to the results of the experiment without intrinsic drift, the results of this experiment contain more insignificant values, especially in the later months of the positive class.

TABLE 5.6: P-values for positive instances, self-defeating influence without intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-3	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-4	0.081	**0.007**	**<.001**	**0.002**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-5	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-8	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-10	**0.007**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-11	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.003**
2011-12	0.302	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**

TABLE 5.7: P-values for negative instances, self-defeating influence with intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-3	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-4	**<.001**	0.578	0.437	**<.001**	**<.001**	0.603	**<.001**	0.185	**0.03**	**<.001**	0.803	**<.001**
2011-5	0.179	0.222	**<.001**	**<.001**	**0.003**	0.075	**<.001**	0.056	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	0.073	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.337	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**	**<.001**	**0.002**	0.076
2011-8	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.835	**<.001**	**0.002**	**<.001**	0.427	**0.039**	**0.043**
2011-9	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.022**	**0.049**	**0.001**	0.62	0.426	**0.013**	**<.001**
2011-10	**<.001**	**<.001**	**<.001**	**<.001**	0.558	0.586	0.877	0.123	**0.009**	0.529	0.403	**0.034**
2011-11	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-12	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**

TABLE 5.8: P-values for positive instances, self-defeating influence with intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-3	**<.001**	**<.001**	0.918	**<.001**	**<.001**	**<.001**	**0.001**	**0.001**	**<.001**	**<.001**	0.066	0.389
2011-4	**<.001**	**<.001**	**<.001**	**<.001**	0.462	**<.001**	**<.001**	**0.006**	**<.001**	**<.001**	**<.001**	**0.019**
2011-5	**<.001**	**<.001**	**<.001**	0.546	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	**<.001**	**<.001**	0.673	**<.001**	**<.001**	0.225	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	**<.001**	0.082	**0.019**	**<.001**	**<.001**	**<.001**	0.833	**<.001**	**<.001**
2011-8	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**<.001**	0.71	0.134	**<.001**	**<.001**	0.957	0.825	**<.001**	**<.001**	**<.001**
2011-10	**0.007**	**0.001**	0.259	0.507	0.516	**0.008**	0.144	0.822	0.42	**0.001**	0.45	0.76
2011-11	**0.004**	0.963	**<.001**	0.423	0.171	0.713	**0.012**	0.723	**<.001**	0.497	0.414	0.639
2011-12	0.887	**<.001**	0.3	**<.001**	0.586	**<.001**	0.176	**<.001**	**0.015**	**<.001**	**0.001**	**0.001**

Table 5.9: P-values for negative values, self-fulfilling influence without intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**0.006**	**<.001**	**<.001**	**0.012**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-3	**0.011**	**<.001**	0.066	**<.001**	**0.001**	**<.001**	**0.011**	**<.001**	**<.001**	**0.001**	0.057	**<.001**
2011-4	0.914	0.063	**<.001**	**0.003**	**<.001**	**<.001**	**<.001**	**<.001**	0.262	**<.001**	**<.001**	**<.001**
2011-5	0.448	**<.001**	**<.001**	**<.001**	0.058	**<.001**	0.096	**0.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-6	**<.001**	0.728	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-7	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-8	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**0.001**	0.431	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-10	**<.001**	**0.004**	**0.025**	0.806	**<.001**	**<.001**	**0.009**	0.24	**<.001**	0.455	**<.001**	**0.025**
2011-11	**0.004**	**<.001**	0.1	**<.001**	**<.001**	0.628	**<.001**	**<.001**	**<.001**	**<.001**	**0.045**	**<.001**
2011-12	**<.001**	**<.001**	0.719	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.206	0.642	**<.001**	**<.001**

Table 5.10: P-values for positive instances, self-fulfilling influence without intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.011**	**<.001**
2011-3	**0.007**	**<.001**	0.923	**<.001**	**<.001**	0.103	**<.001**	**<.001**	**<.001**	**<.001**	0.321	**<.001**
2011-4	0.657	0.305	**<.001**	**<.001**	**0.013**	**<.001**	0.285	**<.001**	**<.001**	**0.026**	0.77	0.799
2011-5	0.104	**<.001**	**<.001**	0.12	**<.001**	0.148	**<.001**	**<.001**	0.306	**0.006**	0.332	**<.001**
2011-6	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.027**	**<.001**	**<.001**	**<.001**	0.179
2011-7	**<.001**	0.833	**<.001**	**<.001**	**<.001**	**<.001**	**0.005**	**<.001**	**<.001**	**<.001**	0.899	0.958
2011-8	**0.002**	**<.001**	**<.001**	0.579	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**0.025**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**
2011-10	**<.001**	**<.001**	**<.001**	**0.007**	**0.026**	**<.001**	**<.001**	0.07	**0.001**	**<.001**	**0.001**	0.742
2011-11	**<.001**	**<.001**	0.517	0.053	0.533	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-12	**<.001**	0.058	0.194	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.49	**<.001**	**<.001**

5.1.5 Self-fulfilling influence without intrinsic drift

The results of this experiment are shown in Appendix B.9 and B.10. The p-values for negative instances in Table 5.9 and for positive instances in Table 5.10. The development of the stream weights is shown in Figure 5.2c. Over time, the weights of streams 0, 4, 6, and 8 (all producing instances with a negative label) increase, and the other weights decrease. The density trajectory and difference of month 2011-2 is shown in Figure 5.7. The density trajectory looks similar to the density trajectory of the experiments without prediction influence. The TN density trajectory is the highest with a value of around 1.25. It has a small positive slope. The difference between predicted and real trajectories is small for all categories. The differences range from 0.025 to -0.02. The p-values for this month are all significant. Overall, insignificant values are spread over Table 5.9 and 5.10. No clear trend is discovered.

5.1.6 Self-fulfilling influence with intrinsic drift

The results of this experiment are shown in Appendix B.11 and B.12. The p-values for negative instances are in Table 5.11 and for positive instances in Table 5.12. The development of the stream weights in Figure 5.2d is very similar to the development of the weight development of the results of the experiment without intrinsic drift. The same streams develop a higher weight over time. The weight of stream 2 decreases slower than in the results of the experiment

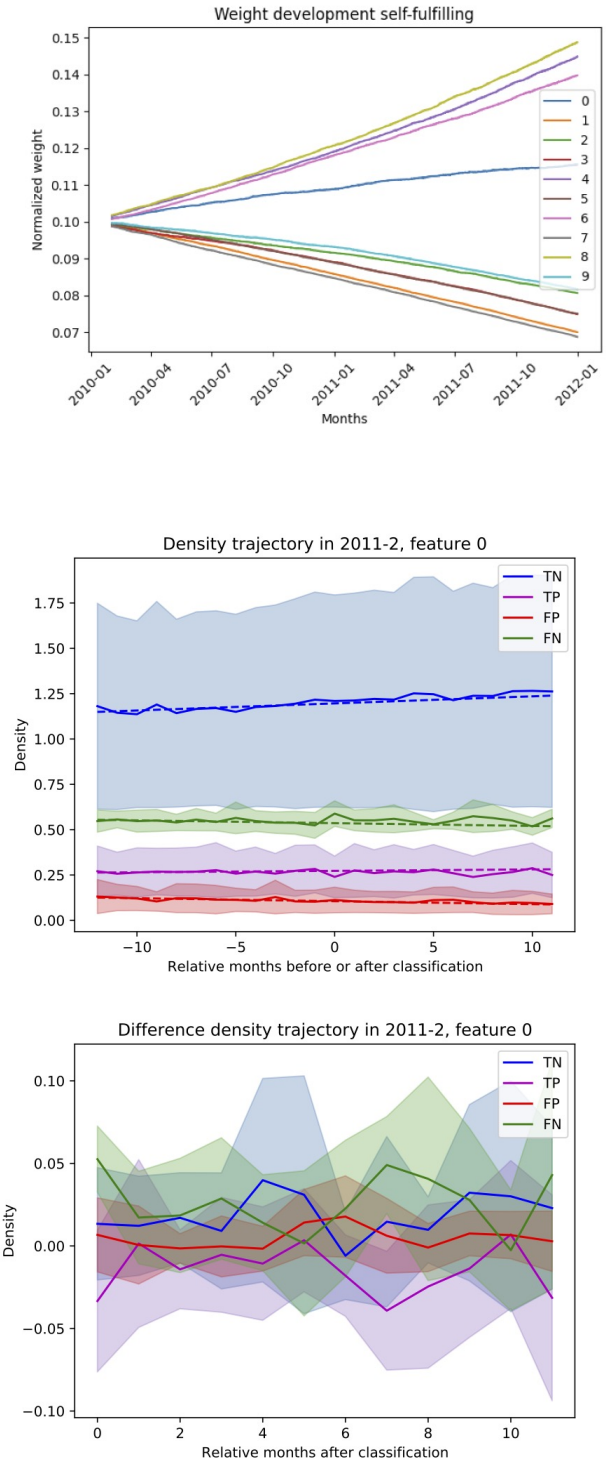


FIGURE 5.7: Density trajectory for month 2011-2, self-fulfilling without intrinsic drift

without intrinsic drift. Again, the month 2011-2 is chosen to be described in more depth. The trajectory is shown in Figure 5.8. Compared to the results of the experiment without intrinsic drift, this experiment seems to have a better distribution amongst the confusion matrix classes. The trajectory densities are

Table 5.11: P-values for negative instances, self-fulfilling influence with intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**0.031**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	0.109	**0.024**	**<.001**	**0.004**	**<.001**	0.789	**0.004**	**<.001**	**0.037**	**0.011**	**<.001**	**<.001**
2011-3	0.95	**0.007**	**0.001**	**0.007**	**0.02**	**<.001**	**<.001**	**0.015**	0.56	0.108	**<.001**	**<.001**
2011-4	**<.001**	**<.001**	0.926	**0.01**	**<.001**	**<.001**	0.098	0.177	**<.001**	**<.001**	**0.007**	**0.019**
2011-5	**<.001**	0.16	**0.003**	**<.001**	**<.001**	**0.048**	**<.001**	**<.001**	**<.001**	0.21	**0.006**	**<.001**
2011-6	**<.001**	0.083	**<.001**	**<.001**	**0.004**	0.716	**0.001**	**0.001**	0.083	0.068	**0.001**	0.163
2011-7	0.911	**<.001**	**<.001**	0.859	**0.004**	**<.001**	**<.001**	**0.01**	**0.022**	**<.001**	**0.004**	**<.001**
2011-8	**<.001**	**<.001**	**<.001**	0.682	0.128	0.141	**<.001**	0.907	0.192	**0.003**	0.098	**<.001**
2011-9	**<.001**	**<.001**	0.084	**0.006**	0.103	**<.001**	0.056	0.079	**<.001**	**0.001**	**<.001**	0.564
2011-10	**0.009**	**<.001**	**<.001**	**<.001**	0.227	0.062	**<.001**	**<.001**	**<.001**	0.557	**<.001**	**0.031**
2011-11	**0.018**	**<.001**	**<.001**	**<.001**	**0.026**	**<.001**	**<.001**	**<.001**	0.415	**<.001**	**0.004**	**<.001**
2011-12	**<.001**	**<.001**	0.817	0.392	**<.001**	0.056	**<.001**	**<.001**	**0.003**	0.442	**<.001**	**<.001**

Table 5.12: P-values for positive instances, self-fulfilling influence with intrinsic drift

	0	1	2	3	4	5	6	7	8	9	10	11
2011-1	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-2	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.011**	**<.001**
2011-3	**0.007**	**<.001**	0.923	**<.001**	**<.001**	0.103	**<.001**	**<.001**	**<.001**	**<.001**	0.321	**<.001**
2011-4	0.657	0.305	**<.001**	**<.001**	**0.013**	**<.001**	0.285	**<.001**	**<.001**	**0.026**	0.77	0.799
2011-5	0.104	**<.001**	**<.001**	0.12	**<.001**	0.148	**<.001**	**<.001**	0.306	**0.006**	0.332	**<.001**
2011-6	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.027**	**<.001**	**<.001**	**<.001**	0.179
2011-7	**<.001**	0.833	**<.001**	**<.001**	**<.001**	**<.001**	**0.005**	**<.001**	**<.001**	**<.001**	0.899	0.958
2011-8	**0.002**	**<.001**	**<.001**	0.579	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**	**<.001**	**<.001**
2011-9	**<.001**	**<.001**	**0.025**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**0.001**
2011-10	**<.001**	**<.001**	**<.001**	**0.007**	**0.026**	**<.001**	**<.001**	0.07	**0.001**	**<.001**	**0.001**	0.742
2011-11	**<.001**	**<.001**	0.517	0.053	0.533	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**
2011-12	**<.001**	0.058	0.194	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	**<.001**	0.49	**<.001**	**<.001**

quite stable over time. The linear regression model for TN increases a bit and the regression for FN decreases a bit. For the positive instances, all p-values are significant, for the negative instances months 0 and 5 are not significant. Overall, Table 5.11 and Table 5.12 contain relatively many insignificant p-values compared to the other experiment results.

5.2 Lending Club data results

The density trajectories for the features employment length, loan amount, and debt-to-income (dti) are presented in this section. The complete results can be found in the appendix. Per section, the results of a specific month per feature are described.

5.2.1 Employment Length

The density trajectories for the feature employment length are shown in Appendix C.1, the differences between the predicted densities and calculated densities are shown in Appendix C.2. In general, the density trajectories of the accepted loan applications are very stable at a density value of 0.4. The density trajectories of the rejected loan applications are at a density value of around 1.5. The trajectory has a dip starting in the month 2015-11. This dip is already visible starting in the density trajectory of the month 2015-01.

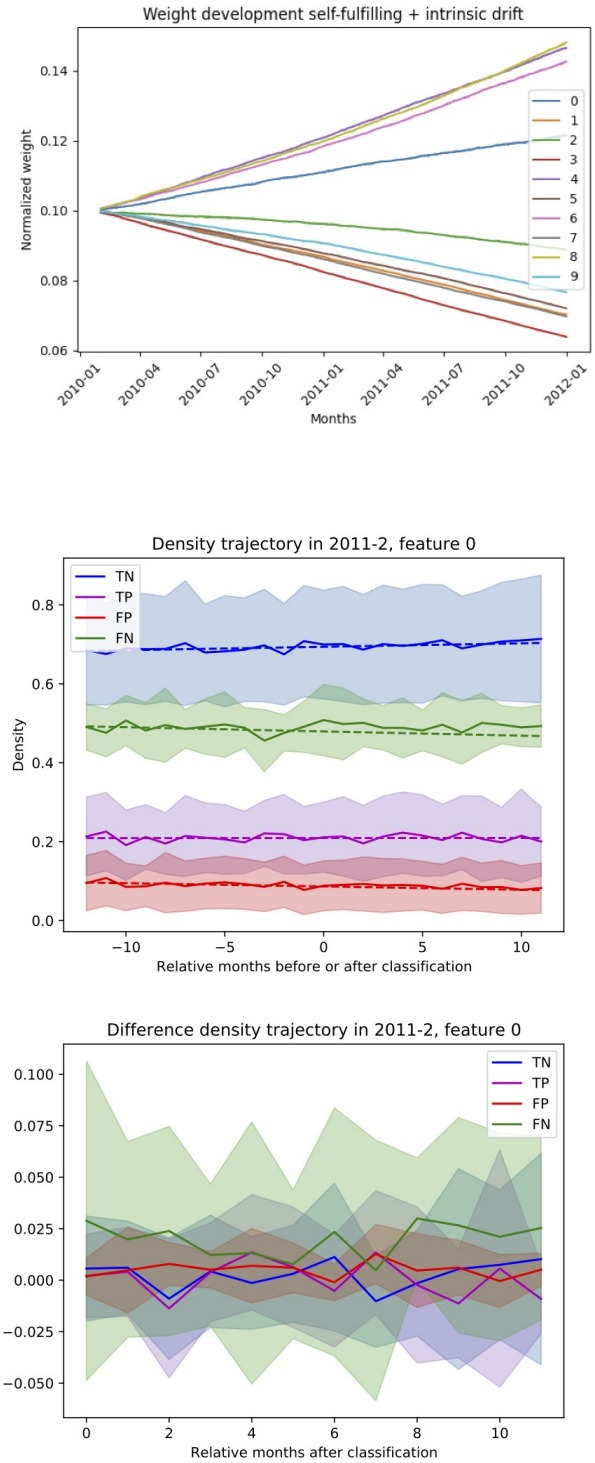


FIGURE 5.8: Density trajectory for month 2011-2, self-fulfilling with intrinsic drift

The difference between the predicted and real density ranges between 0.05 and -0.3. Especially the difference for the rejected loan applications shows peaks and troughs throughout the months. The density trajectory for month 2015-10 is highlighted in Figure 5.9. In the density trajectory of the rejected

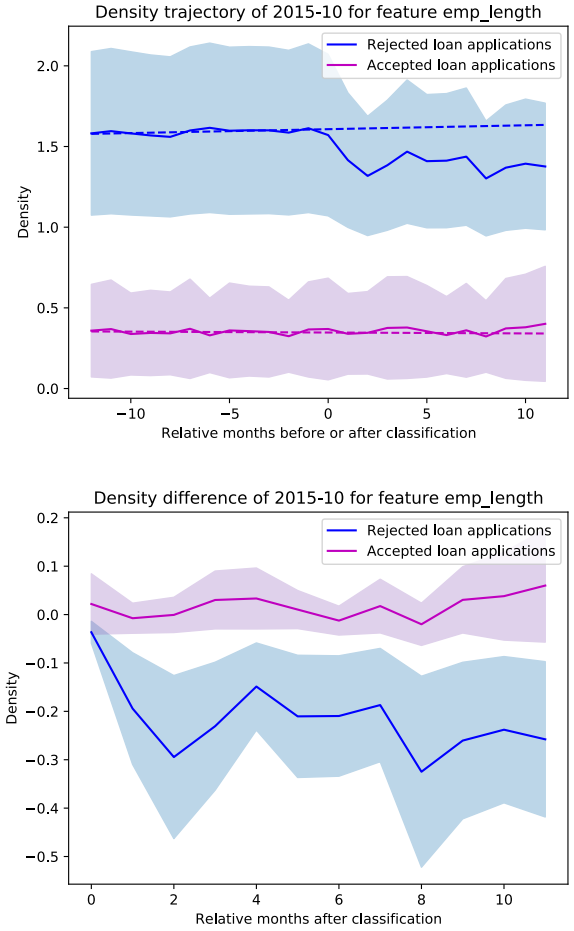


FIGURE 5.9: Density trajectory and difference for feature employment length for month 2015-10

loan applications, a stable line from twelve months before the classification until the classification month self is visible. After the classification, the density trajectory deviates from the linear regression model predictions. This change in the density trajectory is an indication that the decision rule for employment length might have changed for the rejected loans. The trajectory of the accepted loan applications is stable before and after classification. The change in rejected loans did not have an impact on the density of the employment length of the accepted loans.

