Bank fraud simulation, utilizing synthetic data to detect to test fraud detection strategies.

KEYWORDS: Bank Fraud, Monte Carlo Simulation, Synthetic Data, Stochastic Gradient Dissent, Gradient Boosting Classifier, Concept Drift

Stephen Montgomery, DePaul University, School of Computing, smontg12@depaul.edu
John Bolgert, DePaul University, School of Computing, jbolgert@depaul.edu

ABSTRACT

Fraud continues to negatively impact consumers, providing a solution in itself has become a new industry. Accurately classifying fraud and preventing fraudulent transactions faces many challenges. Many solutions have been proposed utilizing machine learning to overcome this classification problem. The Fraud-Detection Systems (FDS) deployed today are facing many challenges needing to be overcome to effectively identify fraudulent transactions. Genuine consumer behavior and fraudulent techniques are constantly changing in parallel; this phenomenon is referred to as concept drift which will invalidate the data model. Most data obtained is highly imbalanced as fraudulent transactions typically make up less than 1% of all transactions. False negatives and false positives do not carry the same cost to the business, this feature in tandem to the high imbalanced creates an issue with using an appropriate performance metric within the model. In our paper, we plan to address these core problems in many ways. First, we propose utilizing a custom cost function along with F1 score to overcome the excessive cost of false positives. In addition, we are implementing concept drift detection mechanism, that will retrain our model if concept drift has been detected utilizing statistical test. To evaluate these solutions, we will be running a simulation to test the concept drift mechanism when the population distribution shifts.

INTRODUCTION

Consumers are being negatively impacted by fraud at an increasing rate. In 2022 alone, consumers lost 8.8 billion dollars in the United States due to fraudulent transactions [16]. This figure was a 30% increase from that of 2021. As fraud continues to negatively impact consumers, the need for a solution will continue to rise. The global fraud detection and prevention market share is expected to grow from $43.9 billion in 2023, to that of $183.6 billion by 2030 [17]. As the world continues to digitize fraud is an ever-increasing presence in the world, especially online. There are many solutions that are currently being implemented to combat fraud, but they all have their disadvantages. The models we plan to test in our fraud detection system will be that of Random Forest Classifier, KNN Classifier, MLP Classifier, Stochastic Gradient Descent Classifier (SGD) and Gradient Boosting Classifier (GBC). Today most machine learning solutions that have been developed to counter fraud face three main problems, class imbalance, performance evaluation, and concept drift. In our study we plan to focus on attempting to resolve these three main problems.

Fraudulent transactions make up around 1% of total transactions that take place. Most data sets if not all that are used to train models to detect fraud will have this imbalance. Class imbalance is not a problem that is unique to fraud so we plan to leverage many solutions that have been tested in other industries as well as fraud to overcome the impact class Imbalance will have on our model. We will be utilizing random oversampling (ROS), random under sampling (RUS), and syntenic minority oversampling technique (SMOTE), to address this first problem.

The second problem most fraud detection systems face today, is the assumption that the cost of a false positive and false negative is the same. The cost to an organization is not the same as false positives still can obtain a high cost due to the resources needed to investigate the claim. To resolve this, we plan to implement a custom cost function to allow a firm to optimize towards decreasing the cost associated with fraud based on the input values they give. For our study we will be utilizing previous research done to use input values that represent the average across the industry. We will also be optimizing our model's performance with f1 score.

The third problem faced by fraudulent detection systems is the parallel change in consumer behavior and fraudulent actors. This change in distribution, also known as concept drift, invalidates a model as the data used to train does not represent the current population. To combat this third issue, we will be developing a mechanism that will detect if concept drift has occurred within our dataset. If found, this mechanism would trigger model retraining so the model is more in tuned with the current strategies that fraud actors are deploying. In order to test this mechanism will be running a 20-day simulation where we will be introducing a shift in the population of our synthetic data to test the performance of our concept drift mechanism as well as how the model performance changes with that of retraining. These are the three main problems and solutions we are planning to implement as part of our study.

LITERATURE REVIEW

Class Imbalance

Fraudulent transaction databases all share a common issue, that of class imbalance. Fraud typically makes up a small percentage of the overall total transactions. The low percentage of fraudulent transactions creates an imbalance between the majority and minority class. This imbalance causes the model to be biased towards the majority class as it treats the minority class as noise. [11]. Class imbalance issue is not unique to that of fraudulent transactions but is prevalent in many other industries. Techniques that can be utilized to overcome the issue with class imbalance for fraudulent transactions are that of random oversampling *(ROS),* random under sampling *(RUS),* synthetic minority oversampling technique *(SMOTE)* [12]. ROS method randomly duplicates examples from the minority class, creating new datapoints until a more balanced distribution is achieved. This oversampling approach method is widely used and highly effective. [13] In contrast to ROS that create new data records, the RUS method randomly deletes majority class instances until a more balanced distribution is achieved. RUS is the most common and effective under sampling technique used to tackle class imbalance [14] SMOTE, is another oversampling technique first proposed by reference [15], that generates synthetic data from the minority class data instances. This is not an exhaustive list of techniques that are available to mediate the issue of class imbalance, but these are the techniques we will be utilizing within our study due to the reliability and performance these methods produce.

Cost Based evaluation

Fraud has a high negative cost to any firm, however, there is also a cost associated with false positive events. Classification algorithms that utilize accuracy to measure performance assume all misclassifications carry the same cost. Due to the need currently for human intervention to investigate flagged records, there becomes an excessive cost to these false positive events. False positive records account for nearly 19% of the total cost of fraud, while fraud itself only accounts to 7%.[7] To focus on a high accuracy of model performance, assuming all misclassification has the same cost, would surely eliminate the cost of fraud on a firm but in turn has the potentially to further increase the cost of false positive events due to the influx of resources needed to continue to investigate all flagged items. According to [6], In 2014 alone 114 billion transactions were blocked due to being flagged fraudulent while the true cost of fraud for that same year alone was 9 billion. When optimizing the model for performance, assuming all misclassification costs are the same creates a risk for a firm to increase the operational costs with combating fraud.

The issue resulting from assuming all misclassifications costs being the same is not a novel one, this has been investigated by many authors. Focusing on minimizing the total cost rather than minimizing the number of fraudulent records entirely would allow a firm to optimize profits. Utilizing a cost-based evaluation for fraud detection was first proposed in 1999 by [8]. Determining the cost of a false positive event is a complicated endeavor as there would be many implicit costs in tandem with the hard costs for resources needed. From a customer experience standpoint having a fraudulent transaction occur to them with no investigation would surely hurt their image of the respective firm they use. The inverse of that is having a transaction blocked that is not deemed fraudulent, also has the potential to hurt their image with their customers. Two thirds of customers who had a transaction blocked due to a false positive event reduced and/or stopped their patronage with the respective merchant. *[5]* This could have a greater cost than initially understood to the brand image at risk with managing the tradeoffs of false positive and false negative events.

This cost would not be the same across all firms and for all false positive incidents. It would vary dramatically by a firm's operation costs and their protocols in place to handle fraud investigations. This is data that many firms hold private due to the insights it would provide to their respective competitors. Therefore, to determine the industry standard of the cost of a false positive we are relying on the previous work done by [1] which determined an overhead cost of $10 in case of false positives. This value was derived by utilizing the Statistical Data warehouse of the ECB from 2018.

Concept Drift

A significant hurdle that many fraud detection systems face today is the ability to adapt to the fast-changing methods that fraudsters utilize to commit fraud. Not only do fraudsters change their tactics to evade detection but consumer spending habits change throughout the year due to holidays and broader economic conditions. The change in the underlying data distribution is referred to as Concept Drift. The issue that concept drift proposes with fraud detection is that the data that models are being trained on do not represent the latest tactics that fraudsters are utilizing today. Reference [10] was one of the first authors to do a comprehensive review of real-world data to prove that concept drift is occurring within fraud detection. They analyzed over 54 million transactions over 10 months and successfully demonstrated that concept drift is affecting the data stream for fraudulent records.

Concept drift is not unique alone to fraud detection but occurs in many other applications. To resolve this issue, one must first be able to detect that concept drift is occurring. Reference [9] was able to successfully identify concept drift within synthetic data utilizing the statistical test of equal proportions. This represents an active approach to concept drift detection. Once concept drift is identified, an active approach then retrains the classifiers so that most recently observed samples are considered to represent the current population of fraudsters. [10]. We will be utilizing an active approach to detect concept drift within our study.

Models

Due to the high impact fraud has, a wide variety of models have been tested to predict fraud by previous studies. A comprehensive review of literature produced between 2009-2019 found the following techniques tested to predict fraud includes, Support Vector Machines (SVN), Hidden Markov Model (HMM), Artificial Neural Network (ANN), K-nearest neighbor (KNN), Bayesian Network (BN), Decision Tree (DT), and Logistical Regression (LR). [22] Since 2019 there has been other studies that have been utilizing other techniques. Work performed by [18] has shown that Gradient Boosting Classifier works well in predicting bank fraud. In additional [19] has shown that Random Forest Classifier model performed well with predicting fraud. There has also been a lot of work done in comparing the performance of these models against each other. A study performed by [21] found that ANN models outperformed that of logistical regression models. [20] did a comparative study of KNN, Naïve Baines, logistic regression and found KNN model to best performance of the three with predicting fraudulent transactions. Due to the high need to accurately identify fraud, many studies have been conducted testing a variety of machine learning techniques to overcome this classification problem. Within our study we will specifically be focusing on testing the initial performance of Random Forest Classifier, KNN Classifier, MLP Classifier, Stochastic Gradient Descent Classifier (SGD) and Gradient Boosting Classifier (GBC)

THEORETICAL DEVELOPMENT

Model Testing & Selection

The first phase of our study was to select two models that would be utilized within our 20-day simulation. The models that were selected were chosen based on which models had the best performance in terms of their f1, cost savings and computation load time. Only two models were chosen due to the computation constraints we faced with running all models and optimizations within our 20-day simulation.

Data Exploration

For our study, we will be utilizing a bank account fraud dataset from Kaggle. This data set contains 1,000,000 records. The data has thirty-two columns, three of which are categorical variables, seven binary variables, and twenty being numerical variables. For a complete list of all columns that were available within this dataset please see Table 9 found in the appendix. The dataset is highly imbalanced with just around 1% of the records being fraud. Six of the variables have null records, being populated with negative values to indicate a null record.

Before model development and selection, we used the python graphical package Matplotlib to plot the distributions of the fraud and non-fraud records for each numeric feature to identify outliers or any differences observed between the distributions. This analysis revealed that some of the numeric features contained negative values. These negative values were used to flag null records within the data collection process of the dataset we utilized within our study. For the categorical features we plotted the fraud and non-fraud relative percentages for each feature. This gave us insight into which values of a feature typically contain fraud for a given categorical feature. The distributions plots for both the numeric and categorical features can be found in the appendix in figures 10-39.

Data Preprocessing

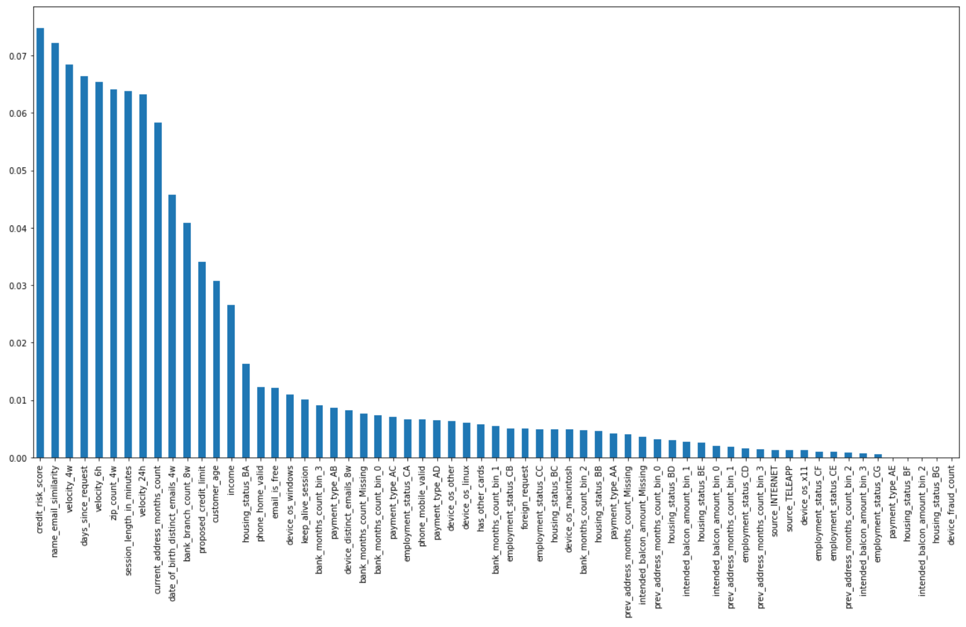
Initially we tried to use the entire banking fraud dataset to optimize our model but quickly found out the size of the data would cause memory and storage issues for our simulation. To prevent this, we decided to perform all analysis on the 7th month segment of the dataset which represented around 96,000 records. This segment of the overall dataset is also used as the base data for the simulation.

First to handle negative values which indicated a null record, we measured the count of instances a negative value occurred for fraud within our dataset. If the fraud records for a feature contained negative values for more than 10%, we then treated that feature as categorical and discretized the data giving the negative values their own bin. If there was less than 10% missing data in the fraud records, we filled the negative values with the median value of that feature. Using this method, we converted the following features into categorical variables using equal width binning: prev_address_months_count, intended_balcon_amount and bank_months_count. Additionally, we treated all numeric values with less than 5 unique values as a categorical feature.

After handling the missing values, we converted the categorical features into numeric features using one hot encoding, then applied a min max scaling function to normalize the data by setting all values of the features to values between 0 and 1. After processing the data, we then split it into training and testing segments of 67/33 train/test split.

Finally, we used Scikit-learn feature importance algorithm from its Random Forest Classifier package to rank the importance of all 63 features. The ranking of all 63 features can be found in Figure 1. This figure displays that the features that had the highest ranking were that of credit_risk_score, name_email_simularity and velocity_4w.

Figure 1: Feature Importance Ranking



Model Testing

Prior to running the simulation, we tested different combinations of predictive models, feature amounts, sampling techniques, and sampling ratios. This optimization was done in two phases. First, we iterate over each combination of feature amount, sampling technique and sampling ratio. Then for each of these combinations we run a model specific grid search optimization to obtain the optimal model. Table 1 below shows the different feature amounts, sampling techniques and sampling ratios that were applied to each model object.

Table 1: Model Parameters used in Grid Search

Feature Amount	5, 10, 15, 25, 40, 63
Sampling Techniques	Random Under Sampling (RUS), Random Over Sampling (ROS), Synthetic Minority Oversampling (SMOTE)
Sampling Ratios	25%, 50%, 75%

All models used were built utilizing the Scikit-learn library. All sampling techniques utilized the Imblearn library to perform the associated sampling technique. The model specific grid search also utilized Scikit-learn's grid search package. These models additionally went through a model specific optimization grid search after the combination of feature amounts, sampling techniques, and sampling ratios were applied to the base data. These model specific grid searches are depicted below in Table 2.

Table 2: Model Specific Paramaters used in Grid Search

Model	Model Parameters
Random Forest Classifier	n_estimators' : [25,50,75], 'max_depth':[2,4,5]]
KNN Classifier	n_neighbors' : [3,5,10], 'weights':['uniform','distance']
MLP Classifier	hidden_layer_sizes':[(20,10,5,1),(8,4,1),(4,2)], 'activation':['logistic', 'relu'], 'solver':['sgd','adam'], 'max_iter':[10,20,35] 'learning_rate_init':[.0001,.001],
Stochastic Gradient Descent Classifier (SGD Classifier)	alpha':[.01,.001,.0001], 'max_iter':[200,1000,2000]
Gradient Boosting Classifier (GBC Classifier)	n_estimators':[50,100,200], 'max_depth':[2,3,5], 'min_samples_split':[2,10]

To assess the predictive power of these models we measured each model's f1 score recall and its cost savings using **S**cikit-**l**earn's metrics package. To determine the cost savings of a model we created our own cost savings equation that would be applied to each model. The equation can be found below. S represents the savings of the model, TP represents the amount of correctly predicted fraud, FP represents the amount of false positives, CFN represents the cost of a false negative, and CFP represents the cost of a false positive. This cost function was designed so that it can be customized by each respective firm to use their own CFN and CFP. In the case of our study, for CFP we used a cost of $10 which we derived from the previous work of [1] to determine an industry standard. For CFN metric we used the median transaction amount of fraud instances from our base dataset which turned out to be $34.7.

$$S = TP * CFN - FP * CFP$$

For each possible combination of feature amount, sampling technique and sampling ratio we optimized the individual model based on its model specific parameters outlined above. For each of these combinations we measured the optimized models optimal f1 score, recall and our custom savings function dollar amount. To find the optimal f1 and cost savings amount we designed an optimization function that used the predicted probability of all test records of the optimized model. This optimization function iteratively measured the f1 score and cost savings at each 1% probability threshold of the test records. The optimization function then returns the optimal probability threshold and metric per optimal f1 and cost savings function. After all iterations were tested, we plotted the optimized cost savings, f1 score and recall metric per combination of sampling technique and sampling ratio per model per feature amount tested. The full results of each model can be found in the appendix in figure 40-57.

We decided to select the models with the best f1 and cost-saving metrics as these metrics represent a blend between recall and precision. Additionally, we recorded the time it took for each model to test each combination and produce the results. We include time taken to optimize in our selection criteria as our simulation would require the model to retrain and predict and re-optimize many times. Below Table 3 displays the computation time in seconds each model took to complete its full optimization.

Table 3: Model Computation Time

Model	Time Taken (Seconds)
Random Forest Classifier	2,471
KNN Classifier	9,862
MLP Classifier	22,233
SGD Classifier	188
Gradient Boosting Classifier	92,028
Ada Boost Classifier	13,982

We observed that the models with higher f1 scores also tended to have higher cost savings. Almost every model showed a clear positive trend between the f1 score and cost savings with the number of features used in the optimization. After considering the f1 scores, cost savings and time taken to optimize we selected the Stochastic Gradient Descent Classifier (SGD) and the Gradient Descent Classifier (GBC) to use in our simulation. The SGD was the fastest to optimize and had relatively high f1 and cost savings metrics. The GDC took the longest to optimize but achieved the highest f1 score and cost savings of all the models tested. The optimal combinations per feature amount for the SGD and GDC models is summarized below in figures 2-4 for their f1 and cost savings metrics. Additionally, table 4 below displays all the python software packages that were utilized within our study.