5.2.2 Loan Amount

The density trajectories of the feature loan amount are shown in Appendix C.3. The differences between the calculated and predicted densities are shown in Appendix C.4. Overall, the trajectory of the rejected loan applications is stable with a density of around 0.12, and the density of the accepted loan applications is around 0.05. On average, the linear regression models have a small slope. The density differences for the rejected loan applications are close to 0 throughout all months. The density trajectories of month 2016-01 are shown

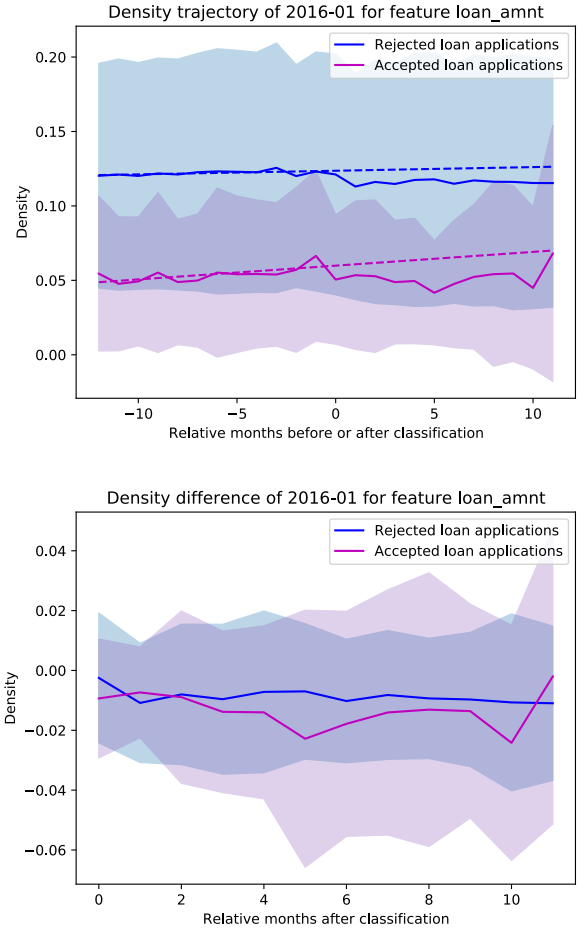


FIGURE 5.10: Density trajectory and difference for feature loan amount for month 2016-01

in Figure 5.10. The linear regression model of the rejected loans is fitted almost perfectly on the density trajectory. The density trajectory decreases after the classification month and is stable at the lower density value. The accepted loan applications have a linear regression model with a positive slope, but the real density trajectory does not show a positive slope after classification. The difference in density for the rejected loan applications is stable throughout the months, the accepted loan applications show more differences between months after classification.

5.2.3 Debt-to-income Ratio

The density trajectories for the dti feature are shown in Appendix C.5. The differences between the trajectories and the predicted trajectories are shown in C.6. The accepted loan applications have a very stable trajectory over the months. From 2014-12 till 2016-06, a small peak is showing one month before classification. In the trajectories for rejected loan applications, since month 2013-11 the area for the standard deviation decreases. This is already visible since the graph of 2013-03. The smaller standard deviation happens when

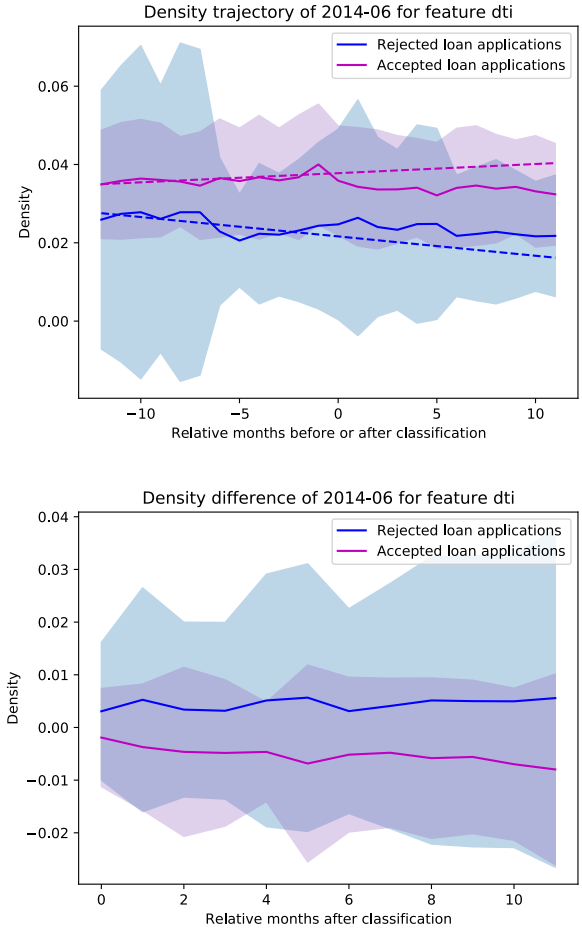


FIGURE 5.11: Density trajectory and difference for feature dti for month 2014-06

the density decreases to 0.02. In later months, the trajectory of rejected loan applications stays stable at around 0.02. The differences range from 0.01 to -0.01, only in the last month a big difference of almost -0.05 is shown. The month 2014-06 is picked to be described in more depth. The density trajectories and differences are shown in Figure 5.11. An interesting pattern is visible: The linear regression model of the rejected loan applications has a negative slope, while the slope of model based on the accepted loan applications shows a positive slope. The true density trajectories both deviate from the expected values and have close to no slope. The standard deviation of the rejected loan applications is big in -12 till -5 but decreases in size quickly. In the graph with the differences, both lines are very stable at around 0.05 and -0.05.

Part IV

Conclusion

Chapter 6

Discussion

6.1 Interpretation of the results

The concept of prediction influence is relatively new, and a detection approach has never been developed before. In this thesis, we tried to find a method that can detect prediction influence. The evaluated approach calculates the density trajectory of months before and after classification per confusion matrix class. The density trajectories are based on a KDE fitted on the same true class. A linear regression model is created based on the density trajectory before the classification happens. The differences between the calculated densities and predicted densities are compared within each true class (TP and FN, and TN and FP). If the differences are significantly different, this could be an indication that prediction influence is present. The evaluation of this method is done on synthetically generated data, and data from the Lending Club. For the Lending Club data, the last step of comparing the difference within each class is not performed, because of the lack of true labels. In this section, the results in Chapter 5 are discussed.

6.1.1 Synthetic Experiment

No influence

Two different versions of the data streams are conducted without synthetically created prediction influence: with and without intrinsic drift. We expected to find no difference for the density difference within the same true class. The intrinsic drift should be accounted for by the linear regression. In the performance of the classification models, the model without intrinsic drift has an overall accuracy of around 63%, and the model with intrinsic drift has an overall accuracy of 60%. This difference is as expected: the model performs worse with intrinsic drift. Comparing the density trajectories in Appendix B.1 and Appendix B.3, the linear regression models without intrinsic drift have a smaller slope than the linear regression models with intrinsic drift. We would expect that the linear regression models in the experiment without intrinsic drift would have a slope close to zero because the feature values do not change over time. The density differences shown in Appendix B.2 and Appendix B.4 do not show big differences. This is as expected because the intrinsic drift is compensated by the larger slopes in the linear regression. The p-values in Table

5.1, Table 5.2, Table 5.3, and Table 5.4 all contain mostly significant p-values. In the results of these experiments, we would have expected that the correctly and incorrectly predicted values within a class would not be significantly different. The results show a different picture.

Self-defeating influence

The two different experiments of the self-defeating influence were both performing quite badly in terms of accuracy, just above 50% for the classification model with intrinsic drift and around 51% for the model without intrinsic drift. This type of influence harms the performance of a model. The intrinsic drift creates an even worse accuracy, which is as expected. The weight development of the streams is shown in Figure 5.2a and Figure 5.2b. This development is what we expected: If one stream creates instances that are hard to predict, the weight increases. After a while, the model will adjust itself so that the instances will be correctly predicted. This creates room for another stream to get an increasing weight. The experiment with intrinsic drift has two main streams, while the experiment without intrinsic drift has multiple streams that are alternating in peaks. Possibly, intrinsic drift reduces the chance for streams with low weights to peak. The density trajectories, which are shown in Appendix B.5 and in Appendix B.5 are looking similar. The pattern of the weight development is also very similar, this is what we would have expected. The p-values in Table 5.5, Table 5.6, Table 5.7, and Table 5.8 are mainly small and significant. This is not different compared to the results of the experiments without prediction influence, which we would have expected.

Self-fulfilling influence

Both results of the experiments, with and without intrinsic drift, show an increasing trend in accuracy over time. This is an expected result. The streams that deliver instances that can be correctly classified are chosen more often over time. The weight development in Figure 5.2c and Figure 5.2d show a similar development. The stream weight development in Figure 5.2c shows three weights increasing over time, one stream increasing very slowly, and five streams decreasing in weight over time. The development in Figure 5.2d looks similar, but the weight of stream 0 increases a bit more and the weight of stream 2 has a slower decline compared to the weight of stream 2 in the graph without intrinsic drift. The classifier adapts to these streams, which explains why the classification model performs better without than with intrinsic drift. The density trajectories for the self-fulfilling influence without intrinsic drift show linear regression models with a small positive slope. If similar successful feature values are chosen, a positive slope is expected. The density trajectories for the experiment with intrinsic drift fit linear regression models that have a slope close to zero. Due to the weight development, many negative instances are chosen, which leads to an overall high density for TN and FN. The p-values in Table 5.9, Table 5.10, and Table 5.12 mainly show small significant values. The p-values in Table 5.11 show relatively many values that are not significant.

The expectation was to find many significant p-values, but apparently, the self-fulfilling influence is hard to detect.

6.1.2 The Lending Club

For the lending club data, only the density trajectories have been created. The data set lacks the true label, and therefore it is not known which instances are correctly or incorrectly classified. For the feature employment length, it shows that the trajectories of the rejected instances do not follow the predictions of the linear regression models. The feature loan amount has trajectories that follow the linear regression models. In the last few months, the accepted loan applications start to differ from the linear regression models. The dti feature shows for both the accepted and rejected loan applications a difference between the regression lines and the true density trajectory. These results indicate that some drift is happening in the data. Unfortunately, it is hard to find the cause. One of the possibles causes could be a change in business decision rules or a change in definition.

6.2 Implications

The results do not show a clear difference between the results of the experiments with or without prediction influence. This means that the way the approach is currently implemented and tested, cannot detect prediction influence. However, there are differences in the density trajectories of the experiments. The current approach does not detect these differences, but it is a promising starting point for future research. The weight of the streams in the data generator show lines that are expected in the different settings. In future research about prediction influence, this data generator can be a good generator when testing approaches on synthetic influences.

6.3 Limitations

In this thesis, an approach for a relatively unresearched topic is developed and tested. Due to this nature, the thesis is subjected to a few major limitations, which are discussed in the following list:

- **Parameters**: Both the data generator and the detection approach have several parameters (see Table 4.1 and Table 4.2). Due to the limited amount of time available in a thesis project, it was not possible to evaluate all possible combinations of these parameters. A consequence of this is that other parameter values could have given different results, leading to a different discussion and conclusion. Furthermore, a month was chosen as a time unit for the temporal bandwidth and track_length. In the synthetic experiment, all months contain around 2000 instances (65 instances per day). It is a possibility that the prediction influence is visible within a month and not between months.

- **Self-fulfilling influence**: The self-fulfilling influence in our approach has several limitations: A successful stream gains more and more weight, which will lead to a very unbalanced data stream. Our detection approach requires instances of every confusion matrix class, thus the data generator with a self-fulfilling approach does not work with the detection approach over a longer time. In this thesis, the multipliers for the weights have been set to very small values, and a random sampler has been added to be able to predict labels of both classes. These solutions lead to instances in all confusion matrix classes, but the problem is not solved and is fundamental.
- **Lack of true labels in Lending Club**: The Lending Club data provides predicted labels by dividing the instances into accepted and rejected. The accepted loans can be divided into correctly and incorrectly classified by checking whether the loans are completely paid back. The rejected loans do not have this possibility. The information on accepted loans being paid back is outside of the scope of this thesis. Therefore, for both labels, the detection approach cannot be fully executed. Even the density trajectories are different compared to the synthetic experiments. The KDEs are fitted on the same subset as it is scored. In the synthetic experiment, the subset that is used to fit the KDE (complete true class) is bigger than the subset that is scored (one confusion matrix class).

Chapter 7

Conclusion

In this chapter, the contents of the thesis are briefly summarized. The problem statement is defined in Chapter 1, the related literature is described in Chapter 2, the detection approach is defined in Chapter 3. In short, per month instances are scored on KDEs based on different months. A linear regression model is fitted on the scores that are created on KDEs that are fitted on previous months. The predicted values of linear regression model in future months are compared with the densities scored on KDEs fitted on future months. The differences between predicted and the densities in the trajectories are compared within the true class (TP and FN, and TN and FP). By using linear regression models, (linear) intrinsic drift is accounted for. If the difference within the true class differs, this is an indication that a shift within a true class takes place. The approach is tested on synthetic streams with and without prediction influence, and with and without intrinsic drift. The results of these experiments are described in Chapter 5. The results showed different density trajectories, but for all results, the output of comparing the density difference between the correctly and incorrectly classified samples was very similar. Using the current parameters, the detection approach is not detecting prediction influence. Next to synthetic data, a part of the approach is tested on real-world data. The Lending Club Data is down-sampled to 5000 instances per month. The density trajectories of the individual features are shown in Section 5.2.

In Section 1.1, multiple research questions were stated. The questions are addressed and answered here.

> RQ: *To what extent can we distinguish between intrinsic concept drift and prediction influence drift in a binary data set?*

This question is answered by testing the detection approach with multiple scenarios. The detection approach can distinguish between intrinsic drift and prediction influence drift if the results of the scenarios with prediction influence with and without intrinsic drift are similar. The scenarios without prediction influence with and without intrinsic drift should not detect any prediction influence. Unfortunately, the results of all experiments look very similar. The density trajectories of the different experiments look different. To answer the research question, the current method cannot distinguish between intrinsic concept drift and prediction influence drift. However, the density trajectories look promising.

SRQ1: *How can we detect the presence of drift?*

This sub-research question is answered in the Related Literature in Chapter 2. The related literature focuses on three different categories: concept drift detection, change point detection, and anomaly detection. Within the two categories, two main methods can be distinguished: detection approaches based on error rates and detection approaches based on distribution. Since the start of drift detectors, the approaches have not improved much [44]. State-of-the-art detectors are still not able to detect different types of drift.

SRQ2: *How can we simulate the interaction between a machine learning model and a data generator?*

The data generator described in Section 3.1 tries to simulate the interaction between a model and a data generator. The interaction is simulated by giving individual streams a weight. The starting weight of all streams is set to 1. When a self-fulfilling influence is applied, the weight of a stream that contains correctly classified samples increases. The weight of a stream that contains incorrectly classified samples decreases. For the self-defeating approach, it is the other way around: weights of streams with wrongly classified samples increase, while streams with correctly classified samples have a decreasing weight. Results of this stream development are shown in Figure 5.2.

SRQ3: *How can we detect prediction influence extending an existing concept drift detection method?*

One of the initial ideas was to extend an existing concept drift detection method. In the literature review in Chapter 2 we found that concept drift detectors are not good in detecting different types of concept drift. Therefore, it was decided not to pursue this idea further.

SRQ4: *How can we evaluate the prediction influence detector?*

The evaluation of the prediction influence detector is described in Section 3.3. For the evaluation, it was decided to compare multiple settings with the same parameters. This led to six different experiments: No influence without intrinsic drift, no influence with intrinsic drift, self-defeating influence without intrinsic drift, self-defeating influence with intrinsic drift, self-fulfilling influence without intrinsic drift, and self-fulfilling with intrinsic drift. In future research, different parameter settings can be compared with each other.

SRQ5: *What is the performance of the prediction influence detector on real-world data?*

Due to a lack of true labels, this sub-research question cannot be adequately answered. In Section 5.2, the density trajectories of real data features are shown and described. The last step, performing a statistical test between samples of correctly and incorrectly classified samples within a class, could not be executed. Therefore, we cannot measure the performance of the prediction influence detector.

Chapter 8

Future Work

In this thesis, a first approach towards detecting prediction influence is developed and tested. Unfortunately, the results do no show a distinction between experiments with and without prediction influence. The concept of prediction influence is becoming more known, and this thesis is a first step in finding an approach that detects this specific type of drift. The limitations in this thesis were mainly due to limited time, as well as not having access to previous research that tried to detect prediction influence. This gives a lot of room for potential future work. Based on preliminary explorations of further directions, in the following list, some the suggestions are described:

1. **Optimal parameters**: The current approach has many parameters, which are not optimally researched. Prediction influence is believed to be time-sensitive, thus searching for the optimal settings for parameters such as the temporal bandwidth and track length is recommended.
2. **Random classification**: In the current thesis, a Gaussian Naive Bayes classifier is used for the synthetic data experiment. It adapts automatically to newer instances. An interesting idea is to use a random classifier that flips its classification every n months. By using this approach, it becomes clear how long it takes before the prediction influence is visible in the density trajectories.
3. **Classify Lending Club Data**: A specific suggestion for future work regarding the real-world data set is to classify the Lending Club Data. Data from earlier years can be used as a training set, data from newer years as a test set. The labels given by the data set are the predicted labels, the labels generated by the classifier as the true labels. It is impossible to know whether rejected instances are correctly rejected, thus using a classifier allows us to apply the detection method on this data set.
4. **A/B testing**: A/B testing can be used to verify whether experiment set-ups with and without prediction influence are significantly different. In this thesis, the different experiments are shallowly compared with each other. A/B testing gives a scientific, statistical result whether the different experiments give different outcomes.
5. **Focus on decision boundary**: In the current approach, the whole feature space is used to detect a shift in the probability density. However,

most shifts within a class will occur near the feature space. By focusing on the feature space near the decision boundary, the detected shift can be detected easier.