Figure 2: Stochastic Gradient Descent Optimal cost savings

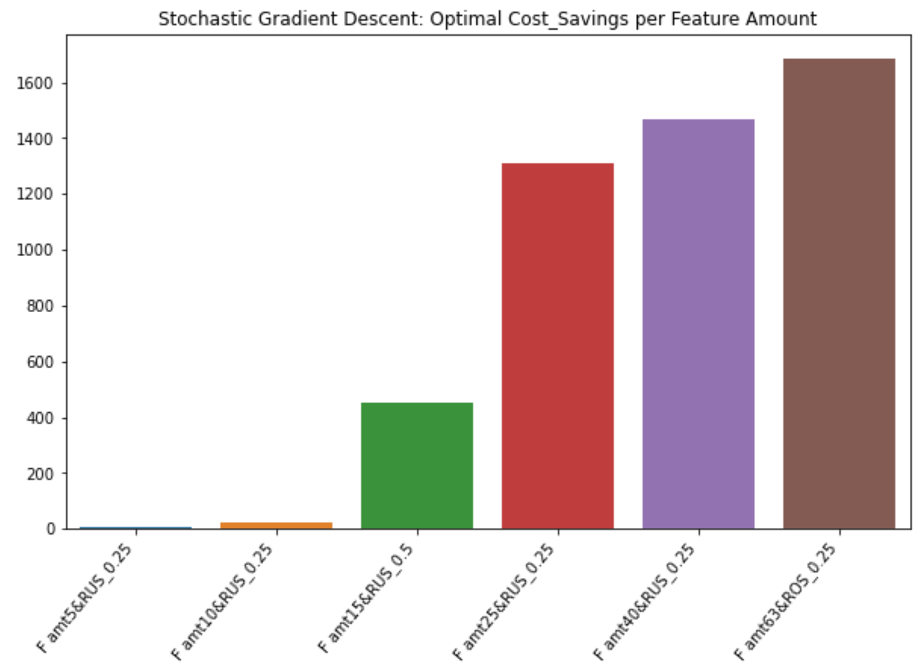


Figure 3: Stochastic Gradient Descent Optimal F1 Scores

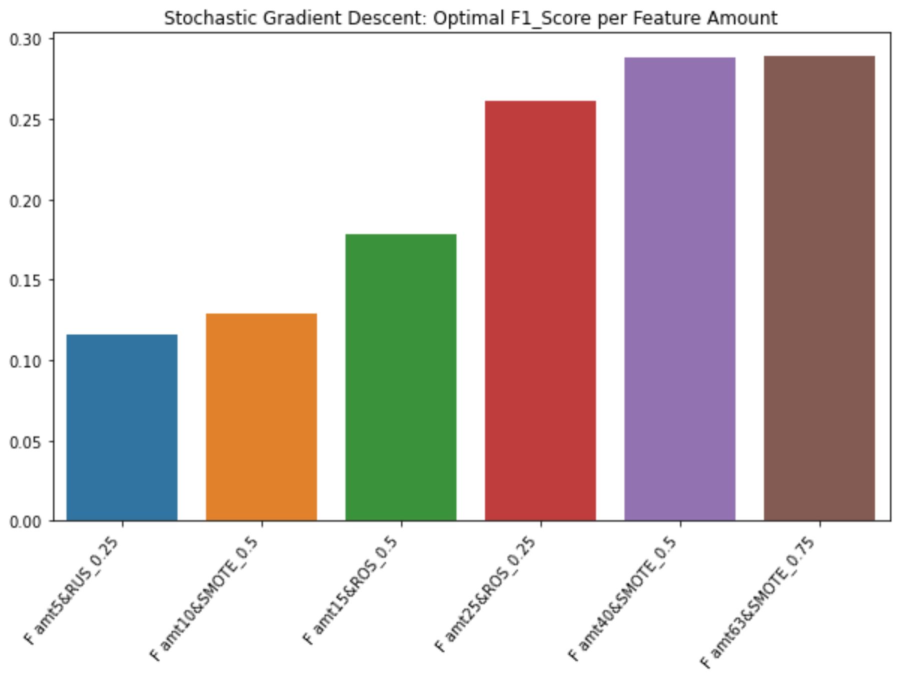


Figure 3: Gradient Boosting Optimal Cost Savings

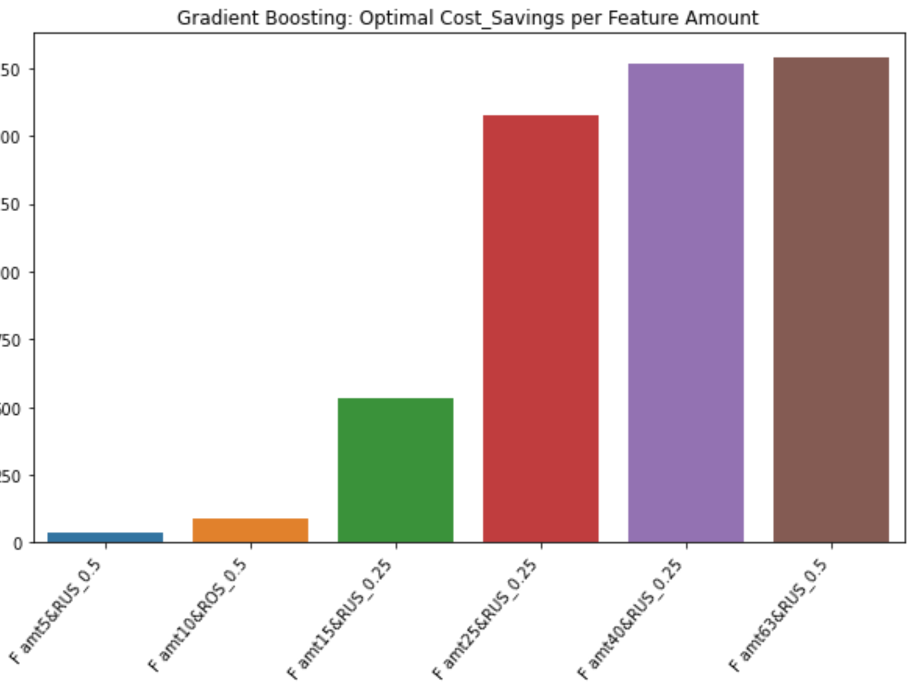


Figure 4: Gradient Boosting Optimal F1 Scores

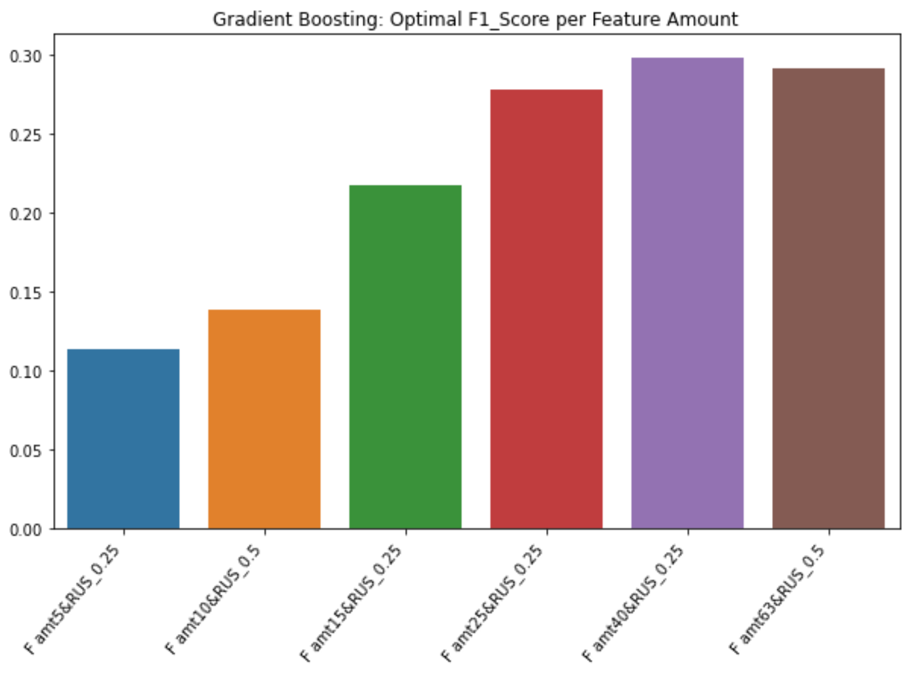


Table 4: Python packages utilized

Python Package	Description
Pandas	Pandas python package was utilized to create data frame and perform data transformations.
NumPy	NumPy python package was utilized to perform array transformations.
Seaborn	Seaborn python package was utilized for data visualization within our study.
Matplotlib	Matplotlib python package was utilized for data visualizations within our study.
Sklearn	Sklearn python package was utilized in the creation of all the models tested within our study along with grid search function and performance metrics utilized.
Imblearn	Imblearn python package was utilized for Random over sampling (ROS), random under sampling (RUS), and SMOTE imbalanced sampling techniques.
SDV	Synthetic Data Vault python package was utilized for synthetic data generation within our study.
SciPy	SciPy python package was utilized for stat functionality.
Time	Time python package was utilized to determine model runtime/optimization during our study.

METHODS

Simulation Overview

Our simulation measured f1 score, recall, precision, accuracy and cost savings of the 18 different strategies that were tested. These 18 strategizes consisted of a combination of model, retrain method and concept drift detection method. The models, concept drift detection methods and retrain methods are described in Table 5 below. In addition to having an associated model, retrain method and concept drift detection method our models have a model dictionary that stores all the necessary objects used in the simulation. This model dictionary contains 14 values, some of which remain the same from iteration to iteration while others are updated whenever retraining occurs. Each strategy has the same model dictionary but may have different values stored in the respective positions. Table 5 summarizes what is stored in each strategies model dictionary

Table 5: Model Dictionary

Position	Object Name	Description	Updated during retraining	Reset after iteration
0	Model instance	Used to predict fraud probabilities	Yes	Yes
1	Re training method	Used to decide what data to retrain on	No	No
2	Concept drift detection method	Used to decide which concept drift detection algorithm to call	No	No
3	F1 training metric	Used in performance concept drift detection algorithm	Yes	Yes
4	Base data	The original base data	No	Yes
5	Training Fraud Probability distribution	Used in population stability index concept drift detection algorithm	Yes	Yes
6	Model optimization grid search dictionary	Used to fit the strategy's model to new data during retraining	No	No
7	Optimal fraud prediction probability	Used as a minimum probability threshold for strategy to predict fraud	Yes	Yes
8	Model Results	Data frame used to store model results for each day of each iteration	No	No
9	Detection Results	Data Frame to tack each time the strategy detected concept drift for each day of each iteration	No	No
10	Required columns	List to ensure that each new day's data has the same columns that the base model was trained on	No	No
11	Normalization Object	Used in preprocessing to normalize the new simulated data	Yes	Yes
12	Last day concept drift was detected	Used by concept drift detection algorithms to filter the simulated data	Yes	Yes
13	Fraud probability distances	Used by the feature distribution concept drift detection algorithm to compare current and previous days distances	Yes	Yes

Our simulation is an iterative process that contains two loops. The outer loop is the number of iterations the simulation runs for while the inner loop is the number of days or new simulated data sets each iteration creates. At the start of each iteration, all strategies are initialized back to their original state. Within each iteration there are 2 phases.

During phase 1 the new days data is simulated. This new data is simulated based on the previous day's data. For days 4, 9 and 14 the simulated data fraud feature distributions are randomized to represent concept drift. After the new data is synthesized, it is added to a data frame that holds all data for the current iteration. No retraining occurs within the first three days of the simulation. This is because this serves as our baseline metrics and statistics that are utilized within our concept drift detection algorithm. After the third day the strategies may begin to detect concept drift and retrain when necessary.

In phase 2 the simulation iterates over each of the 18 strategies and each strategy makes predictions on the new days simulated data and attempts to detect concept drift. This process is broken down into the following subphases.

Subphase 2a preprocessing: during this phase the strategy preprocesses the base data using its min max scaler object and previously seen features. Each strategy has the potential to have different scaler objects used to normalize the data and thus the result of normalization could look different from strategy to strategy. The results of this phase are two data frames holding the dependent and independent feature values that are used by the model to make predictions and calculate its performance metrics for that day.

Subphase 2b predictions: Next the strategy uses its predictive model and optimal fraud probability metric to produce fraud probabilities for each record of the newly simulated data. Then the optimal fraud probability is used as a minimum threshold to predict fraud. The resulting fraud predictions are used to measure the strategies f1 score, recall, precision, accuracy and cost savings metrics for that day of that iteration.

Subphase 2c concept drift detection: finally, the strategy deploys its concept drift detection method. If its method detects concept drift the algorithm will initiate retraining based upon the strategies retraining method. The resulting retrained model and associated values needed for future concept drift detection are then updated in the strategies model dictionary.

Simulation data preparation

To prepare the data for the simulation we first imported the base data using Pandas and filtered it to only use month 7 of the overall data. We also saved month 6’s non fraud records to be included in the simulation as noise throughout each iteration of the simulation. After importing the data, we discretized features with the negative values that had greater than 10% instances for the fraud records. For the features that did not meet this condition we replaced the negative values with the median value of the feature. Additionally, if a numeric feature had less than 5 unique values, we directly converted it to a categorical feature. If the feature had less than 25 unique values, we split the values into two bins depending on the values relationship to the median value of the feature. Finally, if the numeric feature had less than 100 unique values, we used equal width binning to discretize the numeric feature into 5 bins. Next, we transformed all categorical variables into numeric variables using one hot encoding. Finally, we normalized the data by applying a min max scaler object to the independent variables to normalize their values between 0 and 1. The resulting data from month 7 was used as our base data.

Base Model Training

After preparing the base data we split it into training and testing data using a 67/33 train/test split. This training and testing data was used to optimize the base SGD and GBC models. Additionally, during this step, we used our optimization function discussed in the model exploration section to obtain the training f1 metric and optimal probability threshold to predict fraud. The SGD model utilized a Random oversampling with a sampling strategy of 25% whilst the GBC model used random under sampling with a sampling strategy of 50%. These sampling strategies and sampling ratios were chosen based on the optimal combination observed in the cost savings metrics during the model exploration phase. After training and optimizing the models, we saved the following objects to be used in the simulation - training f1 metric, model instance, fraud probability threshold, fraud prediction probability distribution, optimization dictionary used during the grid search optimization, and normalization min max object

Synthetic data creation

To create new synthetic data the model had not seen before we utilized a combination of the Synthetic Data Vault's (SDV) Gaussian Copula Synthesizer along with injections from the base data the model had not seen before. The SDV’s Gaussian Copula Synthesizer uses Gaussian Copulas to learn the overall distribution of the real data. It accomplishes this by learning the distributions of each feature, normalizing the values, then measuring the covariance of each pair of features. We also performed sampling without replacement records from month 6 of the base data. We chose this month as the base since the model had not seen this data before.

At the beginning of each day of an iteration we run the synthetic data creation function. This function intakes base data which is the data from the previous day instance. For day 0, Month 7 of the original dataset is utilized. The function then splits the base data passed to it into fraud and non-fraud datasets. The non-fraud data set is then passed to the SDV Gaussian Copula Synthesizer to create 20,000 new records of which 1% would be fraud. The simulated data using the SDV package is very similar to the base data it created the simulated data from. Thus these 20,000 new records are very similar to what the model has seen before. To represent a gradual drift in the non-fraud feature distributions, we insert 1,333 or 15% new records into the existing data overwriting 1,333 non fraud rows of the simulated data. These 1,333 non-fraud records come from month 6 of the base data. Any time a record is chosen to be inserted into the new simulated data during an iteration it is marked and not used again until the iteration ends.

If the current iteration day is not 4, 9, or 14 the fraud base data is used to generate new fraud records using the SDV Gaussian Copula Synthesizer. The resulting fraud records will make up 1% of total new day's simulated data. If the iteration day is 4, 9 or 14 then the fraud distribution is randomized. To accomplish this, we pass the non-fraud base data and new simulated fraud data to a noise insertion function. This function takes the base non-fraud data and iterates over each feature. If the feature is numeric, it measures the boundaries of 5 equal sized buckets of the non-fraud feature distribution using Panda's binning function. These 5 buckets are then assigned a number between 1 through 100. These 5 numbers are then divided by their sum to get 5 percentages that add up to 1. These 5 percentages then represent the percentage of new data that will come from the respective numeric feature bin. We then iterate through these 5 percentages using python's random package to select a new pseudo random number that falls between the bin's boundaries. This pseudo random number is then added on to our new fraud feature list. We do this x number of times such that the new number of generated values for the bin equals the number of new fraud records we are adding for each respective bin. For categorical features we follow the same procedure except instead of bins we assign random percentages to each of the categorical feature values. After assigning percentages to each value, we then generate that value the desired number of fraud records. The number of fraud records determined for a features value, is by taking the value frequency and multiplying by the total number of new fraud records needing to be added. After generating new values for numeric or categorical features these new values are shuffled to ensure that they are not sorted within their bins. Then the new values are added to the new fraud synthetic data frame overwriting the values synthetically produced by the SDV.

Each day of each iteration we run the synthetic data function to simulate 20,000 new records of which 1% will be composed of fraud records.

Simulation Strategies

We use the term strategies to represent a single unique combination of model, concept drift detection method and retrain method. In total we use 18 different strategies which are composed of combinations of two models, three concept drift detection methods, and three retain methods. Each strategy is independent of each other and stores its own results throughout the entire simulation. One strategy retaining does not impact or change another strategies model.

Models

Our strategies utilize the SGD and GBC model objects from the Scikit-learn library. These model objects are optimized in the base model training phase. Then using a deep copy each of the base models are multiplied 9 times each and assigned to their respective strategies. We choose these models for their efficiency and high f1 metrics observed during our model selection phase.

Concept Drift Detection Methods

The goal of the concept drift detection methods is to identify when the fraud data has changed enough that the model requires a retrain. Thus, every concept drift detection method will either detect drift and initialize model retraining or not detect drift and thus not alter the strategies model during that day. In the simulation the only time a model is retrained is when its concept drift detection method detects drift. For the simulation we created three concept drift detection methods. These methods are performance comparison, feature distribution comparison, and population stability index calculation.

The performance comparison concept drift detection method compares the model's training f1 metric to that of its current f1 metric and if the current performance is .15 less than the training performance the algorithm initiates a retrain for the strategy. After retraining the algorithm will use the new training f1 score as the new training metric for future comparison for the strategy.

The feature distribution comparison concept drift detection method identifies significant changes in the fraud features by comparing the current days fraud feature distribution to the past day's fraud feature distributions. To accomplish this, we use several statistical measurements. First, we divide the current iteration’s entire data into two sets for comparison. To accomplish this, we split the data between the current day's fraud records (new data) and all previous day’s fraud records between the current day and last day that concept drift was detected (base data). In the scenario the algorithm never detected drift the base data would be all previous day's fraud records for the current iteration. After splitting the data, we then iterate through each feature comparing the difference between that feature in the base and new data. If the feature is numeric, we will be utilizing the SciPy stats package to run a two-sample Kolmogorov-Smirnov test for goodness of fit. If the resulting p-value returned by this statistical test is less than .05 we consider that feature drifted. Due to the sparsity of the values within the categorical features we applied two different statistical tests to measure fraud distribution changes were appropriate. For categorical features with the help of the Panda's library we convert the features into frequency representations of their values then combine the new and base feature frequency vectors into one data frame where the index is the feature values and the columns represent the original dataset. The cells of this data frame are frequencies of the respective value for the dataset. We then normalize each column. Doing this ensures each column adds up to 1. If no 0 values are present within the feature frequency vectors, we utilize the SciPy stats package to compare the two vectors using a Chi-square test for independence. If the resulting p-value is less than .05 we consider the feature drifted. In some cases, one of the feature columns may contain 0 values meaning that the value was not observed in the distribution. In these cases, we cannot use the Chi-square test for independence. Instead, we utilize the SciPy spatial libraries distance package to compute the Jensen-Shannon distance between the two frequency vectors. This distance represents the difference between the new probability distribution and old probability distribution. A greater distance than previous days may indicate that the probability distributions have changed, and drift has occurred. To detect drift using this distance metric we compare the current days distance to the distribution of the previous day's distances omitting days that concept drift was detected. If the current distance is greater than 1.5 standard deviations above the mean distance, we consider the categorical feature drifted. After applying the appropriate statistical test to each of the feature sets, we count how many fraud feature distributions drifted. If more than 12 features drifted the algorithm initiates retraining based on the strategies retraining method. Upon completion of retraining, the algorithm marks the current day as the last day drift was detected and stores the day, the new model, the new optimal probability to predict fraud, fraud probability distribution and scaler object in the strategies model dictionary.

The Population Stability Index (PSI) concept drift detection method compares the current day’s fraud probability distribution with the previous training fraud probability distribution. Using the Panda's library, the two fraud probability distributions are discretized into bins, then the relative percentage frequencies of each bin are compared. The PSI formula used is shown below. $p(x_i)$ represents the probability (percent frequency) of the actual (base data) at the current bin level and $q(x_i)$ represents the probability (percent frequency) of the observed (new data) at the current bin level.

$$\text{PSI} = \boldsymbol{sum} \sum \boldsymbol{p(x_i) - q(x_i)} \cdot \ln\left(\frac{p(x_i)}{q(x_i)}\right)$$

If the PSI calculation was greater than .1 the algorithm initiates re training for the strategy. After retraining is complete the algorithm updates the model, optimal probability to predict fraud, fraud probability distribution and scaler object created during re training.

Retrain Methods

Our 18 strategies use one of three different retraining methods. All retraining methods are similar to each other in all areas except for the data that they use to re-train on. For all retraining methods we use the Imblearn underdamping RandomUnderSampler package to perform random under sampling with a sampling ratio of 50%. All retraining methods follow the same procedure flow. First, they are only called if the strategy's concept drift detection mechanism detects drift. Once they are called, they are fed all the current iterations simulated data. Then depending on the method, they filter the overall data to just the data that will be used to retrain on. This re-training data is then pre-processed. In the processing phase the re-training data is normalized and the categorical variables are transformed to numeric variables via one hot encoding. Additionally in the preprocessing phase any new columns that were not in the original base data used to create the base models are dropped and any columns not present in the re-training data that were present in the base data are added with all values set to 0. The resulting data is then sampled using random under sampling with a sampling strategy of 50%. We used random under sampling as this sampling technique significantly speeds up the simulation run time allowing for more iterations to be added to the overall simulation. The strategies model is then retrained using the fit method and its grid search dictionary. The retrain method then returns the new model, and scaler object used in normalization to the concept drift detection method. The concept drift detection method then updates the strategies model dictionary with the new values needed to continue the simulation. The values that get updated each time a model is retrained, and their respective uses are outlined in Table 6 below. These values are unique to each strategy and stored in its model dictionary and are only updated if the model is retrained or at the end of an iteration.

Table 6: Model Object Values

Model Object Update Value	Uses
concept drift metrics data frame	Performance data frame that contains each day/iteration the strategy detected drift and the days that drift actual occurred
Strategy's predictive model	Used to predict fraud and get fraud probabilities for each day of each iteration
Training F1 Score	Used in the Performance Concept drift detection algorithm to decide if drift has occurred
Optimal Fraud Prediction Probability	Used as the lowest probability threshold in which to predict fraud for each days/iterations new data. This optimal probability is changed and optimized each time re training occurs.
Normalization scalar object	This object is what was used to normalize the data during re training and will replace the old object. This object is used at the beginning of each day to normalize the new simulated data to prepare it for model predictions
Training fraud probability distribution	The training fraud probability distribution is used within the population stability index concept drift detection algorithm to decide if the strategy requires retraining or not.

Retrain method 1 uses the current days data as the base data to retrain on. Retrain method 2 uses the past 3 days data to retrain on and retrain method 3 uses the past 3 days of data but gives more emphasis to the current days data by altering the sampling strategy used in the random under sampling technique. Retrain method 3 uses a 50% sampling strategy on the current days data and 16% on the other two previous days data. By sampling this way, the current days data will be given more weight in the retraining data. A summary of each of the retraining methods is depicted in Table 7.

Table 7: Retrain Methods

Retrain Method	Data Used
Method 1	Current days data
Method 2	Current day + previous 2 days data
Method 3	Current day + previous 2 days data with higher sampling strategy given to current days data

Retraining Optimization Function

Upon retraining each strategy regardless of retrain or concept drift detection method will select a new optimal fraud threshold probability in which it will predict a record to be fraud. This threshold works as a minimum probability threshold in which the strategy predicts fraud. This optimization occurs after the new model has been trained and has predicted the fraud probabilities on the retrain data. These probabilities are then used to find the optimal fraud probability. Similar to the function used in the model selection, this optimization function measures the cost savings at each percent threshold of the retrain data then measures the cost savings of that threshold. After iterating over each percentage point from .01 to 1, by increments of .01, the threshold with the highest cost savings is selected to be the new optimal fraud probability threshold. This threshold is then applied to the fraud probability distribution of the model on the new days data to predict if the record will be categorized as fraud or not by the strategies model. This fraud probability threshold is not updated again until the model retrains or the iteration ends.

RESULTS

Overall Results Analysis

We ran the simulation for 20 iterations with 20 days per iteration. After completing the simulation 400 new datasets each 20,000 records were created from the simulation function. During each iteration the 4th 9th and 14th day introduced a new random distribution for the fraud records. During each day of each iteration, we used the Sklearn metrics package to measure the strategies performance metrics. After gathering all the metrics into a single data-frame we used Seaborn and Matplotlib libraries to plot and analyze the results. Depicted below in Table 8, are the mean metrics for each strategy across all days in every iteration. The highlighted and bolded cells represent the largest value of the column.

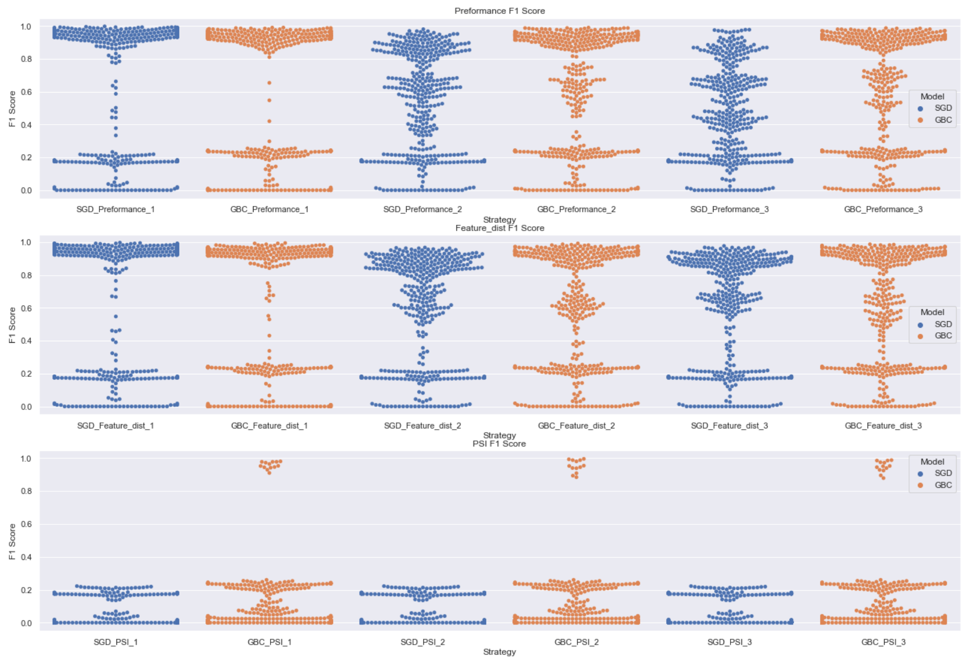
Table 8: Model Results

Model	Detection Method	Retrain Method	F1 Score	Cost-Savings	Precision	Recall
GBC	feature distribution	1	0.664	$4,265	0.717	0.645
GBC	feature distribution	2	0.606	$3,735	0.637	0.611
GBC	feature distribution	3	0.607	$3,725	0.65	0.619
GBC	PSI	1	0.094	$100	0.156	0.074
GBC	PSI	2	0.094	$101	0.156	0.074
GBC	PSI	3	0.094	$101	0.156	0.074
GBC	performance	1	0.657	$4,197	**0.722**	0.633
GBC	performance	2	0.603	$3,720	0.642	0.603
GBC	performance	3	0.603	$3,691	0.648	0.607
SGD	feature distribution	1	0.672	$4,411	0.716	0.67
SGD	feature distribution	2	0.59	$3,640	0.614	0.598
SGD	feature distribution	3	0.594	$3,664	0.621	0.602
SGD	PSI	1	0.042	($318)	0.08	0.029
SGD	PSI	2	0.042	($318)	0.08	0.029
SGD	PSI	3	0.042	($318)	0.08	0.029
SGD	performance	1	**0.673**	**$4,414**	0.716	**0.67**
SGD	performance	2	0.539	$3,161	0.559	0.552
SGD	performance	3	0.47	$2,540	0.496	0.479

Observing the results above led us to conclude that the performance and feature distribution detection methods were superior in all metrics to that of the PSI detection method. Additionally, there seems to be a negative relationship between the number of days used to retrain and the f1 and cost savings metrics. In all combination of model and concept drift detection except PSI the retrain method 1 outperformed all other retain methods. The SGD model in combination with the performance detection method and retain method 1 achieved the highest mean f1 score cost-savings and recall. The GBC model with a performance detection method and retain method 1 achieved the highest mean precision.

While looking at the mean performance can tell us how well on average each strategy did, we can gain more insight by looking into the distribution of performance metrics across all the days of each iteration. The graphic below shows the distributions of the f1 scores for each strategy. Each row of the graphic represents one concept drift detection method. The x axis represents the model and retain method combinations that used that concept drift detection method. The y axis represents the f1 scores achieved. Each dot on the graph represents the f1 score achieved during a day within the simulation for the given model concept drift detection method and retain method.

Figure 5: Performance F1 Scores

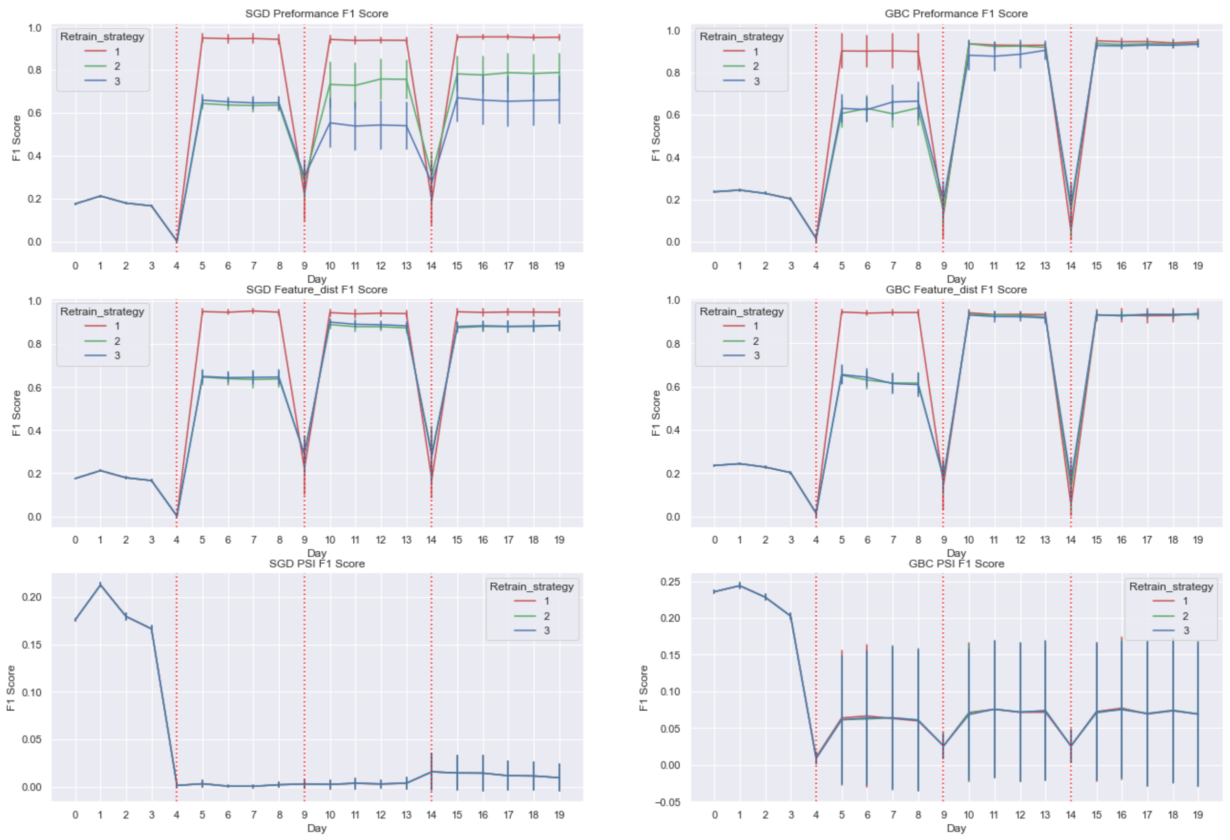


Both the GBC and SGD models seem to have a bi modal distribution of f1 scores for both the performance and feature detection concept drift detection methods. Additionally, the more days the strategy used in training seemed to influence the shape of the f1 metric distributions. The retrain methods 2 & 3 which utilized 3 days in for retaining have a smoother distribution while the retrain method 1 has a stricter bi modal distribution. This pattern is observed across all concept drift detection methods.