6. **Traditional concept drift detector**: One of the sub-research questions in this thesis was about extending a concept drift detection approach. An interesting idea for future work could be to compare the output of normal concept drift detectors to a prediction influence detector. If artificial temporal concept drifts are introduced in a stream, the concept drift detector should detect these drifts, while the prediction influence detector should not.

Chapter 9

Acknowledgements

I would like to thank my first supervisor, prof. habil. Georg Krempl, for the weekly meetings and the enthusiasm about the thesis topic. I have learnt so much about doing fundamental research, and our meetings kept my motivation high. Our brainstorming sessions were incredibly interesting, thank you for these sessions!

In addition, I want to thank prof. dr. A.P.J.M. Siebes, who has taken the role as second supervisor. I appreciate that you were willing to read and examine this thesis.

Of course, I would also like to thank my Avanade supervisor Iman Jabor for our weekly meetings. I could give my thoughts on the process and you were able to remind me of my planning. You gave me some structure in this whole work-from-home period.

While most of my thesis is written at home because of the Covid pandemic, I could sometimes escape my room to work at my friends' places. Thank you, Bart-Peter, Renee, and Tessa! You made my thesis period a bit more fun.

Lastly, I would like to thank all my friends and family that have listened to me when I needed someone to talk to. I value your support!

Appendix A

Pseudo code

Algorithm A.1 Generator Class: NextSample

Require: *batch_size* (default to 1)
  current_sample_x ← *emptylist*
  current_sample_y ← *emptylist*
  for i in range(*batch_size*) **do**
    $weights \leftarrow normalize(weights)$
    $current\_stream \leftarrow random(streams, weights)$
    $X, y \leftarrow current\_stream.next\_sample()$
    $current\_sample\_x \leftarrow X$
    $current\_sample\_y \leftarrow y$
  end for

  return *current_sample_x*, *current_sample_y*

Algorithm A.2 Generator Class: ReceiveFeedback

Require: True label *true*
Require: Predicted label *pred*
Require: Features $features$
Require: Cache of unprocessed instances *cache*
  if *true* exists **then**
    if *cache* is empty OR *pred* and $features$ are in *cache*[0] **then**
      $receive\_feedback\_update()$
      if *cache* is not empty **then**
        remove *cache*[0]
        while *cache*[0] has *true* **do**
          $receive\_feedback\_update()$
          remove *cache*[0]
        end while
      end if
    else
      $cache.append([true, pred, features])$
    end if
  else
    $cache.append([pred, features])$
  end if

Algorithm A.3 Generator Class: ReceiveFeedbackUpdate

Require: True label *true*
Require: Predicted label *pred*
Require: List of stream weights *weight*
Require: *weight_correct*
Require: *weight_incorrect*
Require: Number of current stream *current_stream*
  if $true == pred$ **then**
    $weight[current\_stream] \leftarrow weight[current\_stream] * weight\_correct$
  else
    $weight[current\_stream] \leftarrow weight[current\_stream] * weight\_incorrect$
  end if
  return *weight*

Algorithm A.4 Tracker Class: getDensity

Require: $t, x, y, \hat{y}$
Require: temporal bandwidth in months b
Require: track length in months l
Require: cache
  $cache \leftarrow cache.append(x, y, t, \hat{y})$
  $cache \leftarrow cache[y = y, \hat{y} = \hat{y}]$ {# select instances with similar class label and predicted label}
  if $len(cache.month.unique) \geq b + l$ **then**
    for $month \in cache$ **do**
      $KDE \leftarrow kde.fit(cache[month])$
      for $sample \in cache[month == t]$ **do**
        $density \leftarrow KDE.score(sample)$
        $table[y][\hat{y}].update(value \leftarrow density, column \leftarrow month - t, row \leftarrow sample.index)$ {# if t = 2012-03 and month = 2012-01, then position in table = -2}
      end for
      $current\_KDE \leftarrow kde.fit(cache[month == t])$
      for $sample \in month$ **do**
        $density \leftarrow current\_KDE.score(sample)$
        $table[y][\hat{y}].update(value \leftarrow density, column \leftarrow t - month, row \leftarrow sample.index)$ {# if t = 2012-03 and month = 2012-01, then position in table = 2}
      end for
    end for
  end if

Algorithm A.5 Tracker Class: calculateScore

Require: temporal bandwidth in months b
Require: track length in months l
Require: table for one confusion matrix cell
  for $month \in table.months$ **do**
    for $index, row \in table[month == month]$ **do**
      $ln \leftarrow linear\_model.LinearRegression()$
      $ln.fit(row.loc[-l, 0])$
      $table\_score \leftarrow ln.score(row.loc[0, l])$
      $results\_table.save\_score(table\_score[index] - row[index], month)$
    end for
  end for

  return *results_table*

Algorithm A.6 Tracker Class: ranksums

Require: temporal bandwidth in months b
Require: track length in months l
Require: table correctly predicted (TN or TP) *correct* and table incorrectly predicted (FN or FP) *incorrect*
  for $month \in table.months$ **do**
    for $i \in l$ **do**
      $test \leftarrow ranksums(correct[month][i], incorrect[month][i])$
      $pvalues.append(test.pvalue)$
    end for
    $pvalue\_table[month] \leftarrow pvalues$
    $pvalues \leftarrow []$
  end for

  return *pvalue_table*

Appendix B

Results synthetic data experiments

FIGURE B.1: Density trajectories, no influence without intrinsic drift

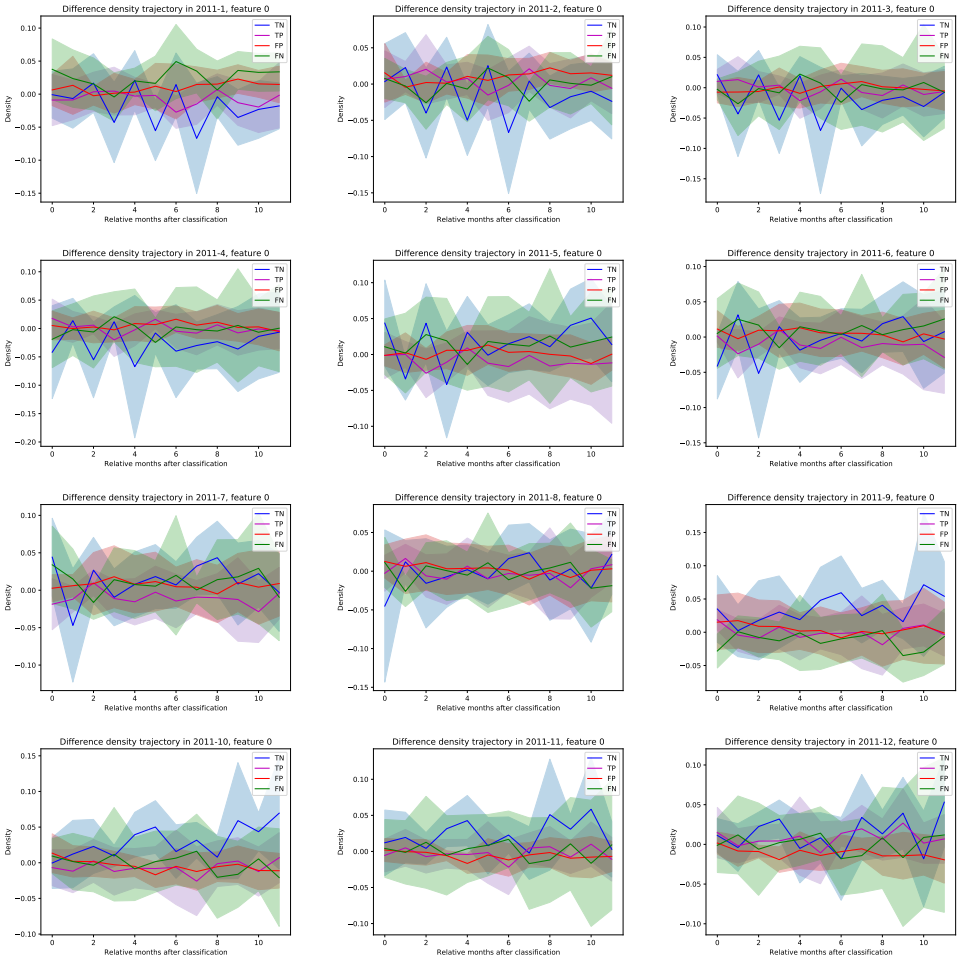


FIGURE B.2: Density difference, no influence without intrinsic drift

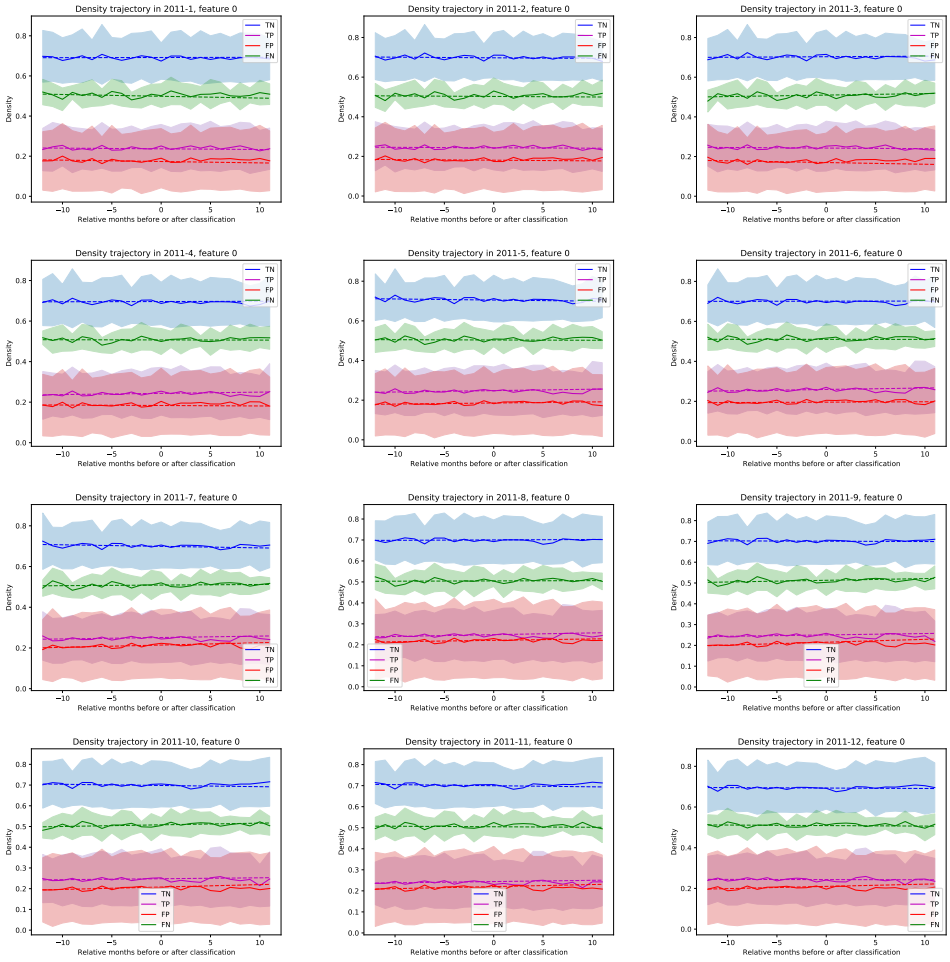


FIGURE B.3: Density trajectories, no influence with intrinsic drift

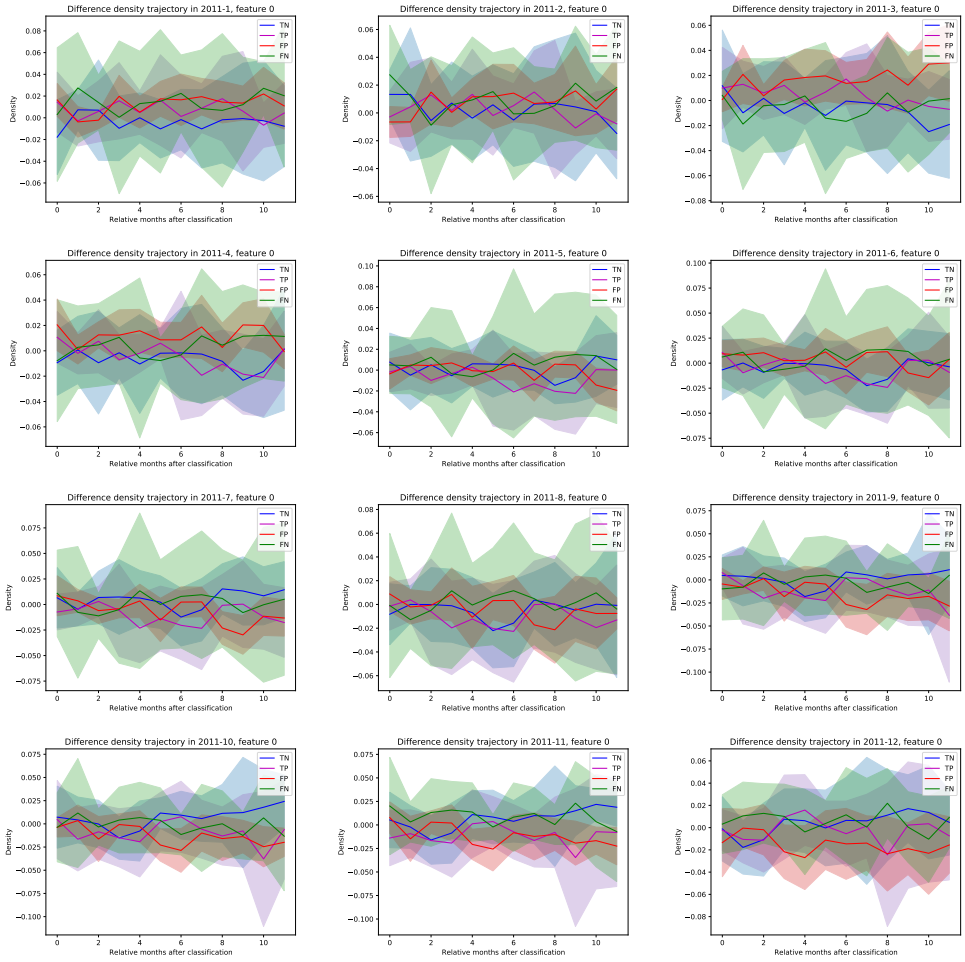


FIGURE B.4: Density difference, no influence, intrinsic drift

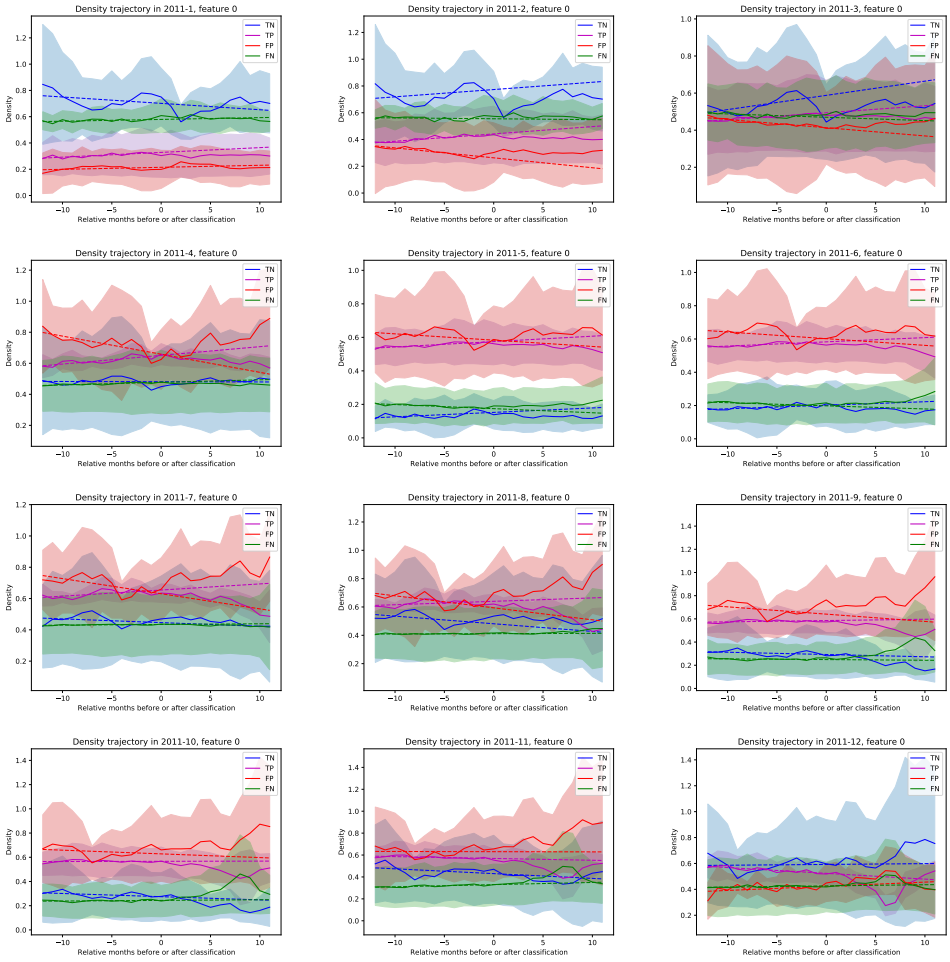


FIGURE B.5: Density trajectories self-defeating influence without intrinsic drift

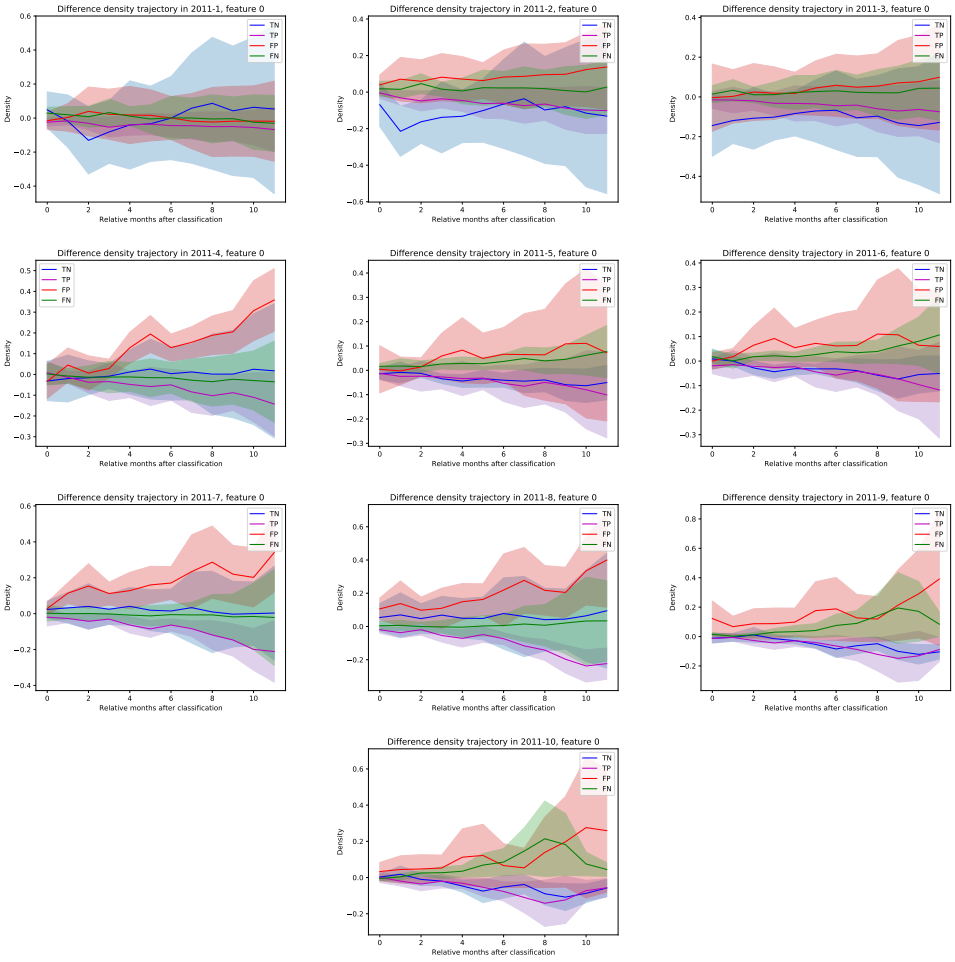


FIGURE B.6: Density difference self-defeating influence without intrinsic drift

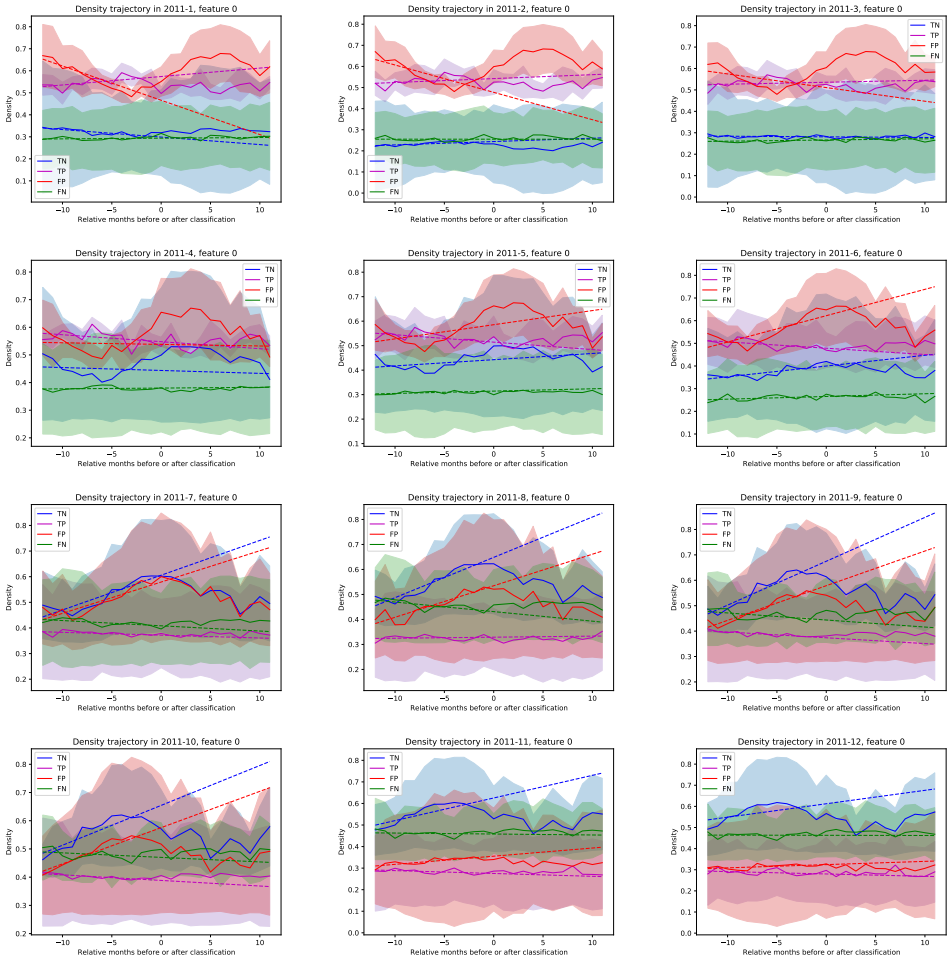


FIGURE B.7: Density trajectories self-defeating influence with intrinsic drift

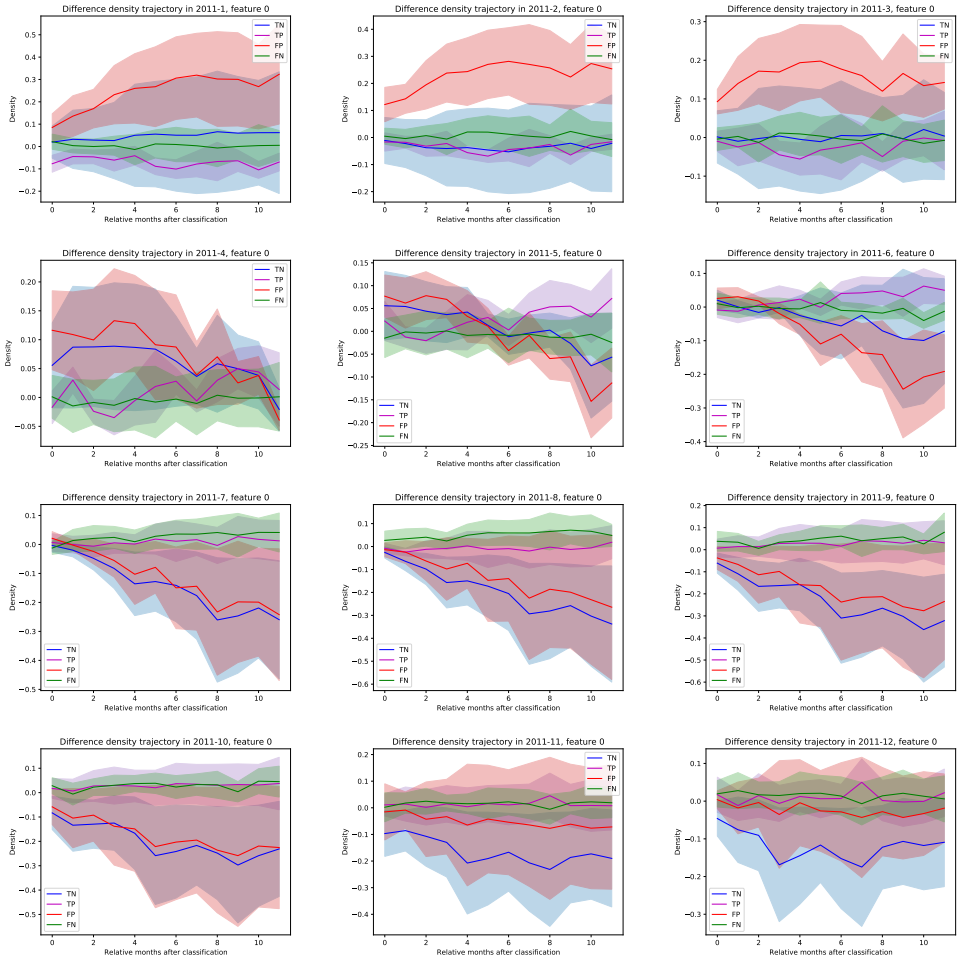


FIGURE B.8: Density difference, self-defeating influence with intrinsic drift

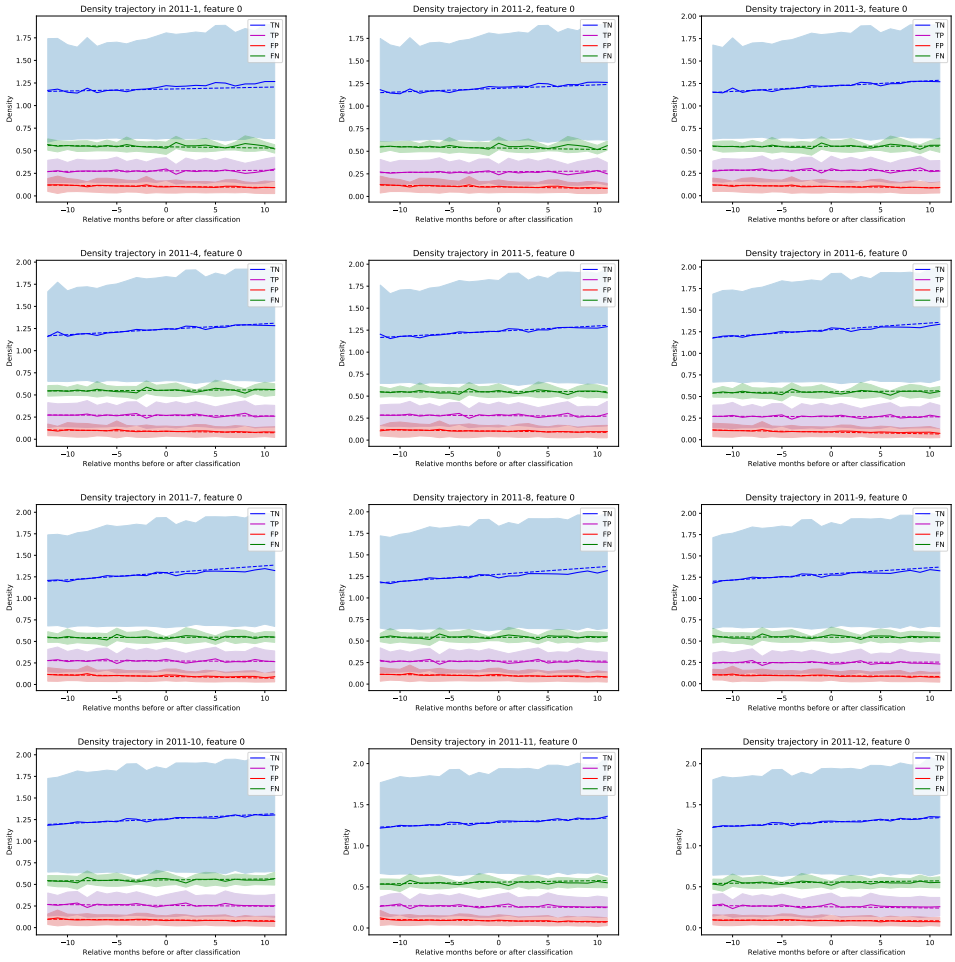


FIGURE B.9: Density trajectories self-fulfilling influence without intrinsic drift

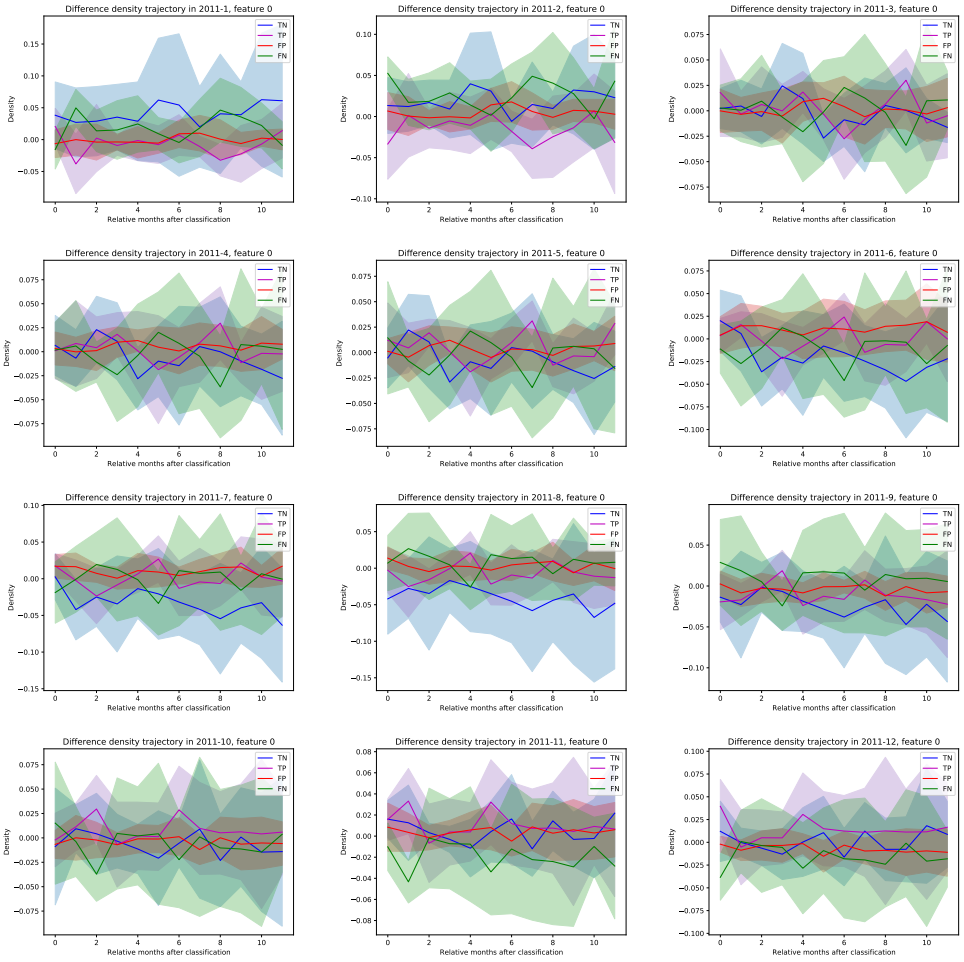


FIGURE B.10: Density difference, self-fulfilling influence without intrinsic drift