Temporal Results Analysis

After analyzing the overall strategy results, we explored the temporal impacts of fraud concept drift on each of the strategies performance over time. To do this we created a time series plot where the x axis represents a day of the iterations, and the y axis represents the average f1 score for that day across all iterations. Because each day is seen 20 times by the strategies error bars were included for each day to observe the variation in results. The graphic's columns below in Figure 6 represent one of the two models, and the rows represent one of the three concept drift detection methods. The colors in each of the plots represent one of the three retaining methods. The doted red line represents the days in which the fraud distributions changed.

Figure 6: Performance F1 Scores over time



As expected, all strategies see a decrease in their f1 metric on days 4, 9, 14 as these were the days that the fraud distributions were randomized. The dramatic increase the following day in f1 score may be due to the strategy detecting the drift and then performing retraining. In general, it appears that the error bars are much larger in the performance concept drift detection method vs. the feature drift detection method. Retain method 1 compared to methods 2 & 3 appears to be achieving a higher f1 score consistently across all models in the performance and feature distribution concept drift detection methods after the first fraud distribution shift on day 4. This gap in performance is sustained until the next day the fraud distribution is changed. Upon the second fraud distribution shift the difference between the f1 metrics for retrain methods 1 to 2 & 3 decrease dramatically. The PSI concept drift detection method's f1 metric steadily decreases until it levels off near 0 indicating that the model may very rarely be re training.

Re training Correlation Results Analysis

Finally, we analyzed the relationship between the total number of times a strategy retrained and the mean f1 score for every iteration. To do this we aggregated the results by the iteration, model, retain method and concept drift detection method. In this aggregated data we measure the mean f1 score and total number of times the strategy retrained. We then plotted these values on a scatter plot. The results are depicted below in Figure 7-9. The x axis represents the total number of times a strategy retained for a given iteration. The y axis represents the mean f1 score for the strategy during that iteration. The columns of the depiction represent one of the two models and the rows represent one of the three concept drift detection methods. The retraining methods are color coded within each of the plots.

Figure 7: Performance detection

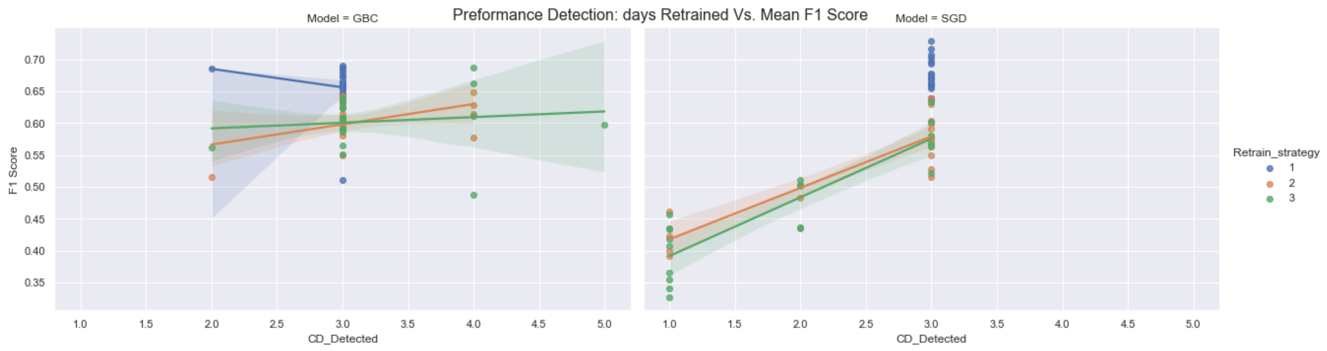


Figure 8: Feature distribution detection

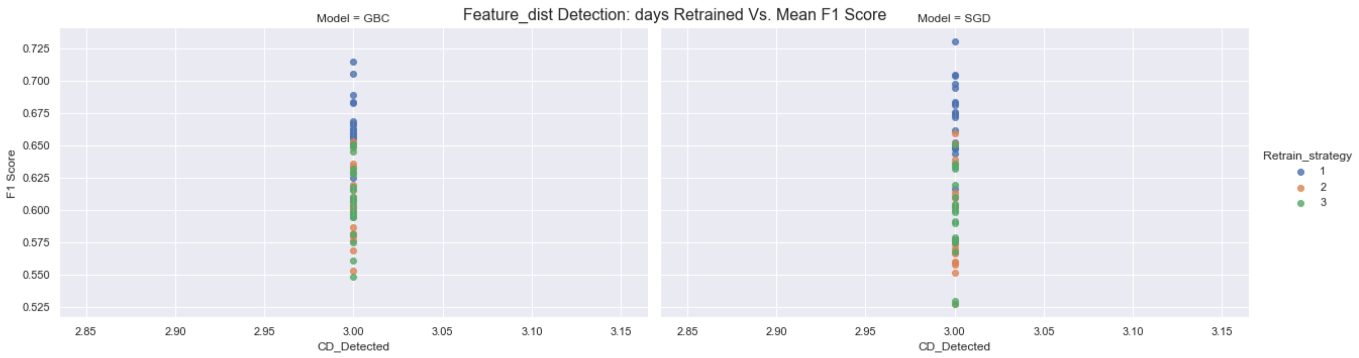
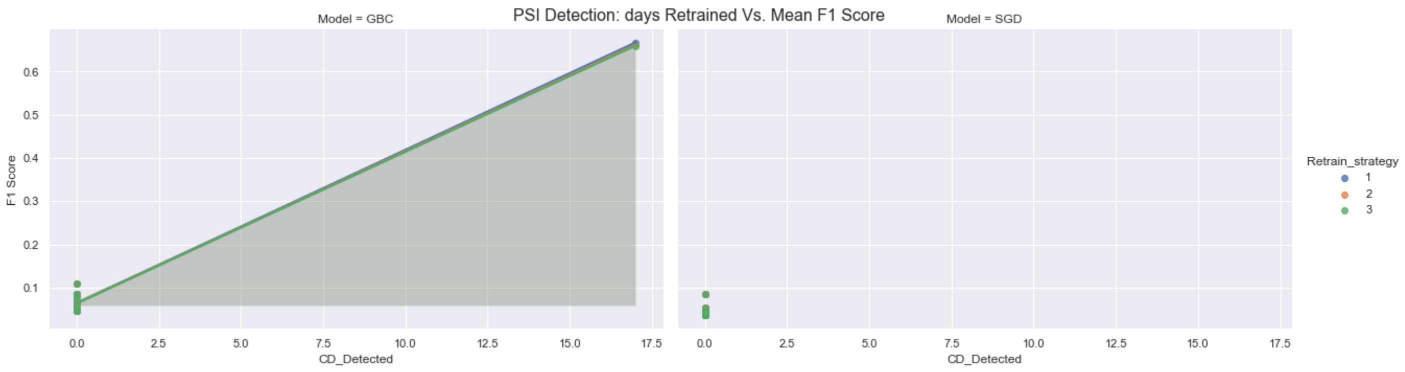


Figure 9: PSI detection



The performance concept drift detection method (row 1 above) seems to show a positive correlation between the times the strategy retained and the mean f1 score it achieved in its iteration. This relationship is strong for the SGD model but much weaker for the GBC model, and even appears to be negatively correlated in retrain method 1 for the GBC model. This negative correlation for the GBC performance detection method may be due to it only retraining when the performance falls below the training performance and thus more re-training means that the strategies performance during the iterations fell below training many times. Thus, poorer performance will always be associated with higher number of retraining for this detection method. All strategies that utilized the feature drift detection method (row 2 above) only retrained three times during an iteration. This may be due to the concept drift method only measuring changes in the fraud distribution which only occurred on the three days were the fraud feature distributions were randomized. Finally, the PSI concept drift detection method (row 3 above) never retained with the SGD model and very rarely retained with the GBC model. The iterations that it did retrain it appears to have retrained almost every day that it could. On the iterations in which the strategy utilizing the PSI detection method did retain it was able to achieve a much higher f1 metric which was comparable to the mean f1 metric of the strategies that utilized the feature distribution and performance concept drift detection methods.

DISCUSSION AND CONCLUSION

The Fraud-Detection Systems (FDS) deployed today are facing many challenges needing to be overcome to effectively identify fraudulent transactions. As part of this study, we have focused on the following problems, concept drift, class imbalance, and high cost of false positives. The sampling methods deployed allowed us to overcome the issue of class imbalanced. Due to computational constraints only random under sampling was utilized during our model retraining. In our study we saw the results of f1 score and our cost function did not deviate much in terms of practical significance. The model that obtained the highest f1 score within our study also obtained the highest cost savings score. This can be said with the inverse that the model that had the lowest f1 score also had the lowest cost savings score. This most likely is due to the input values selected. Future work needs be done on the cost savings function specifically, utilizing different firm operational costs to determine if the cost-savings function then produces models that perform better than optimizing on f1 score.

In addressing concept drift, the concept drift mechanisms and retrain strategies overall appeared to increase models f1 model performance following concept drift days. The f1 scores decreased for all models and strategies when concept drift was introduced, but this expected as the population distributions were altered to simulate a change in fraud actor's behavior. The improvement in f1 scores for all the models indicates that the mechanism was successful in identifying concept drift and model retraining occurred. This mechanism offers firms the prospect to counteract concept drift within their fraud detection systems. More study should be done to see how this mechanism performs with real world data.

AUTHOR CONTRIBUTIONS

Stephen contributed by building the models utilized within the study as well as developing the concept detection drift mechanism and simulation function. John contributed by developing the synthetic data generation and noise introduction function utilized within the study.

REFERENCES

[1]. Wallny, Florian. (2022). False Positives in Credit Card Fraud Detection: Measurement and Mitigation. 10.24251/HICSS.2022.195.

[2]. Bahnsen, A.C., Stojanovic, A., Aouada, D. and Ottersten, B.: "Cost Senitive Credit Card Fraud Detection Using Bayes Minium Risk",12th International Conference on Machine Learning and Appliacations, Miami, USA, 2013

[3]. Bahnsen, A.C., Aouada, D. and Ottersten, B.: "Ensemble of Example-Dependent Cost-Sensitive Decision Trees", *arXiv* 1505.04637, 2015.

[4]. Javelin Research: "Overcoming False Positives: Saving the Sale and the Customer Relationship", Javelin Research Whitepaper, 2015, https://www.javelinstrategy.com/coverage-area/overcoming-false-positives-saving-sale-and-customer-relationship

[5]. *Reducing "False Positive" Determinations in Fraud Detection*,www.taxpayeradvocate.irs.gov/wp-content/uploads/2020/08/ARC16_Volume3_07_ReducingFalsePositive.pdf. Accessed 5 Nov. 2023.

[6]. Fraud Losses and False Positives: The Numbers, SecuredTouch (Dec. 2015)

[7]. "False Positives & Fraud Prevention Tools: J.P. Morgan." *False Positives & Fraud Prevention Tools | J.P. Morgan*, J.P. Morgan, 6 Mar. 2023, www.jpmorgan.com/insights/payments/analytics-and-insights/cnp-fraud-prevention-combat-chargebacks.