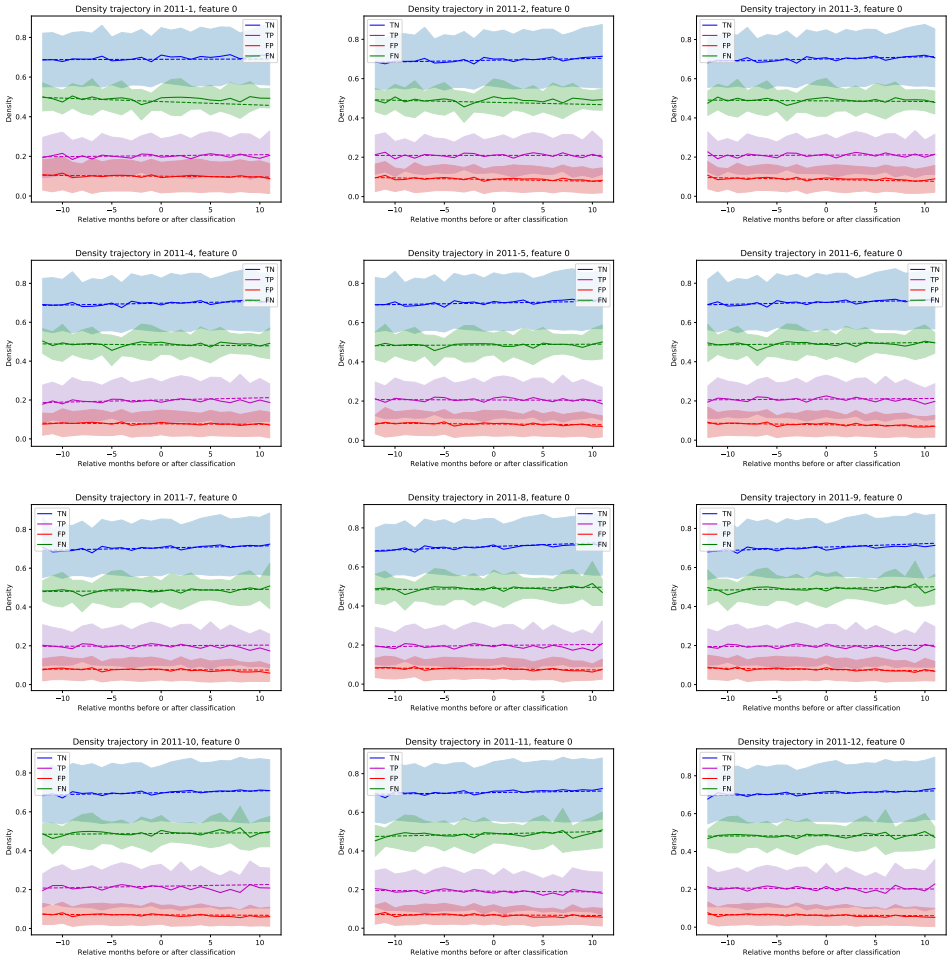


FIGURE B.11: Density trajectories self-fulfilling with intrinsic drift

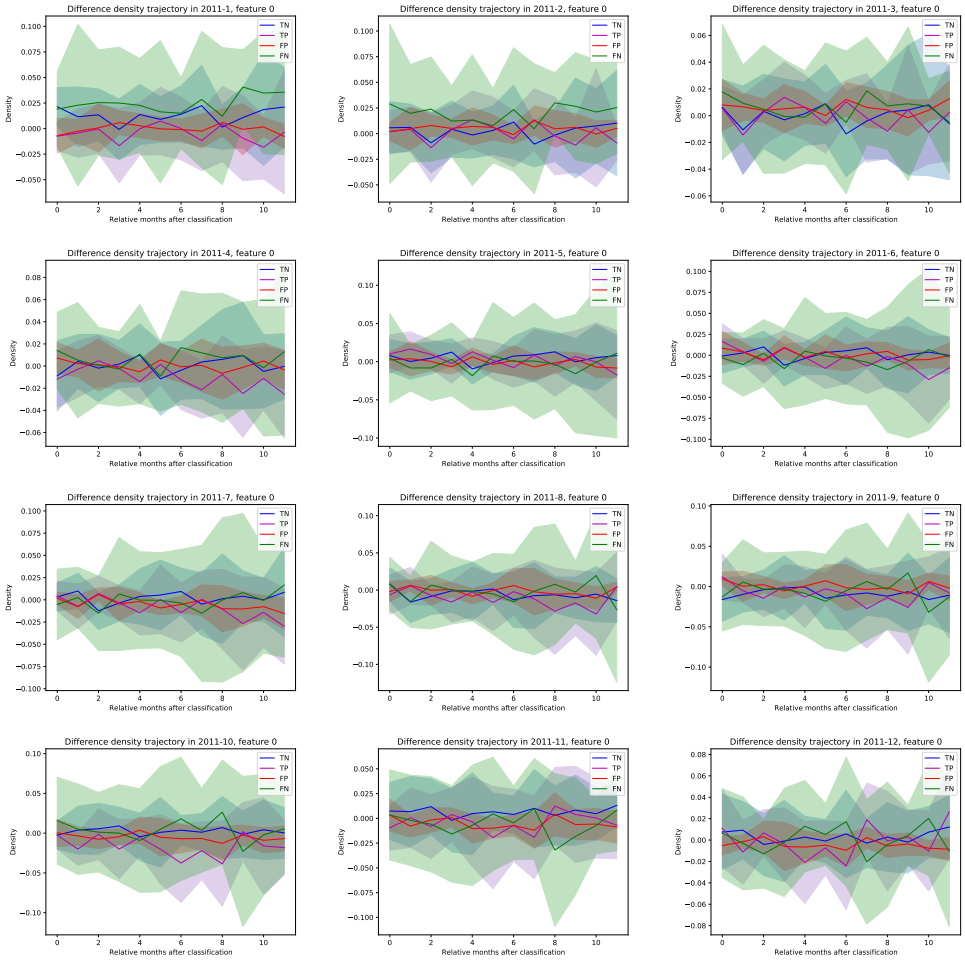
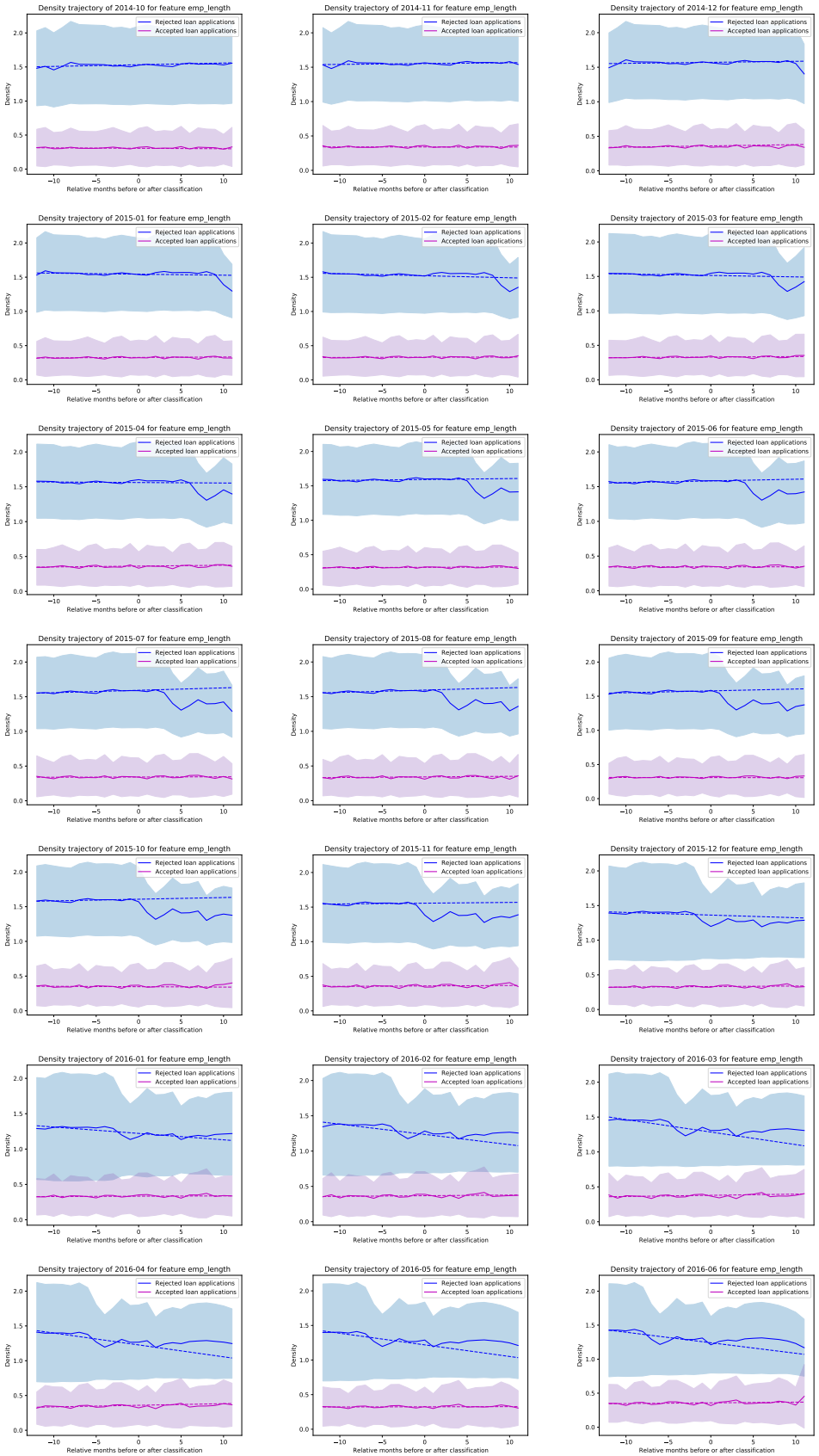


FIGURE B.12: Density difference, self-fulfilling with intrinsic drift

Appendix C

Results Lending Club data


Density trajectory of 2013-01 for feature emp_length
Density trajectory of 2013-02 for feature emp_length
Density trajectory of 2013-03 for feature emp_length
Density trajectory of 2013-04 for feature emp_length
Density trajectory of 2013-05 for feature emp_length
Density trajectory of 2013-06 for feature emp_length
Density trajectory of 2013-07 for feature emp_length
Density trajectory of 2013-08 for feature emp_length
Density trajectory of 2013-09 for feature emp_length
Density trajectory of 2013-10 for feature emp_length
Density trajectory of 2013-11 for feature emp_length
Density trajectory of 2013-12 for feature emp_length
Density trajectory of 2014-01 for feature emp_length
Density trajectory of 2014-02 for feature emp_length
Density trajectory of 2014-03 for feature emp_length
Density trajectory of 2014-04 for feature emp_length
Density trajectory of 2014-05 for feature emp_length
Density trajectory of 2014-06 for feature emp_length
Density trajectory of 2014-07 for feature emp_length
Density trajectory of 2014-08 for feature emp_length
Density trajectory of 2014-09 for feature emp_length
Rejected loan applications
Accepted loan applications
Density
Relative months before or after classification


Density trajectory of 2014-10 for feature emp_length
Density trajectory of 2014-11 for feature emp_length
Density trajectory of 2014-12 for feature emp_length
Density trajectory of 2015-01 for feature emp_length
Density trajectory of 2015-02 for feature emp_length
Density trajectory of 2015-03 for feature emp_length
Density trajectory of 2015-04 for feature emp_length
Density trajectory of 2015-05 for feature emp_length
Density trajectory of 2015-06 for feature emp_length
Density trajectory of 2015-07 for feature emp_length
Density trajectory of 2015-08 for feature emp_length
Density trajectory of 2015-09 for feature emp_length
Density trajectory of 2015-10 for feature emp_length
Density trajectory of 2015-11 for feature emp_length
Density trajectory of 2015-12 for feature emp_length
Density trajectory of 2016-01 for feature emp_length
Density trajectory of 2016-02 for feature emp_length
Density trajectory of 2016-03 for feature emp_length
Density trajectory of 2016-04 for feature emp_length
Density trajectory of 2016-05 for feature emp_length
Density trajectory of 2016-06 for feature emp_length
Rejected loan applications
Accepted loan applications
Density
Relative months before or after classification

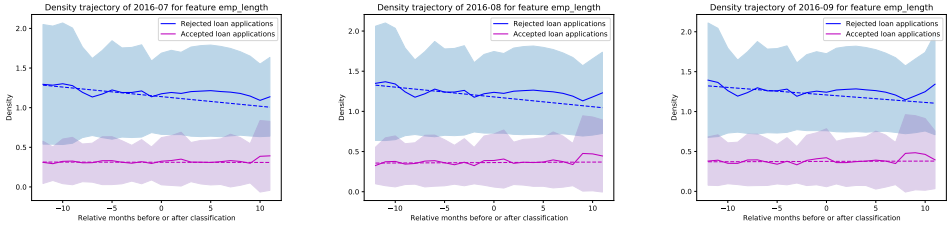
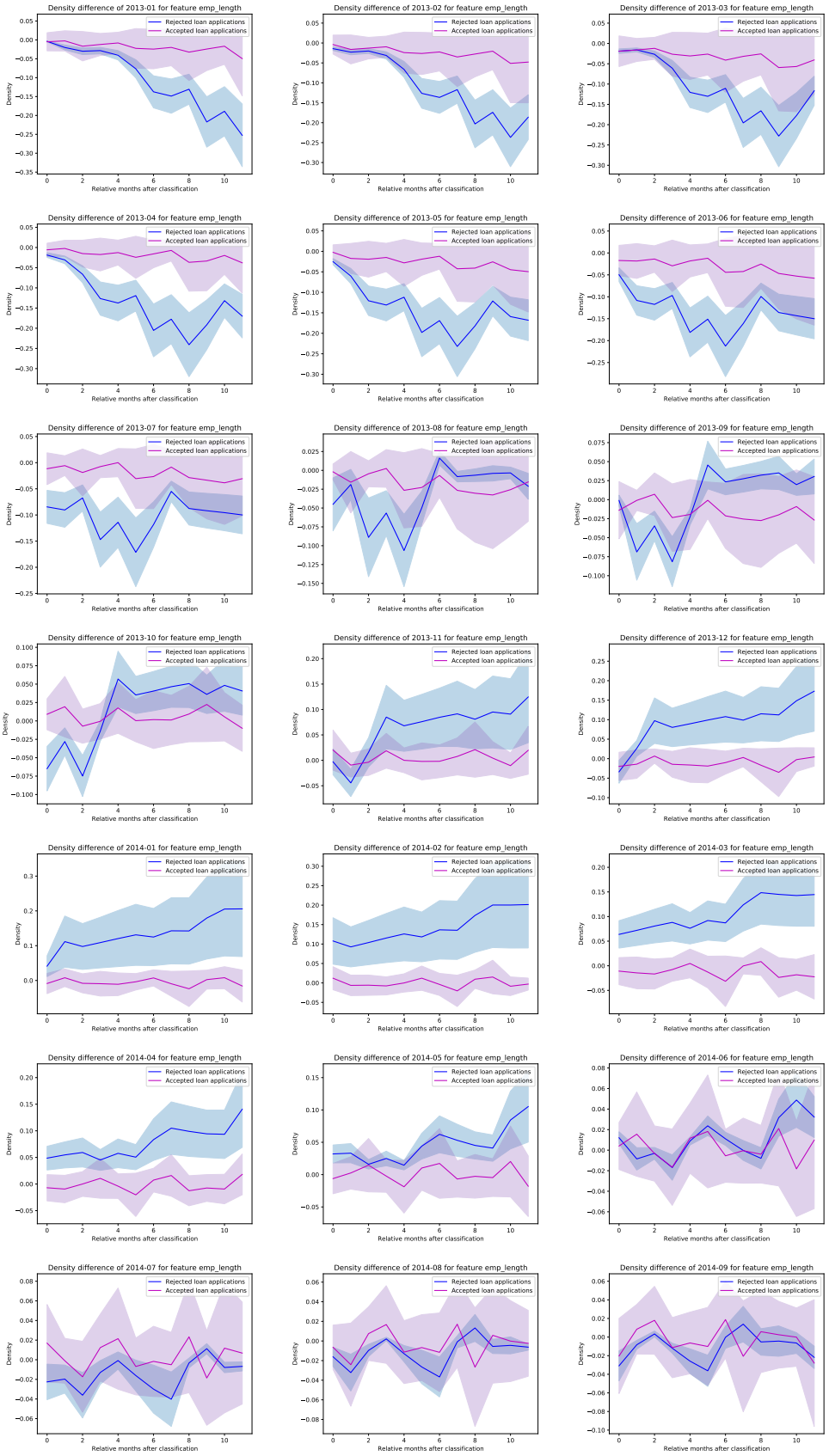
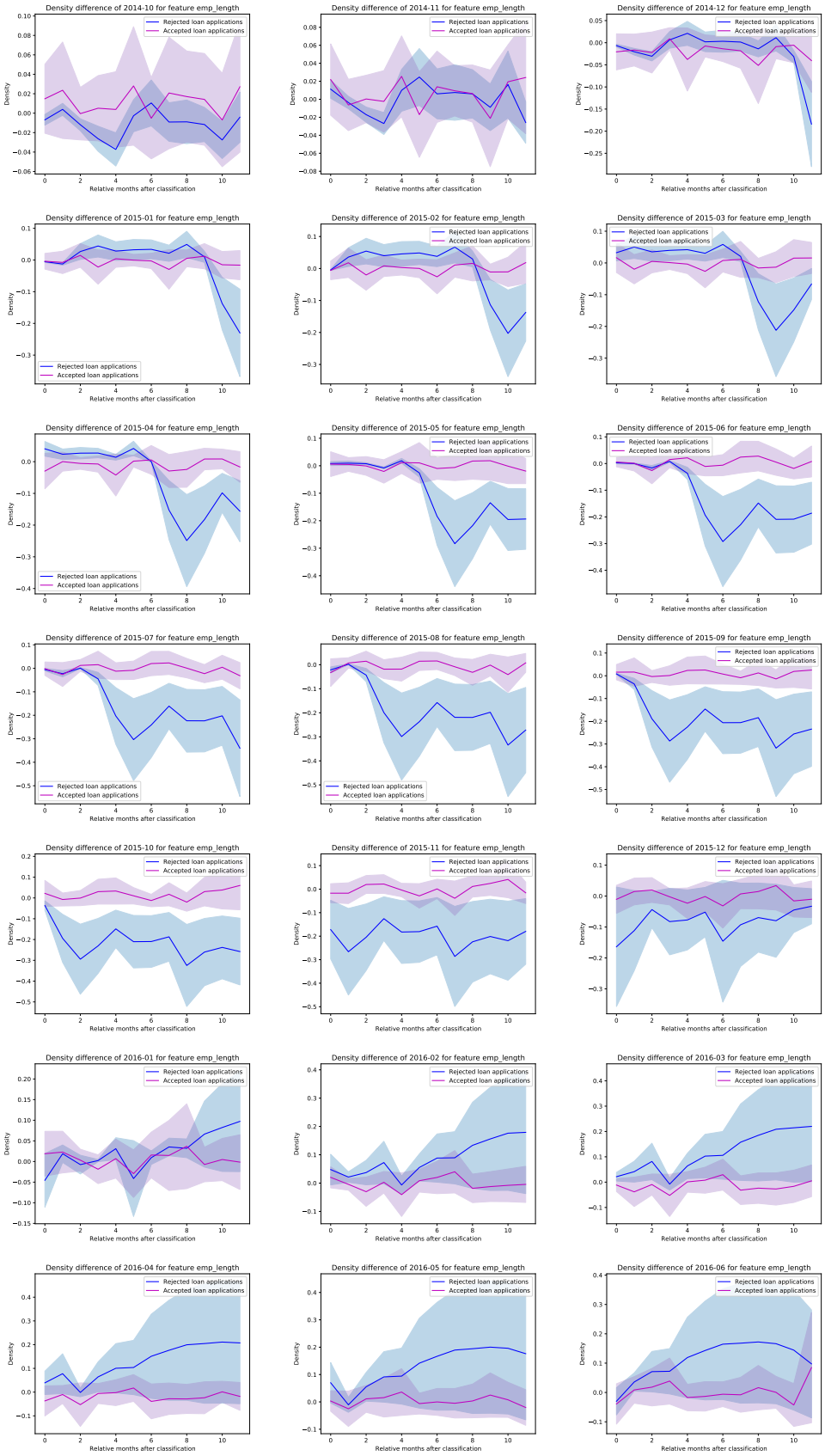


FIGURE C.1: Density trajectory employment length


Density difference of 2013-01 for feature emp_length
Density difference of 2013-02 for feature emp_length
Density difference of 2013-03 for feature emp_length
Density difference of 2013-04 for feature emp_length
Density difference of 2013-05 for feature emp_length
Density difference of 2013-06 for feature emp_length
Density difference of 2013-07 for feature emp_length
Density difference of 2013-08 for feature emp_length
Density difference of 2013-09 for feature emp_length
Density difference of 2013-10 for feature emp_length
Density difference of 2013-11 for feature emp_length
Density difference of 2013-12 for feature emp_length
Density difference of 2014-01 for feature emp_length
Density difference of 2014-02 for feature emp_length
Density difference of 2014-03 for feature emp_length
Density difference of 2014-04 for feature emp_length
Density difference of 2014-05 for feature emp_length
Density difference of 2014-06 for feature emp_length
Density difference of 2014-07 for feature emp_length
Density difference of 2014-08 for feature emp_length
Density difference of 2014-09 for feature emp_length
Rejected loan applications
Accepted loan applications
Density
Relative months after classification


Density difference of 2014-10 for feature emp_length
Density difference of 2014-11 for feature emp_length
Density difference of 2014-12 for feature emp_length
Density difference of 2015-01 for feature emp_length
Density difference of 2015-02 for feature emp_length
Density difference of 2015-03 for feature emp_length
Density difference of 2015-04 for feature emp_length
Density difference of 2015-05 for feature emp_length
Density difference of 2015-06 for feature emp_length
Density difference of 2015-07 for feature emp_length
Density difference of 2015-08 for feature emp_length
Density difference of 2015-09 for feature emp_length
Density difference of 2015-10 for feature emp_length
Density difference of 2015-11 for feature emp_length
Density difference of 2015-12 for feature emp_length
Density difference of 2016-01 for feature emp_length
Density difference of 2016-02 for feature emp_length
Density difference of 2016-03 for feature emp_length
Density difference of 2016-04 for feature emp_length
Density difference of 2016-05 for feature emp_length
Density difference of 2016-06 for feature emp_length
Rejected loan applications
Accepted loan applications
Density
Relative months after classification

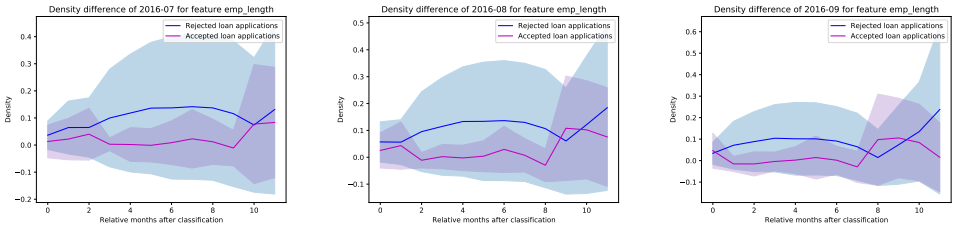
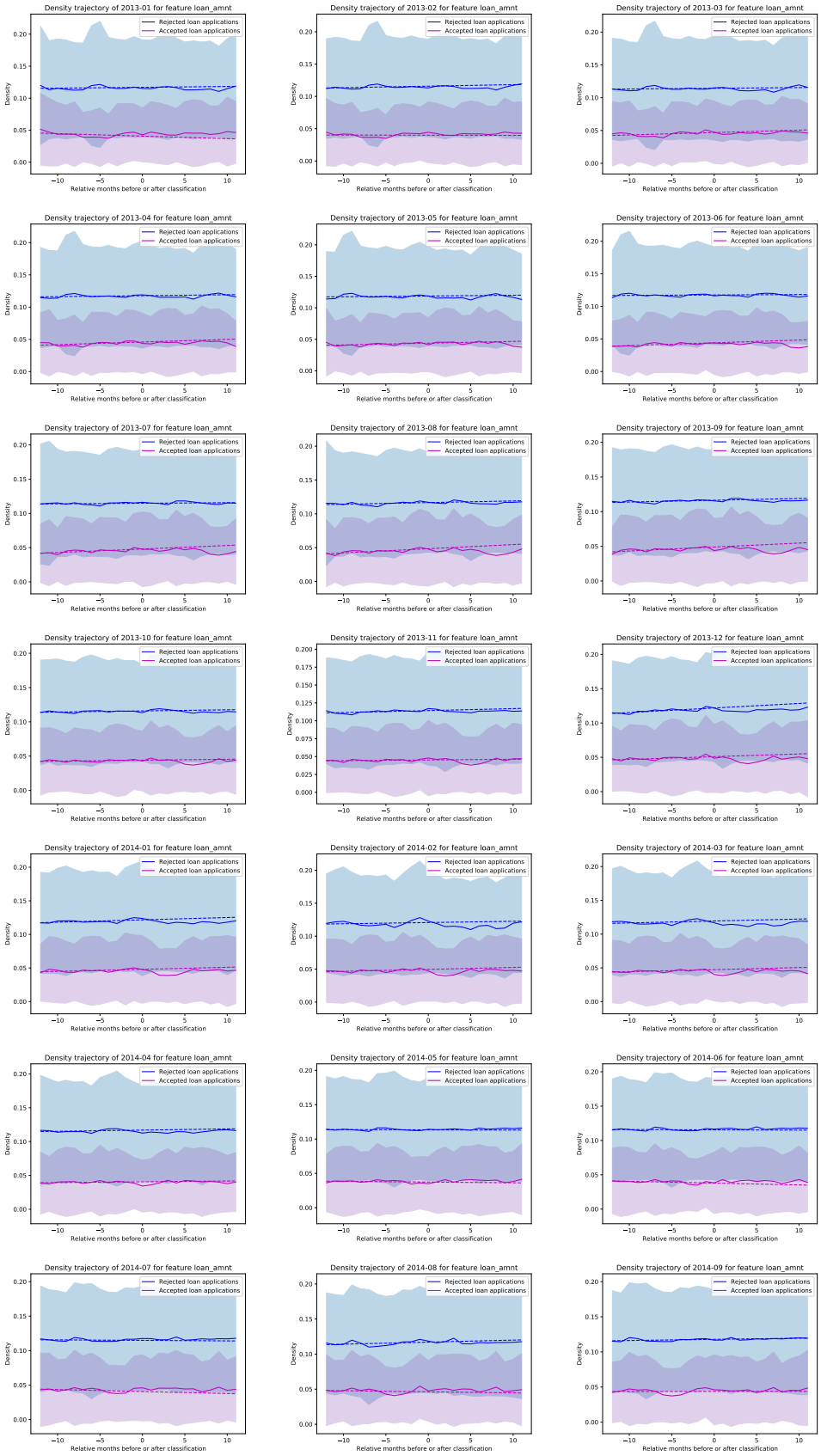
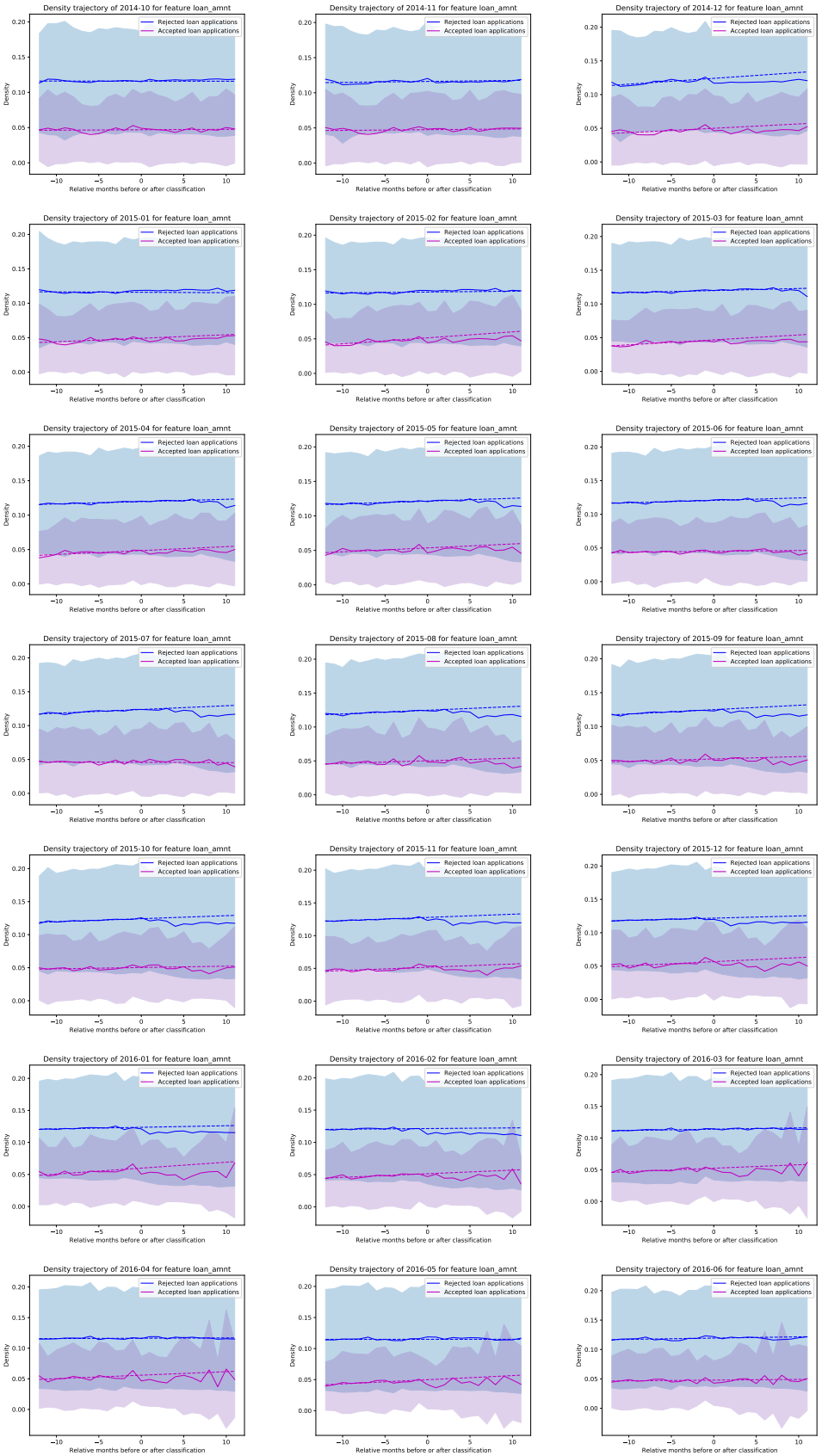


FIGURE C.2: Density difference employment length


Density trajectory of 2013-01 for feature loan_amnt
Density trajectory of 2013-02 for feature loan_amnt
Density trajectory of 2013-03 for feature loan_amnt
Density trajectory of 2013-04 for feature loan_amnt
Density trajectory of 2013-05 for feature loan_amnt
Density trajectory of 2013-06 for feature loan_amnt
Density trajectory of 2013-07 for feature loan_amnt
Density trajectory of 2013-08 for feature loan_amnt
Density trajectory of 2013-09 for feature loan_amnt
Density trajectory of 2013-10 for feature loan_amnt
Density trajectory of 2013-11 for feature loan_amnt
Density trajectory of 2013-12 for feature loan_amnt
Density trajectory of 2014-01 for feature loan_amnt
Density trajectory of 2014-02 for feature loan_amnt
Density trajectory of 2014-03 for feature loan_amnt
Density trajectory of 2014-04 for feature loan_amnt
Density trajectory of 2014-05 for feature loan_amnt
Density trajectory of 2014-06 for feature loan_amnt
Density trajectory of 2014-07 for feature loan_amnt
Density trajectory of 2014-08 for feature loan_amnt
Density trajectory of 2014-09 for feature loan_amnt
Rejected loan applications
Accepted loan applications
Density
Relative months before or after classification


Density trajectory of 2014-10 for feature loan_amnt
Density trajectory of 2014-11 for feature loan_amnt
Density trajectory of 2014-12 for feature loan_amnt
Density trajectory of 2015-01 for feature loan_amnt
Density trajectory of 2015-02 for feature loan_amnt
Density trajectory of 2015-03 for feature loan_amnt
Density trajectory of 2015-04 for feature loan_amnt
Density trajectory of 2015-05 for feature loan_amnt
Density trajectory of 2015-06 for feature loan_amnt
Density trajectory of 2015-07 for feature loan_amnt
Density trajectory of 2015-08 for feature loan_amnt
Density trajectory of 2015-09 for feature loan_amnt
Density trajectory of 2015-10 for feature loan_amnt
Density trajectory of 2015-11 for feature loan_amnt
Density trajectory of 2015-12 for feature loan_amnt
Density trajectory of 2016-01 for feature loan_amnt
Density trajectory of 2016-02 for feature loan_amnt
Density trajectory of 2016-03 for feature loan_amnt
Density trajectory of 2016-04 for feature loan_amnt
Density trajectory of 2016-05 for feature loan_amnt
Density trajectory of 2016-06 for feature loan_amnt
Rejected loan applications
Accepted loan applications
Density
Relative months before or after classification

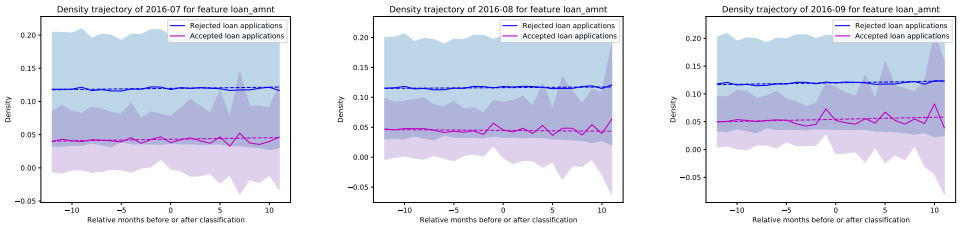
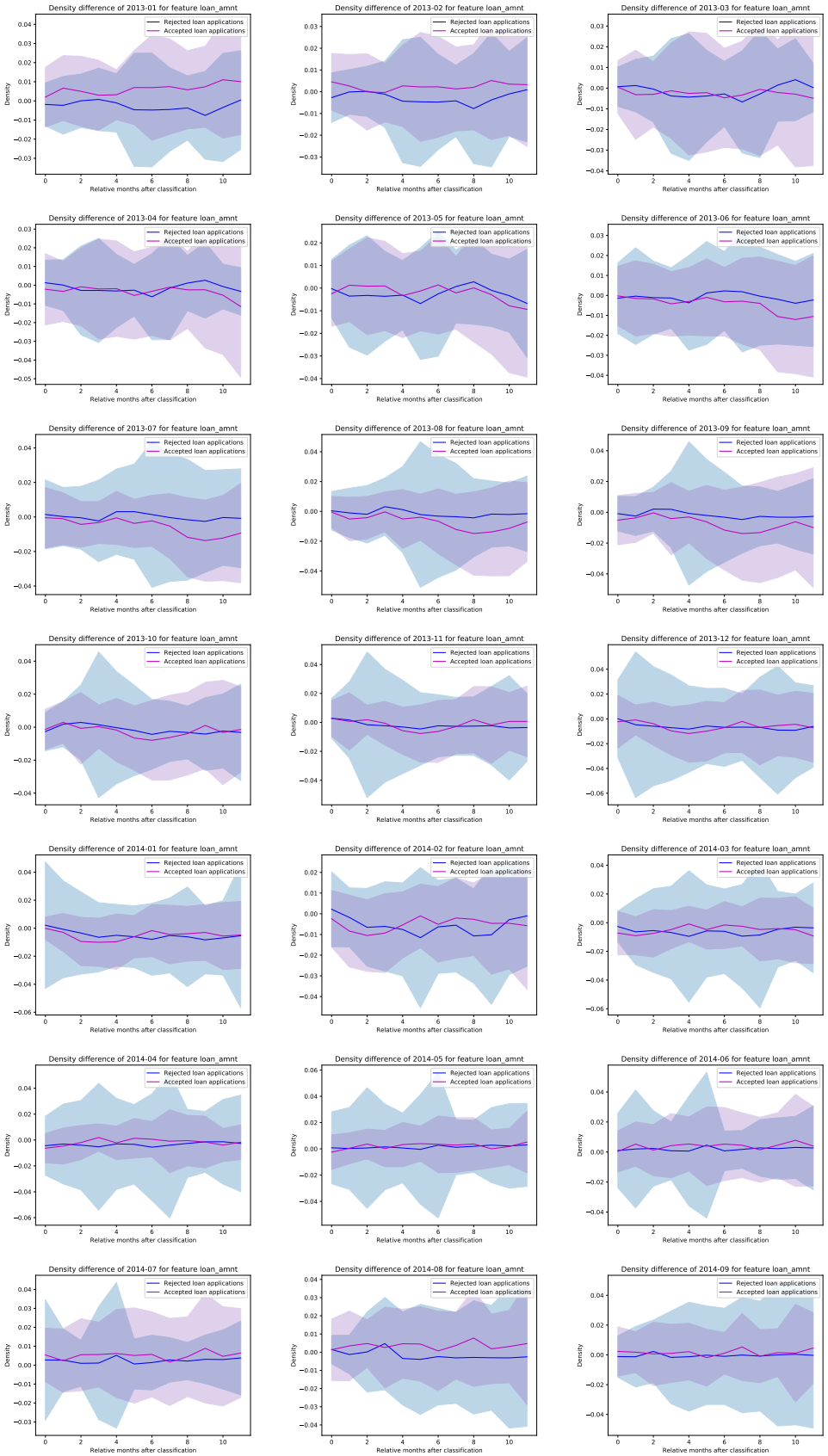
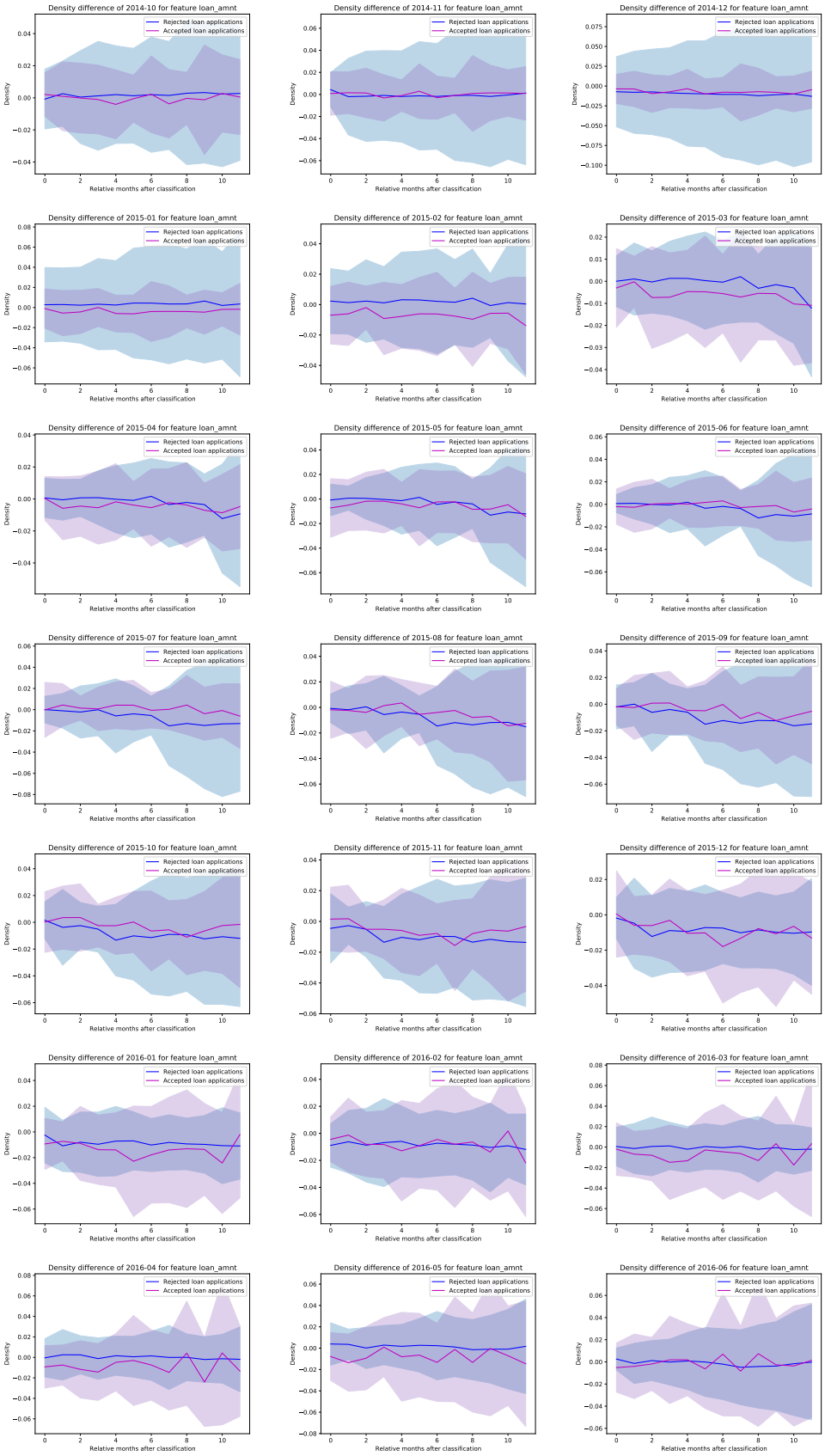