[8]. Chan, Philip K.; Prodromis, A.; Fan, W. and Stoflo, S. (1999): "Distributed Data Mining in Credit Card Fraud Detection, IEEE Intelligent Systems" 14 (6), 1999, pp. 67-74.

[9]. Nishida, K., Yamauchi, K. (2007). Detecting Concept Drift Using Statistical Testing. In: Corruble, V., Takeda, M., Suzuki, E. (eds) Discovery Science. DS 2007. Lecture Notes in Computer Science(), vol 4755. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-75488-6_27

[10]. A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784-3797, Aug. 2018, doi: 10.1109/TNNLS.2017.2736643.

[11]. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(4), 463–484. https://doi.org/10.1109/TSMCC.2011.2161285

[12]. Singh, Amit, et al. "Credit card fraud detection under extreme imbalanced data: A Comparative Study of data-level algorithms." *Journal of Experimental &amp; Theoretical Artificial Intelligence*, vol. 34, no. 4, 2021, pp. 571–598, https://doi.org/10.1080/0952813x.2021.1907795.

[13]. Ling, C. X., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. *IN Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98* pp. 73–79.

[14]. Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced data sets*. Springer International Publishing. https://doi.org/10.1007/978-3-319-98074-4

[15]. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*(1), 321–357. https://doi.org/10.1613/jair.953

[16]. Ritchie, John Newman & Amy, and Simon Fondrie-Teitler and Amritha Jayanti. "New FTC Data Show Consumers Reported Losing Nearly $8.8 Billion to Scams in 2022." *Federal Trade Commission*, 23 Feb. 2023, www.ftc.gov/news-events/news/press-releases/2023/02/new-ftc-data-show-consumers-reported-losing-nearly-88-billion-scams-2022.

[17]. "Fraud Detection and Prevention Market Growth Report [2030]." *Fraud Detection and Prevention Market Growth Report [2030]*, www.fortunebusinessinsights.com/industry-reports/fraud-detection-and-prevention-market-100231. Accessed 6 Sept. 2023.

[18]. Moroke, Ntebogang Dinah, and Katleho Makatjane. "Predictive Modelling for Financial Fraud Detection Using Data Analytics: A Gradient-Boosting Decision Tree." *Applications of Machine Learning and Deep Learning for Privacy and Cybersecurity,* edited by Victor Lobo and Anacleto Correia, IGI Global, 2022, pp. 25-45. https://doi-org.ezproxy.depaul.edu/10.4018/978-1-7998-9430-8.ch002

[19]. Peilin Li. Credit Card Fraud Detection Based on Random Forest Model. Academic Journal of Computing & Information Science (2022), Vol. 5, Issue 13: 55-61. https://doi.org/10.25236/AJCIS.2022.051309.

[20]. J. O. Awoyemi, A. O. Adetunmbi and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," *2017 International Conference on Computing Networking and Informatics (ICCNI)*, Lagos, Nigeria, 2017, pp. 1-9, doi: 10.1109/ICCNI.2017.8123782.

[21]. Y. Sahin and E. Duman, "Detecting credit card fraud by ANN and logistic regression," *2011 International Symposium on Innovations in Intelligent Systems and Applications*, Istanbul, Turkey, 2011, pp. 315-319, doi: 10.1109/INISTA.2011.5946108.

[22]. Khaled Gubran Al-Hashedi, Pritheega Magalingam, Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019, Computer Science Review,Volume 40, 2021,100402, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2021.100402.

APPENDIX

Table 9: Dataset variables

Dataset variable	Variable Type	Description
income	numeric	Annual income of the applicant
name_email_similarity	numeric	Metric of similarity between email and applicant's name.
prev_address_months_count	numeric	Number of months in previous registered address of the applicant. -1 indicates missing value.
current_address_months_count	numeric	Months in currently registered address of the applicant. -1 indicates missing value.
customer_age	numeric	Applicants age in years, rounded to the decade.
days_since_request	numeric	Number of days passed since application was done.
Intended_balcon_amount	numeric	Initial transferred amount for application. Negative values indicade missing record.
payment_type	categorical	Credit payment plan type.
zip_count_4w	numeric	Number of applications within same zip code in last 4 weeks.
velocity_6h	numeric	Velocity of total applications made in the last 6 hours.
velocity_24h	numeric	Velocity of total applications moade in the last 24 hours.
velocity_4w	numeric	Velocity of total applications made in the last 4 weeks.
bank_branch_count_8w	numeric	Number of total applications in the selected bank branch in the last 8 weeks.
date_of_birth_distinct_emails_4w	numeric	Number of emails for applicants with same date of birth in last 4 weeks.
employment_status	categorical	Employment status of the applicant.
credit_risk_score	numeric	Internal score of application risk
email_is_free	binary	Domain of application email
housing_status	categorical	Current resident status fo applicant
phone_home_valid	binary	Validity of provided home phone.
phone_mobile_valid	binary	Validity of provided mobile phone.
bank_months_count	numeric	How old is previous account (if held) in months. -1 indicates missing value
has_other_cards	binary	If applicant has other cards from the same banking company.
proposed_credit_limit	numeric	Applicant's proposed credit limit.
foreign_request	binary	If origin country of request is different bank's country
source	categorical	Online source of application.
session_length_in_minutes	numeric	Length of user session in banking website minutes. -1 indicates missing value.
device_os	categorical	Operative system of device that made request.
keep_alive_sesion	binary	User option on session logout
device_distinct_emails	numeric	Number of distinct emails in banking webseit from the used device in last 8 weeks. -1 indicates missing value.
device_fraud_count	numeric	Number of fradulent applications with used device.
month	numeric	Month where the application was made
fraud_bool	binary	Indicates if the application is fraudulent or not.