FIGURE C.3: Density trajectory loan amount


Density difference of 2013-01 for feature loan_amnt
Density difference of 2013-02 for feature loan_amnt
Density difference of 2013-03 for feature loan_amnt
Density difference of 2013-04 for feature loan_amnt
Density difference of 2013-05 for feature loan_amnt
Density difference of 2013-06 for feature loan_amnt
Density difference of 2013-07 for feature loan_amnt
Density difference of 2013-08 for feature loan_amnt
Density difference of 2013-09 for feature loan_amnt
Density difference of 2013-10 for feature loan_amnt
Density difference of 2013-11 for feature loan_amnt
Density difference of 2013-12 for feature loan_amnt
Density difference of 2014-01 for feature loan_amnt
Density difference of 2014-02 for feature loan_amnt
Density difference of 2014-03 for feature loan_amnt
Density difference of 2014-04 for feature loan_amnt
Density difference of 2014-05 for feature loan_amnt
Density difference of 2014-06 for feature loan_amnt
Density difference of 2014-07 for feature loan_amnt
Density difference of 2014-08 for feature loan_amnt
Density difference of 2014-09 for feature loan_amnt
Rejected loan applications
Accepted loan applications
Density
Relative months after classification


Density difference of 2014-10 for feature loan_amnt
Density difference of 2014-11 for feature loan_amnt
Density difference of 2014-12 for feature loan_amnt
Density difference of 2015-01 for feature loan_amnt
Density difference of 2015-02 for feature loan_amnt
Density difference of 2015-03 for feature loan_amnt
Density difference of 2015-04 for feature loan_amnt
Density difference of 2015-05 for feature loan_amnt
Density difference of 2015-06 for feature loan_amnt
Density difference of 2015-07 for feature loan_amnt
Density difference of 2015-08 for feature loan_amnt
Density difference of 2015-09 for feature loan_amnt
Density difference of 2015-10 for feature loan_amnt
Density difference of 2015-11 for feature loan_amnt
Density difference of 2015-12 for feature loan_amnt
Density difference of 2016-01 for feature loan_amnt
Density difference of 2016-02 for feature loan_amnt
Density difference of 2016-03 for feature loan_amnt
Density difference of 2016-04 for feature loan_amnt
Density difference of 2016-05 for feature loan_amnt
Density difference of 2016-06 for feature loan_amnt
Rejected loan applications
Accepted loan applications
Density
Relative months after classification

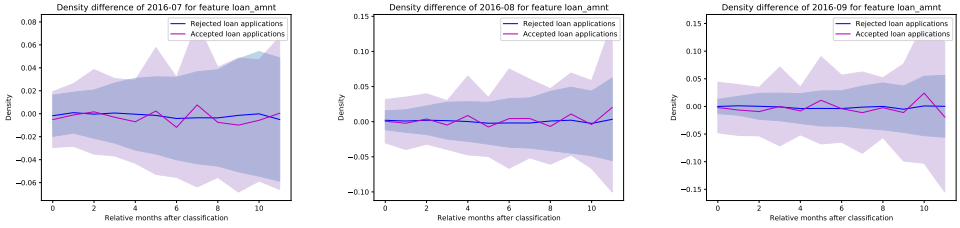
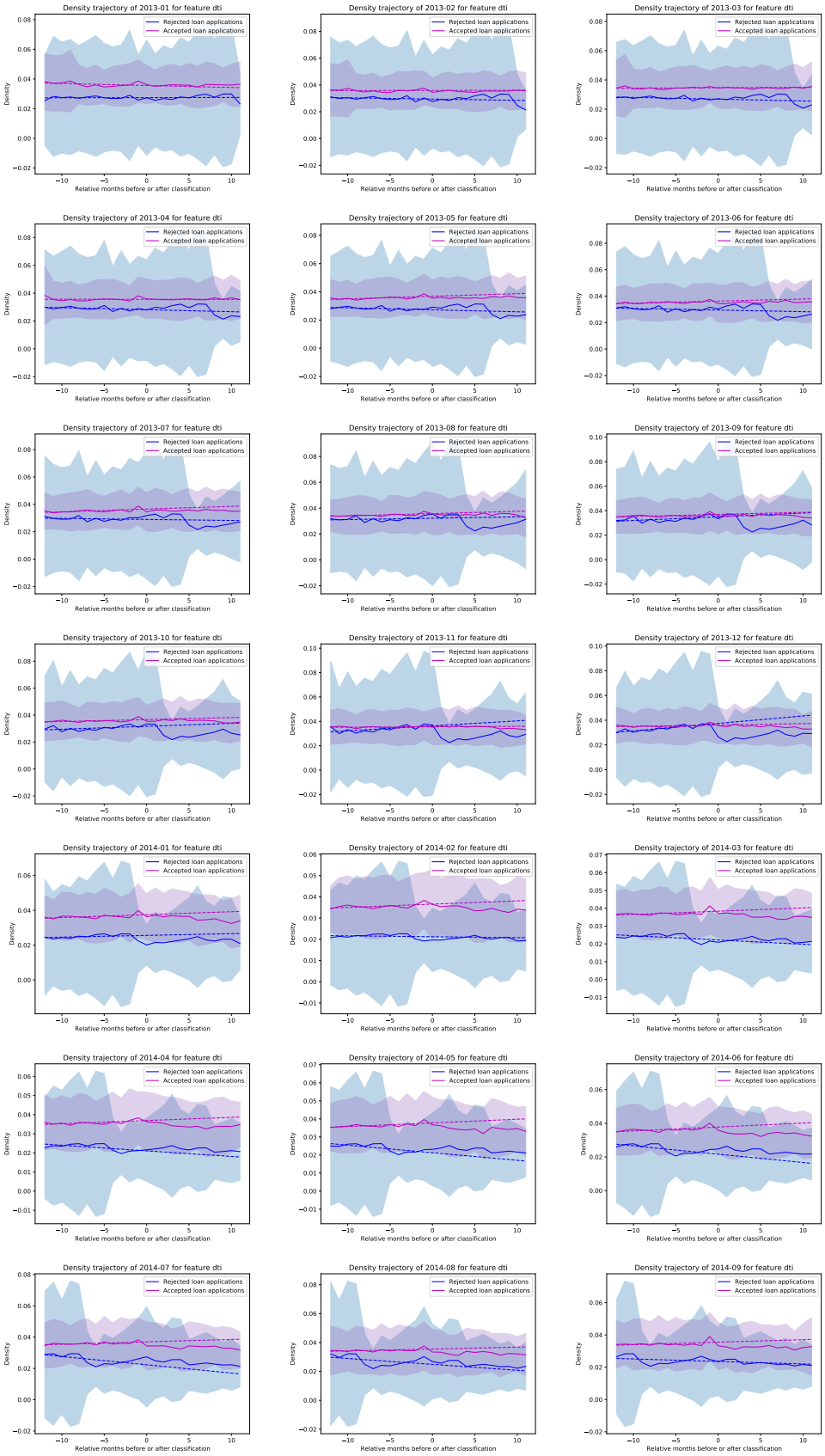
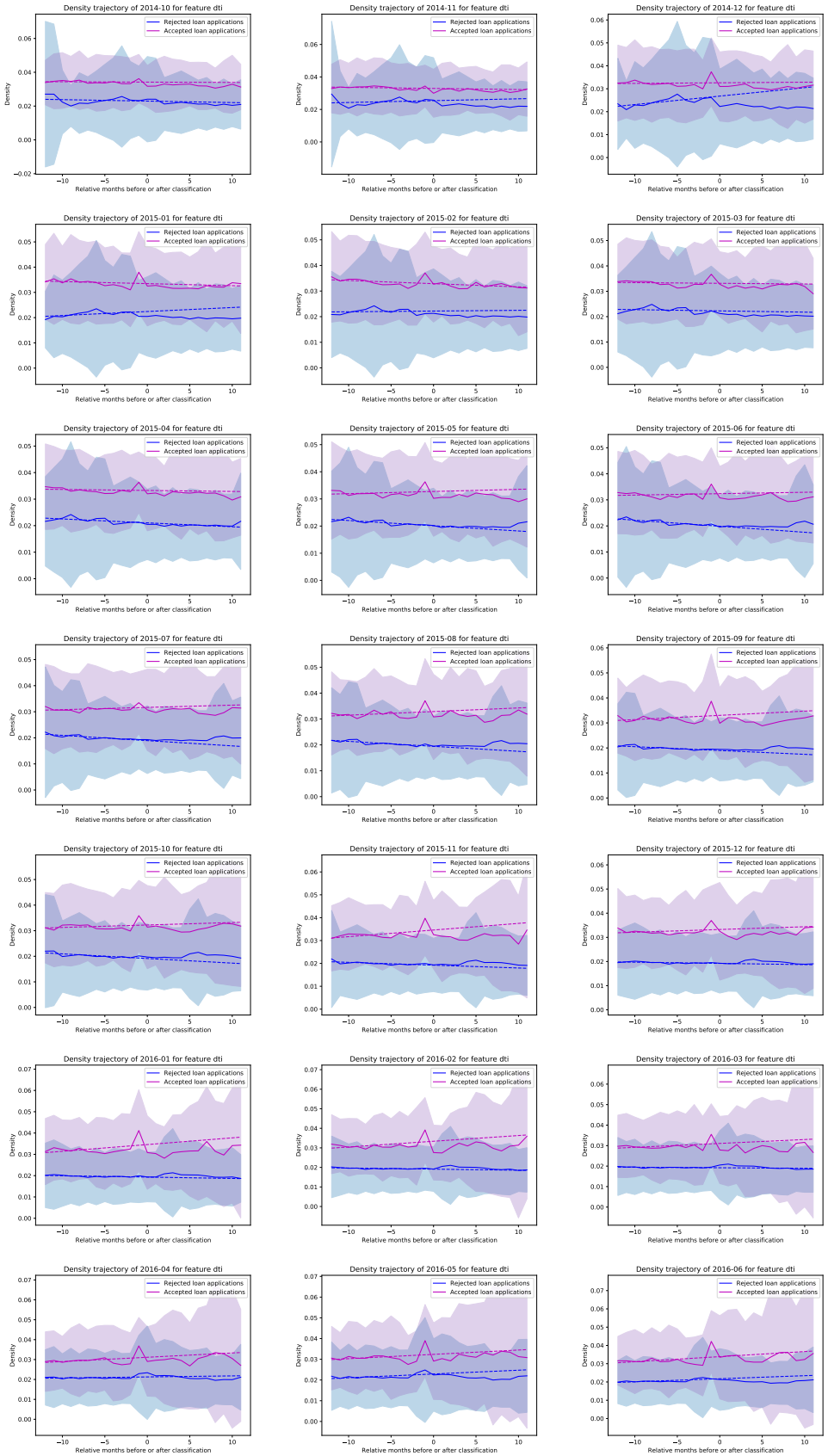


FIGURE C.4: Density difference loan amount


Density trajectory of 2013-01 for feature dti
Density trajectory of 2013-02 for feature dti
Density trajectory of 2013-03 for feature dti
Density trajectory of 2013-04 for feature dti
Density trajectory of 2013-05 for feature dti
Density trajectory of 2013-06 for feature dti
Density trajectory of 2013-07 for feature dti
Density trajectory of 2013-08 for feature dti
Density trajectory of 2013-09 for feature dti
Density trajectory of 2013-10 for feature dti
Density trajectory of 2013-11 for feature dti
Density trajectory of 2013-12 for feature dti
Density trajectory of 2014-01 for feature dti
Density trajectory of 2014-02 for feature dti
Density trajectory of 2014-03 for feature dti
Density trajectory of 2014-04 for feature dti
Density trajectory of 2014-05 for feature dti
Density trajectory of 2014-06 for feature dti
Density trajectory of 2014-07 for feature dti
Density trajectory of 2014-08 for feature dti
Density trajectory of 2014-09 for feature dti
Rejected loan applications
Accepted loan applications
Density
Relative months before or after classification


Density trajectory of 2014-10 for feature dti
Density trajectory of 2014-11 for feature dti
Density trajectory of 2014-12 for feature dti
Density trajectory of 2015-01 for feature dti
Density trajectory of 2015-02 for feature dti
Density trajectory of 2015-03 for feature dti
Density trajectory of 2015-04 for feature dti
Density trajectory of 2015-05 for feature dti
Density trajectory of 2015-06 for feature dti
Density trajectory of 2015-07 for feature dti
Density trajectory of 2015-08 for feature dti
Density trajectory of 2015-09 for feature dti
Density trajectory of 2015-10 for feature dti
Density trajectory of 2015-11 for feature dti
Density trajectory of 2015-12 for feature dti
Density trajectory of 2016-01 for feature dti
Density trajectory of 2016-02 for feature dti
Density trajectory of 2016-03 for feature dti
Density trajectory of 2016-04 for feature dti
Density trajectory of 2016-05 for feature dti
Density trajectory of 2016-06 for feature dti
Rejected loan applications
Accepted loan applications
Density
Relative months before or after classification