Figure 10: Income fraud vs non-fraud distribution

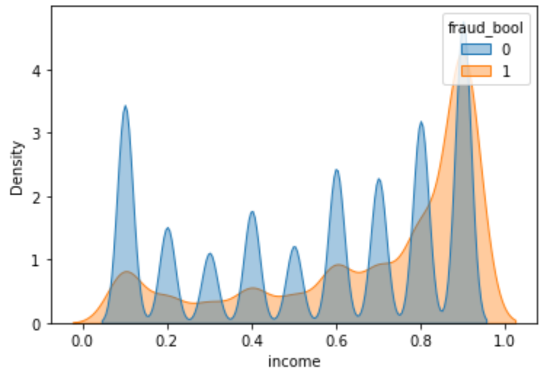


Figure 11: Name email similarity fraud vs non-fraud distribution

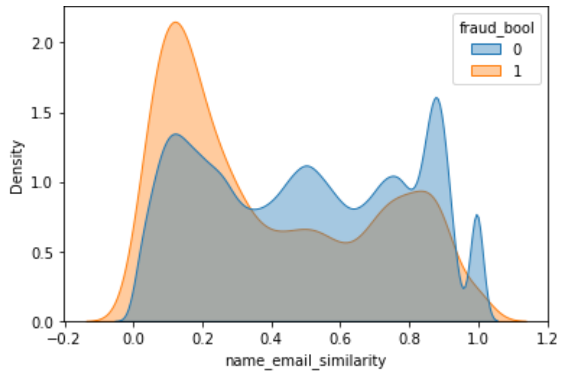


Figure 12: Previous address months count fraud vs non-fraud distribution

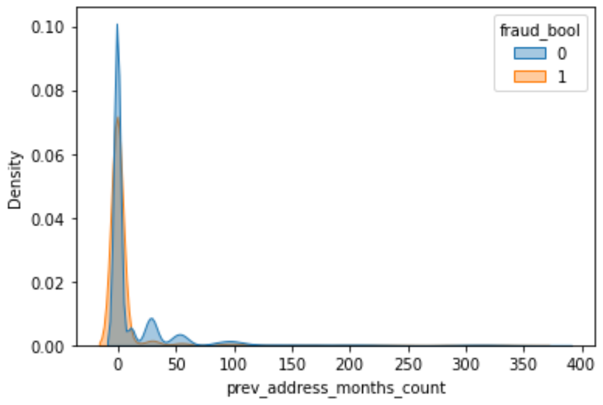


Figure 13: Current address months count fraud vs non-fraud distribution

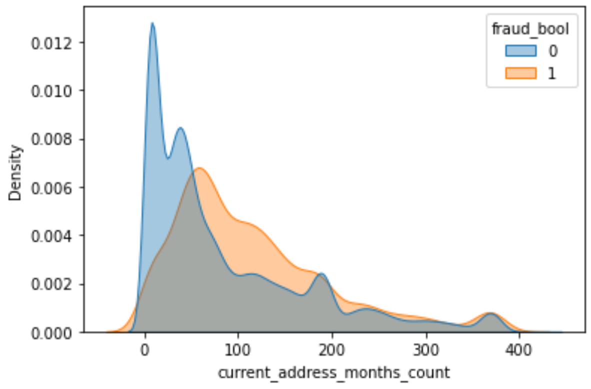


Figure 14: Days since request fraud vs non-fraud distribution

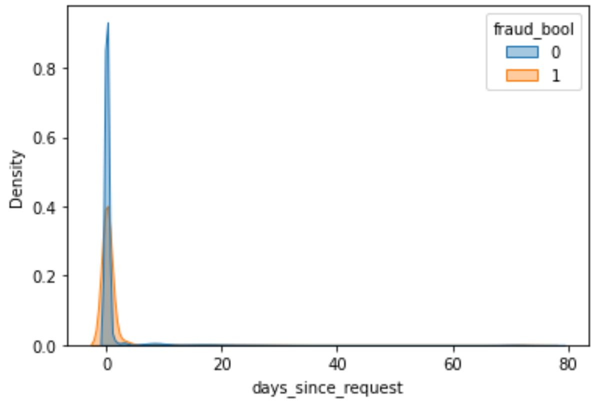


Figure 15: Customer age fraud vs non-fraud distribution

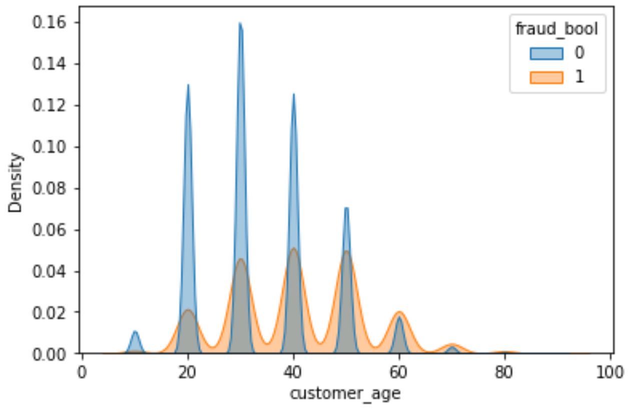


Figure 16: Intended balance amount fraud vs non-fraud distribution

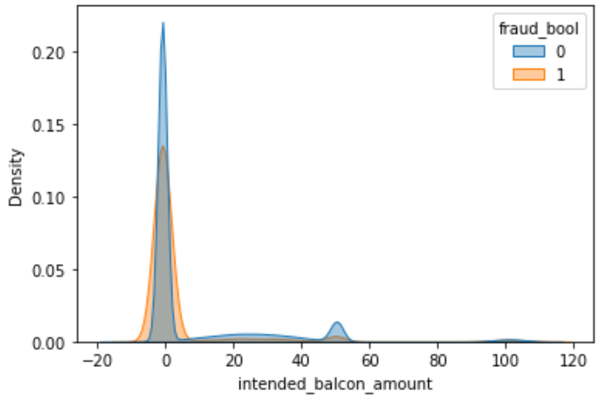


Figure 17: Velocity 6-hour fraud vs non-fraud distribution

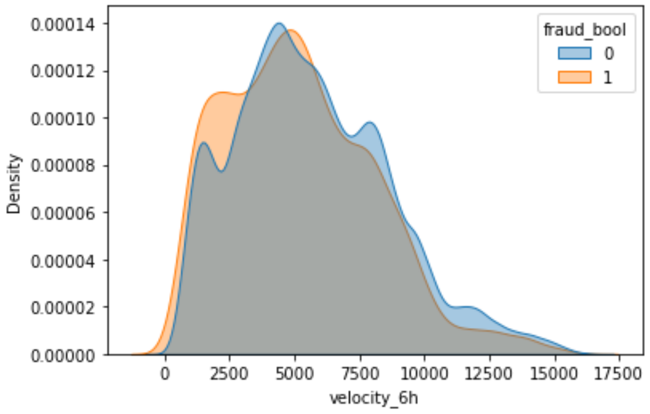


Figure 18: Zip count 4-week fraud vs non-fraud distribution

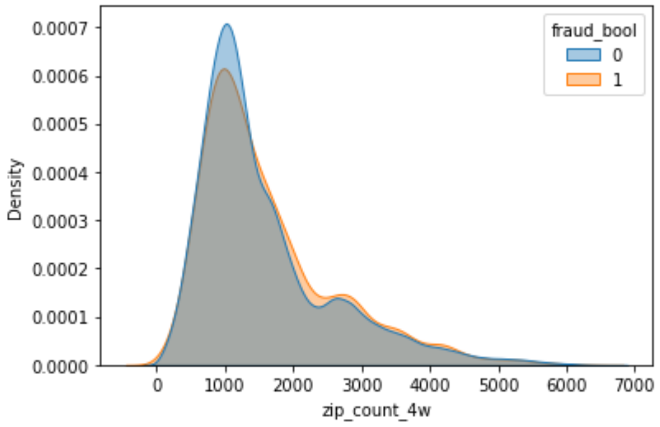


Figure 19: Velocity 24-hour fraud vs non-fraud distribution

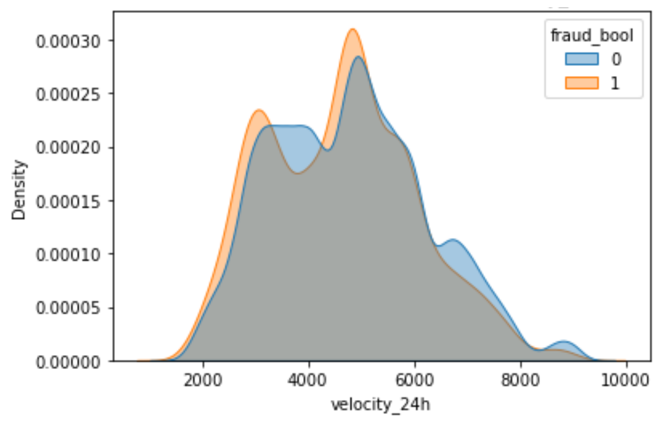


Figure 20: Bank branch count 8-week fraud vs non-fraud distribution

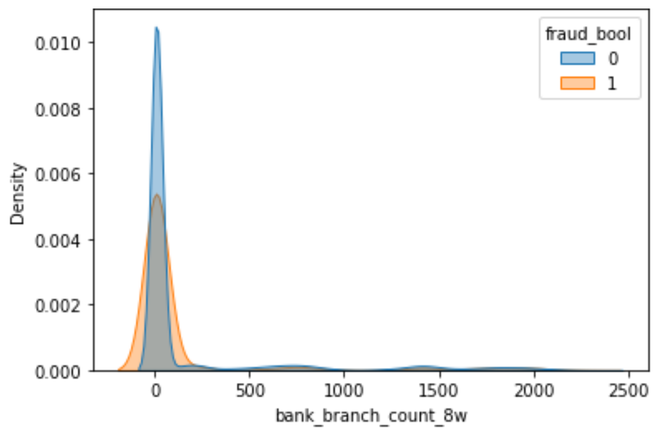


Figure 21: Date of birth distinct emails 4-week fraud vs non-fraud distribution

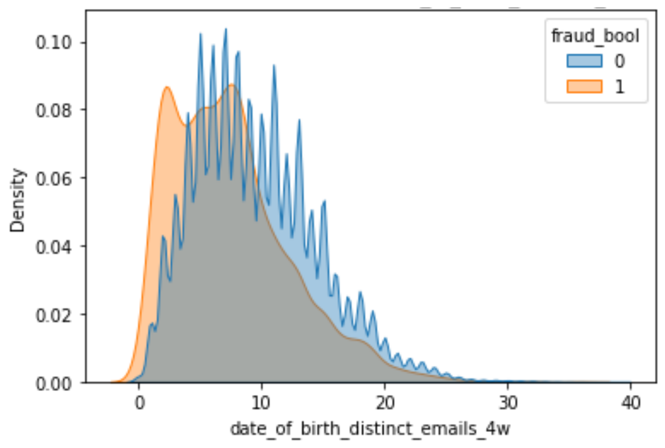


Figure 22: Bank months count fraud vs non-fraud distribution

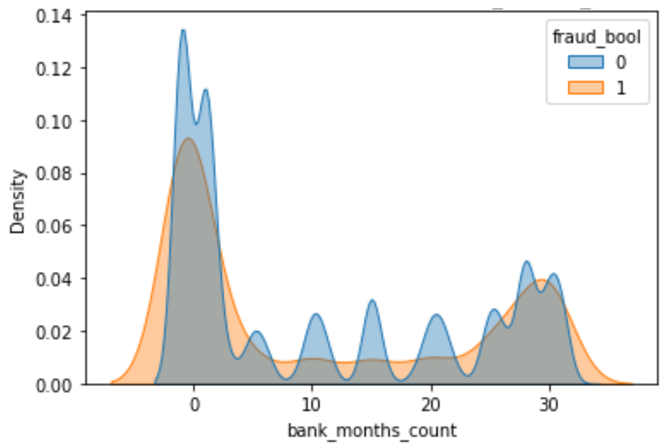


Figure 23: Proposed credit limit fraud vs non-fraud distribution

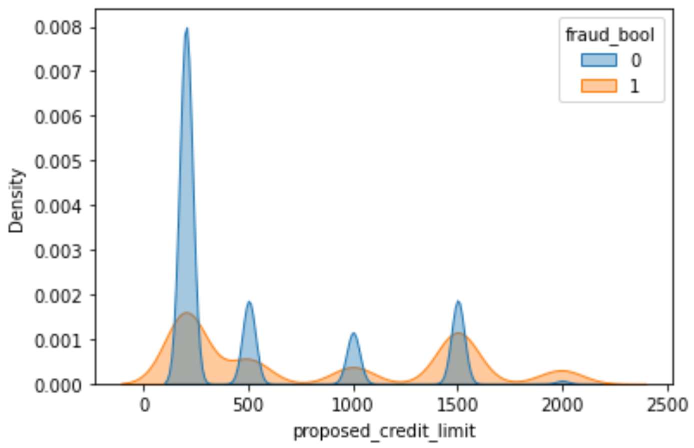


Figure 24: Session length in minutes fraud vs non-fraud distribution

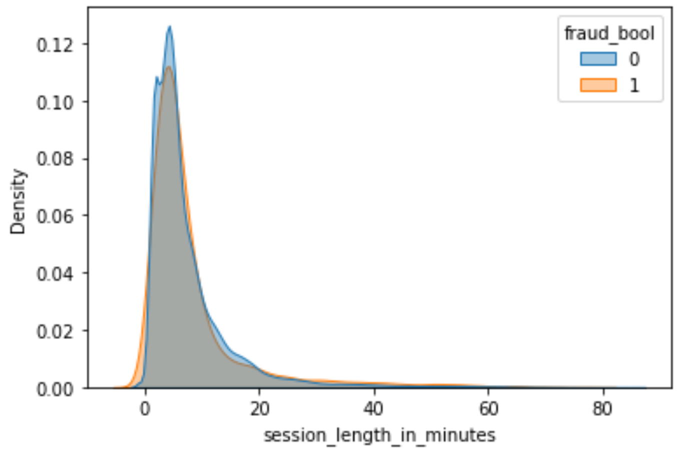


Figure 25: Velocity 4-week fraud vs non-fraud distribution

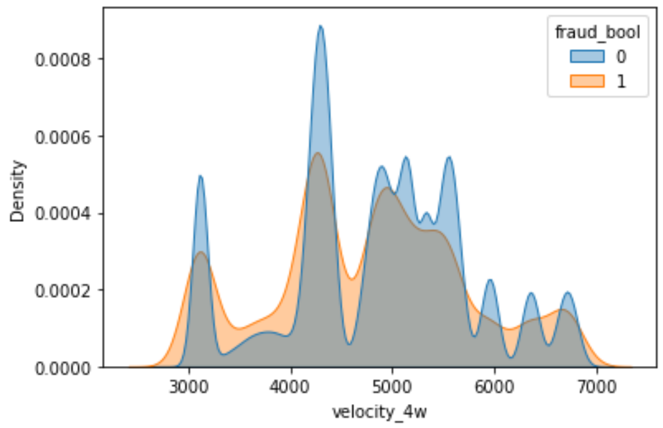


Figure 26: Credit risk score fraud vs non-fraud distribution

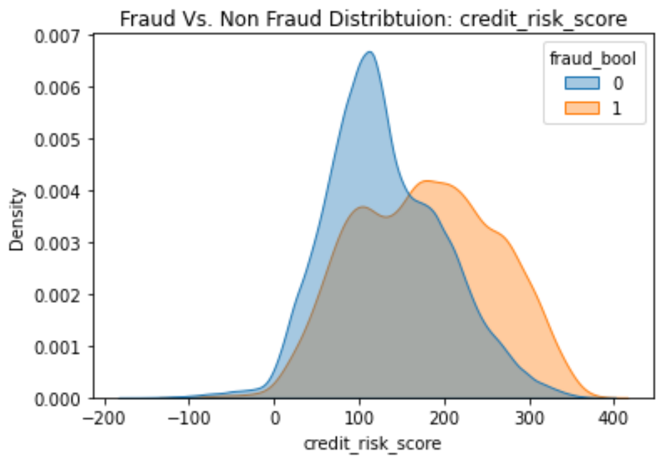