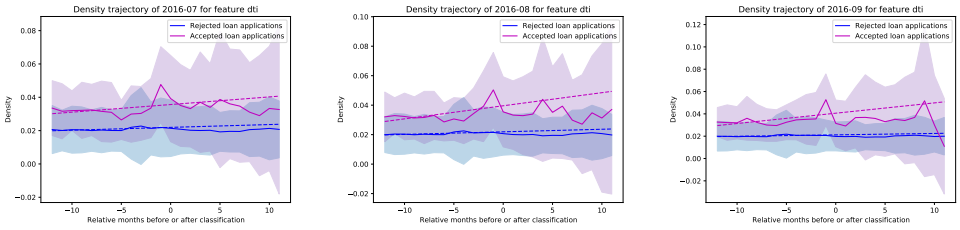
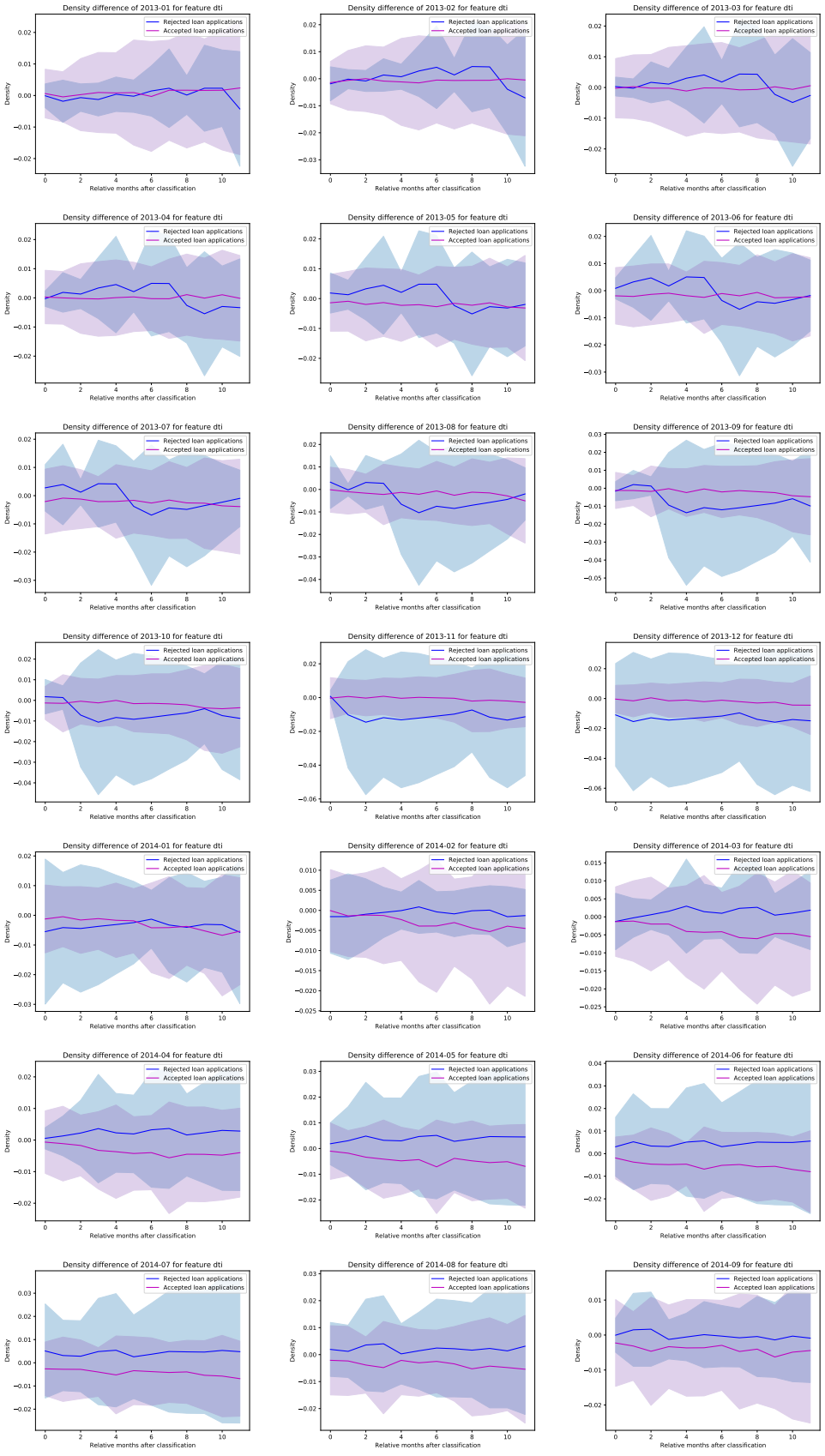
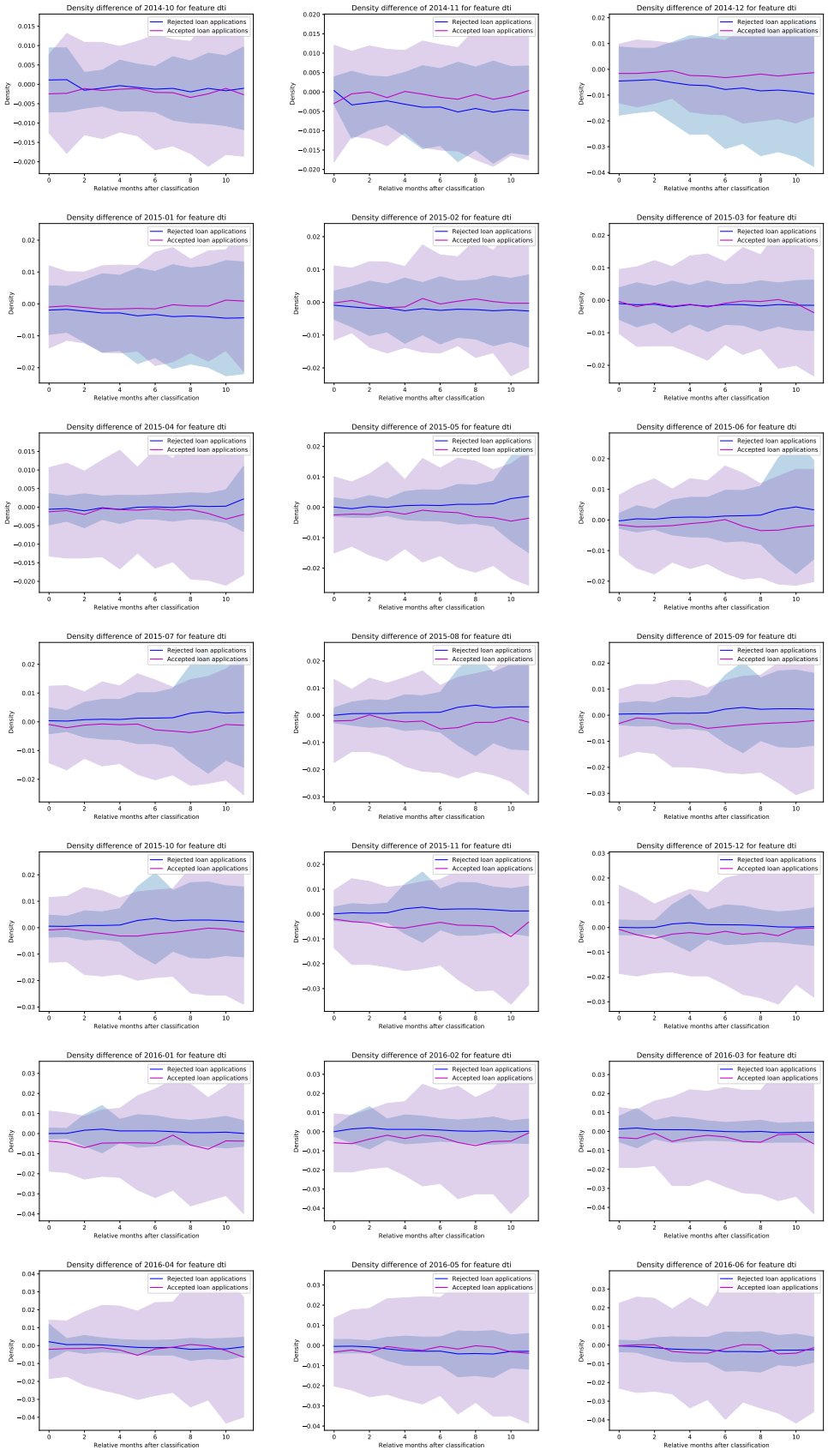


FIGURE C.5: Density trajectory DTI


Density difference of 2013-01 for feature dti
Density difference of 2013-02 for feature dti
Density difference of 2013-03 for feature dti
Density difference of 2013-04 for feature dti
Density difference of 2013-05 for feature dti
Density difference of 2013-06 for feature dti
Density difference of 2013-07 for feature dti
Density difference of 2013-08 for feature dti
Density difference of 2013-09 for feature dti
Density difference of 2013-10 for feature dti
Density difference of 2013-11 for feature dti
Density difference of 2013-12 for feature dti
Density difference of 2014-01 for feature dti
Density difference of 2014-02 for feature dti
Density difference of 2014-03 for feature dti
Density difference of 2014-04 for feature dti
Density difference of 2014-05 for feature dti
Density difference of 2014-06 for feature dti
Density difference of 2014-07 for feature dti
Density difference of 2014-08 for feature dti
Density difference of 2014-09 for feature dti
Rejected loan applications
Accepted loan applications
Density
Relative months after classification


Density difference of 2014-10 for feature dti
Density difference of 2014-11 for feature dti
Density difference of 2014-12 for feature dti
Density difference of 2015-01 for feature dti
Density difference of 2015-02 for feature dti
Density difference of 2015-03 for feature dti
Density difference of 2015-04 for feature dti
Density difference of 2015-05 for feature dti
Density difference of 2015-06 for feature dti
Density difference of 2015-07 for feature dti
Density difference of 2015-08 for feature dti
Density difference of 2015-09 for feature dti
Density difference of 2015-10 for feature dti
Density difference of 2015-11 for feature dti
Density difference of 2015-12 for feature dti
Density difference of 2016-01 for feature dti
Density difference of 2016-02 for feature dti
Density difference of 2016-03 for feature dti
Density difference of 2016-04 for feature dti
Density difference of 2016-05 for feature dti
Density difference of 2016-06 for feature dti
Rejected loan applications
Accepted loan applications
Density
Relative months after classification

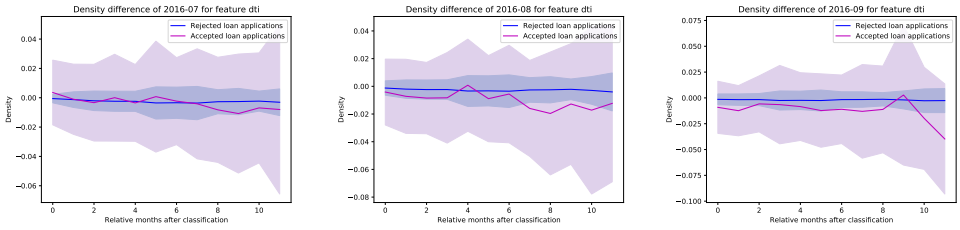


FIGURE C.6: Density difference DTI

Bibliography

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md. Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, February 2016. ISSN 0167739X. doi: 10.1016/j.future.2015.01.001. URL `https://linkinghub.elsevier.com/retrieve/pii/S0167739X15000023`.

[2] C. Alippi and M. Roveri. Just-in-Time Adaptive Classifiers—Part I: Detecting Nonstationary Changes. *IEEE Transactions on Neural Networks*, 19(7):1145–1153, July 2008. ISSN 1045-9227, 1941-0093. doi: 10.1109/TNN.2008.2000082. URL `http://ieeexplore.ieee.org/document/4470009/`.

[3] Samaneh Aminikhanghahi and Diane J. Cook. A Survey of Methods for Time Series Change Point Detection. *Knowledge and information systems*, 51(2):339–367, May 2017. ISSN 0219-1377. doi: 10.1007/s10115-016-0987-z. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5464762/`.

[4] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, 2002.

[5] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006.

[6] Roberto Souto Maior Barros and Silas Garrido T. Carvalho Santos. A large-scale comparison of concept drift detectors. *Information Sciences*, 451-452:348–370, July 2018. ISSN 00200255. doi: 10.1016/j.ins.2018.04.014. URL `https://linkinghub.elsevier.com/retrieve/pii/S0020025518302743`.

[7] Michele Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice hall Englewood Cliffs, 1993.

[8] Albert Bifet and Ricard Gavaldà. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448. Society for Industrial and

Applied Mathematics, April 2007. ISBN 978-0-89871-630-6 978-1-61197-277-1. doi: 10.1137/1.9781611972771.42. URL https://epubs.siam.org/doi/10.1137/1.9781611972771.42.

[9] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavalda. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148, 2009.

[10] Dariusz Brzeziński and Jerzy Stefanowski. Accuracy Updated Ensemble for Data Streams with Concept Drift. In Emilio Corchado, Marek Kurzyński, and Michał Woźniak, editors, *Hybrid Artificial Intelligent Systems*, Lecture Notes in Computer Science, pages 155–163, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-21222-2. doi: 10.1007/978-3-642-21222-2_19.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, July 2009. ISSN 0360-0300, 1557-7341. doi: 10.1145/1541880.1541882. URL https://dl.acm.org/doi/10.1145/1541880.1541882.

[12] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

[13] Allen B. Downey. A novel changepoint detection algorithm. *arXiv:0812.1237 [stat]*, December 2008. URL http://arxiv.org/abs/0812.1237. arXiv: 0812.1237.

[14] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification. john wiley & sons. *Inc., New York*, 2, 2001.

[15] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Rich Zemel. Fairness Through Awareness. *arXiv:1104.3913 [cs]*, November 2011. URL http://arxiv.org/abs/1104.3913. arXiv: 1104.3913.

[16] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with Drift Detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence – SBIA 2004*, Lecture Notes in Computer Science, pages 286–295, Berlin, Heidelberg, 2004. Springer. ISBN 978-3-540-28645-5. doi: 10.1007/978-3-540-28645-5_29.

[17] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

[18] Hanqing Hu, Mehmed Kantardzic, and Tegjyot S. Sethi. No Free Lunch Theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining and Knowledge Discovery*, 10(2): e1327, 2020. ISSN 1942-4795. doi: 10.1002/widm.1327. URL `http://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1327`. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1327.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[20] jeremyDT. jeremydt/p2p-lending-with-ai: Second release with source files, updated paths and link to the published paper in the readme file. Jun 2020. doi: 10.5281/zenodo.3900134.

[21] Yoshinobu Kawahara and Masashi Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining*, 5(2):114–127, April 2012. ISSN 19321864. doi: 10.1002/sam.10124. URL `http://doi.wiley.com/10.1002/sam.10124`.

[22] Anton Khritankov. Analysis of hidden feedback loops in continuous machine learning systems. *arXiv:2101.05673 [cs]*, 404:54–65, 2021. doi: 10.1007/978-3-030-65854-0_5. URL `http://arxiv.org/abs/2101.05673`. arXiv: 2101.05673.

[23] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004.

[24] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2018. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2018.2876857. URL `https://ieeexplore.ieee.org/document/8496795/`.

[25] Kristian Lum and William Isaac. To predict and serve? *Significance*, 13 (5):14–19, October 2016. ISSN 17409705. doi: 10.1111/j.1740-9713.2016.00960.x. URL `http://doi.wiley.com/10.1111/j.1740-9713.2016.00960.x`.

[26] Gabriel Machado Lunardi, Guilherme Medeiros Machado, Vinicius Maran, and José Palazzo M. de Oliveira. A metric for Filter Bubble measurement in recommender algorithms considering the news domain. *Applied Soft Computing*, 97:106771, December 2020. ISSN 15684946. doi: 10.1016/j.asoc.2020.106771. URL `https://linkinghub.elsevier.com/retrieve/pii/S1568494620307092`.

[27] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019.

[28] Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, 22(5):730–742, 2009.

[29] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, 22(110): 1–8, 2021. URL http://jmlr.org/papers/v22/20-1380.html.

[30] Anand M Narasimhamurthy and Ludmila I Kuncheva. A framework for generating data to simulate changing environments. In *Artificial Intelligence and Applications*, pages 415–420, 2007.

[31] Kyosuke Nishida and Koichiro Yamauchi. Detecting Concept Drift Using Statistical Testing. In Vincent Corruble, Masayuki Takeda, and Einoshin Suzuki, editors, *Discovery Science*, Lecture Notes in Computer Science, pages 264–269, Berlin, Heidelberg, 2007. Springer. ISBN 978-3-540-75488-6. doi: 10.1007/978-3-540-75488-6_27.

[32] The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

[33] Eli Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[34] R. Polikar, L. Upda, S.S. Upda, and V. Honavar. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 31(4): 497–508, November 2001. ISSN 10946977. doi: 10.1109/5326.983933. URL http://ieeexplore.ieee.org/document/983933/.

[35] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, pages 377–382, San Francisco, California, 2001. ACM Press. ISBN 978-1-58113-391-2. doi: 10.1145/502512.502568. URL http://portal.acm.org/citation.cfm?doid=502512.502568.

[36] Wenlong Sun, Olfa Nasraoui, and Patrick Shafto. Iterated Algorithmic Bias in the Interactive Machine Learning Process of Information Filtering:. In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 110–118, Seville, Spain, 2018. SCITEPRESS - Science and Technology Publications. ISBN 978-989-758-330-8. doi: 10.5220/0006938301100118. URL http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006938301100118.

[37] Dang-Hoan Tran. Automated Change Detection and Reactive Clustering in Multivariate Streaming Data. *arXiv:1311.0505 [cs]*, November 2013. URL http://arxiv.org/abs/1311.0505. arXiv: 1311.0505.

[38] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167: 107299, February 2020. ISSN 01651684. doi: 10.1016/j.sigpro.

2019.107299. URL https://linkinghub.elsevier.com/retrieve/pii/S0165168419303494.

[39] JD Turiel and T Aste. Peer-to-peer loan acceptance and default prediction with artificial intelligence. *Royal Society open science*, 7(6):191649, 2020.

[40] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[41] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[42] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, 2003.

[43] Heng Wang and Zubin Abraham. Concept Drift Detection for Streaming Data. *arXiv:1504.01044 [cs, stat]*, May 2015. URL http://arxiv.org/abs/1504.01044. arXiv: 1504.01044.

[44] Scott Wares, John Isaacs, and Eyad Elyan. Data stream mining: methods and challenges for handling concept drift. *SN Applied Sciences*, 1(11):1412, November 2019. ISSN 2523-3963, 2523-3971. doi: 10.1007/s42452-019-1433-0. URL http://link.springer.com/10.1007/s42452-019-1433-0.

[45] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, July 2016. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-015-0448-4. URL http://link.springer.com/10.1007/s10618-015-0448-4.

[46] Geoffrey I. Webb, Loong Kuan Lee, François Petitjean, and Bart Goethals. Understanding Concept Drift. *arXiv:1704.00362 [cs]*, April 2017. URL http://arxiv.org/abs/1704.00362. arXiv: 1704.00362.

[47] Geoffrey I. Webb, Loong Kuan Lee, Bart Goethals, and François Petitjean. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, 32(5):1179–1199, September 2018. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-018-0554-1. URL http://link.springer.com/10.1007/s10618-018-0554-1.

[48] Xiaodan Xu, Huawen Liu, and Minghai Yao. Recent Progress of Anomaly Detection. *Complexity*, 2019:1–11, January 2019. ISSN 1076-2787, 1099-0526. doi: 10.1155/2019/2686378. URL https://www.hindawi.com/journals/complexity/2019/2686378/.

[49] Indrė Žliobaitė, Mykola Pechenizkiy, and João Gama. An Overview of Concept Drift Applications. In Nathalie Japkowicz and Jerzy Stefanowski, editors, *Big Data Analysis: New Algorithms for a New Society*, volume 16, pages 91–114. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26987-0 978-3-319-26989-4. doi: 10.1007/978-3-319-26989-4_4. URL http://link.springer.com/10.1007/978-3-319-26989-4_4. Series Title: Studies in Big Data.