Figure 27: Payment type fraud vs non-fraud distribution

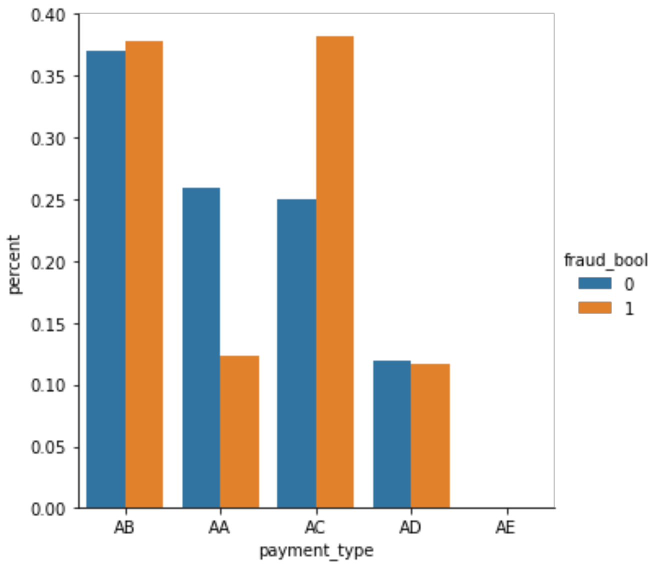


Figure 28: Employment status fraud vs non-fraud distribution

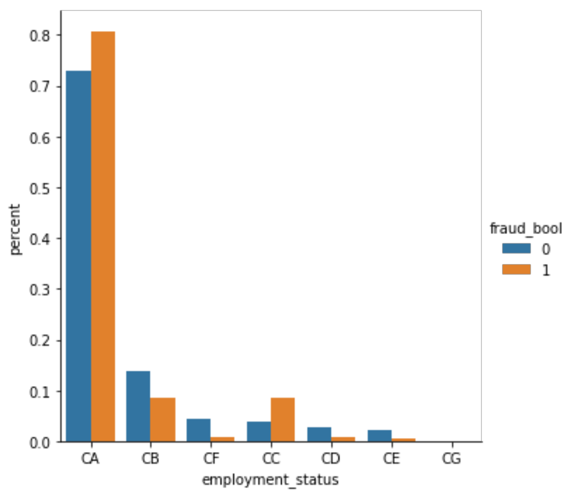


Figure 29: Housing status fraud vs non-fraud distribution

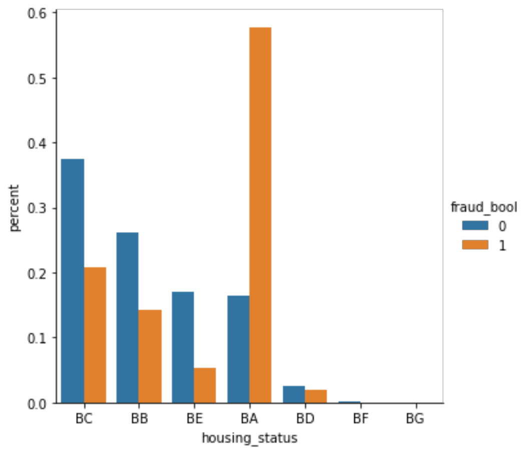


Figure 30: Email is free fraud vs non-fraud distribution

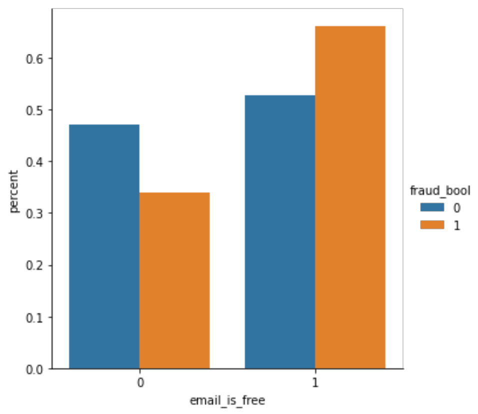


Figure 31: Phone home valid fraud vs non-fraud distribution

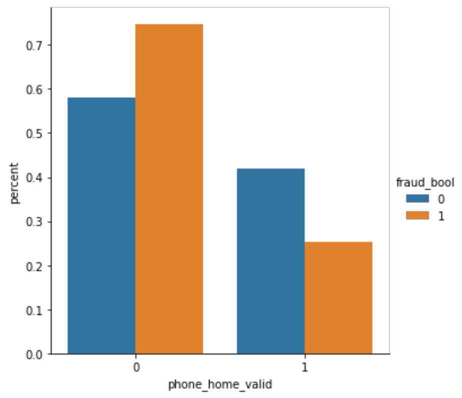


Figure 32: Phone mobile valid fraud vs non-fraud distribution

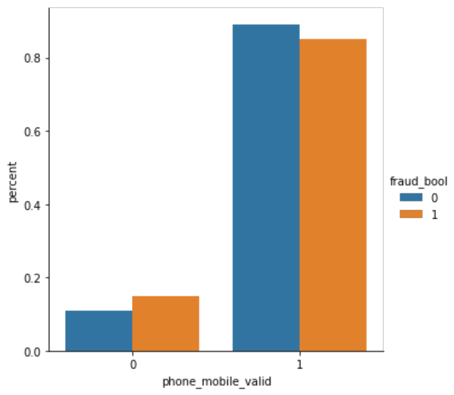


Figure 33: Has other cards fraud vs non-fraud distribution

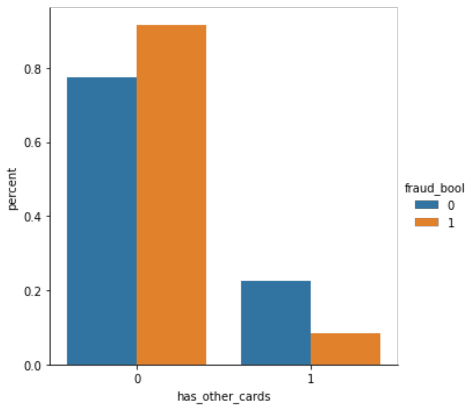


Figure 34: Foreign request fraud vs non-fraud distribution

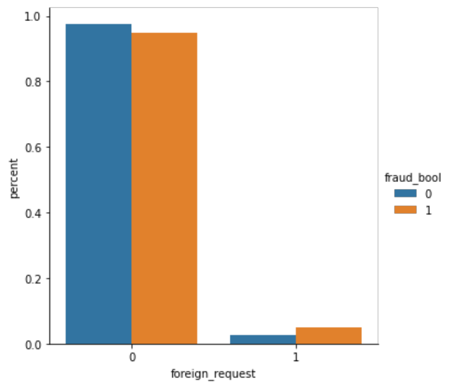


Figure 35: Source fraud vs non-fraud distribution

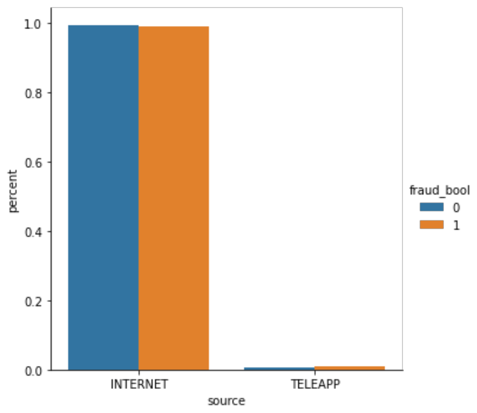


Figure 36: Device operating system vs non-fraud distribution

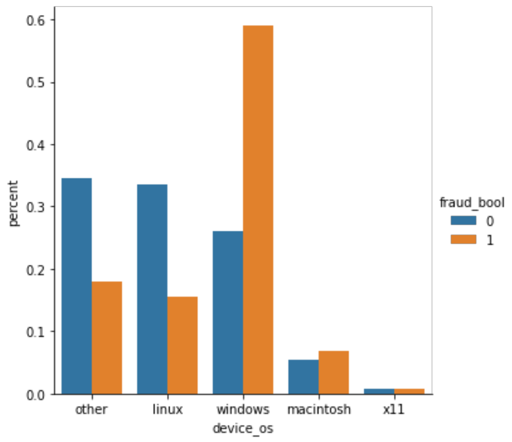


Figure 37: Keep session alive system fraud vs non-fraud distribution

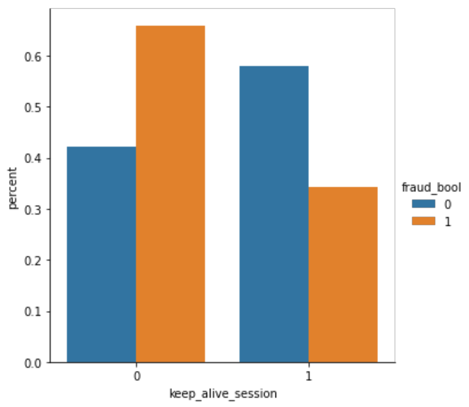


Figure 38: Device distinct email 8-week fraud vs non-fraud distribution

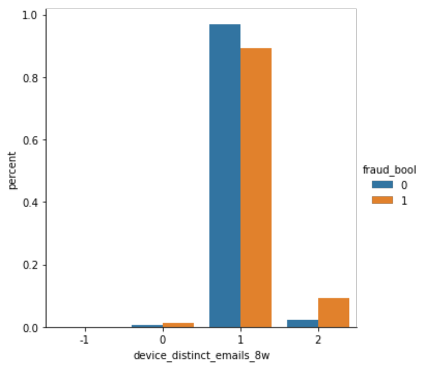


Figure 39: Month fraud vs non-fraud distribution

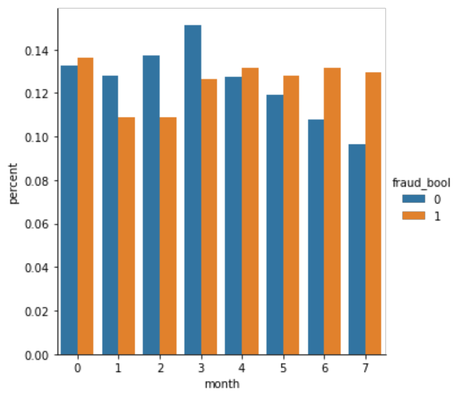


Figure 40: Random Forest optimal cost savings per feature amount

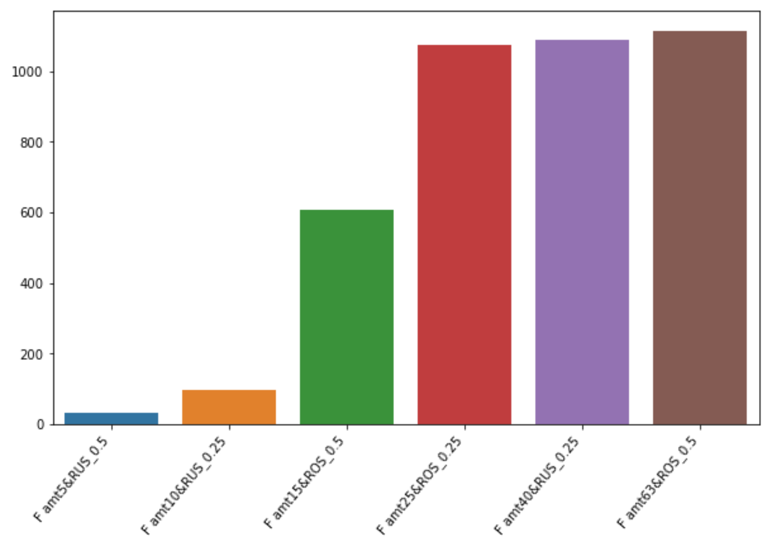


Figure 41: Random Forest optimal F1 score per feature amount

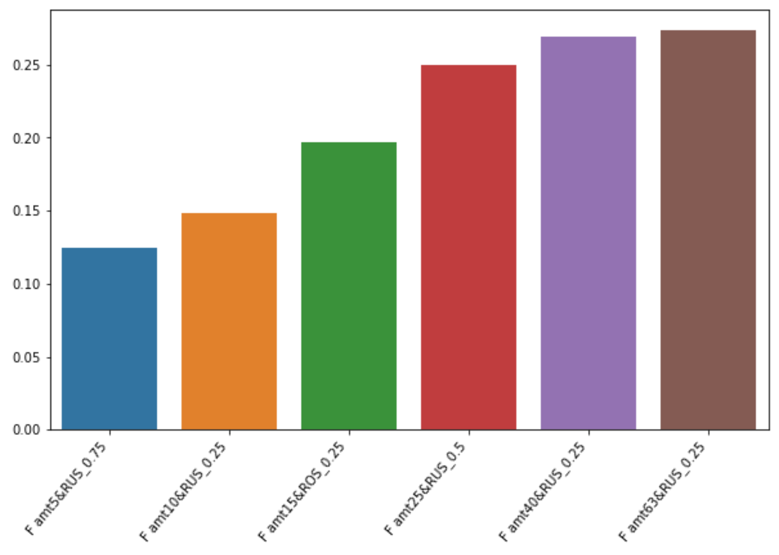


Figure 42: Random Forest optimal recall per feature amount

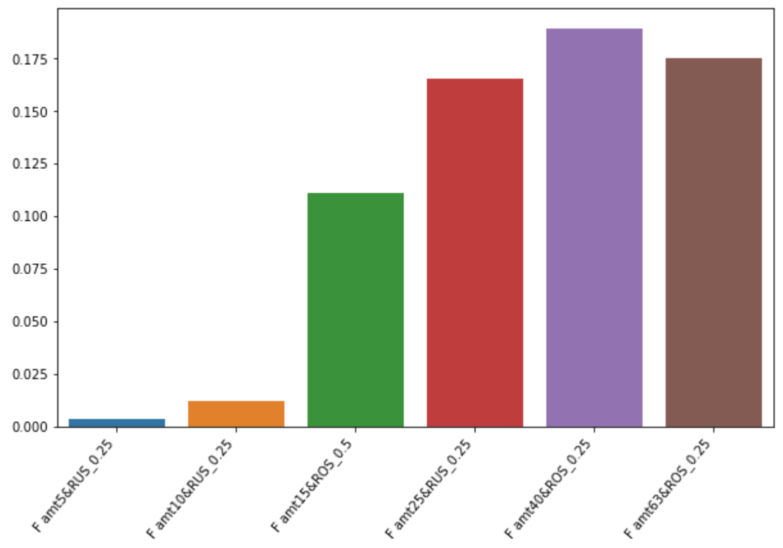


Figure 43: K Nearest Neighbor optimal cost savings per feature amount

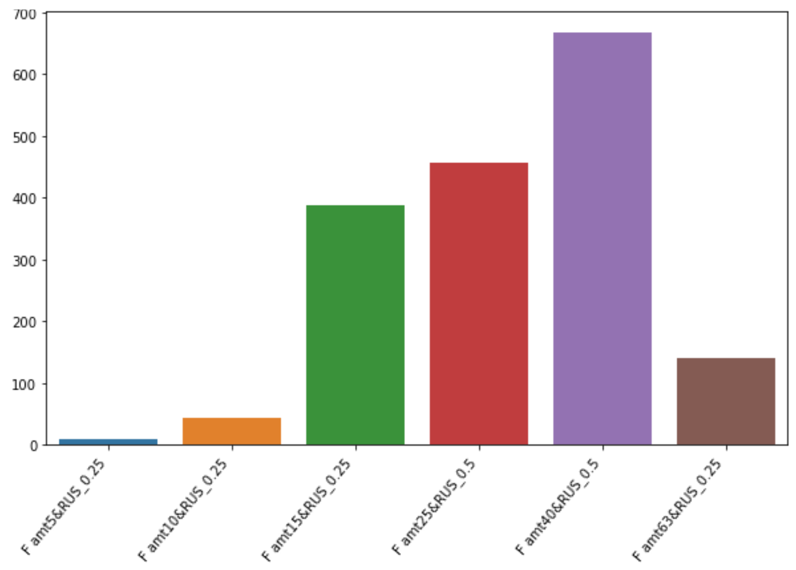


Figure 44: K Nearest Neighbors optimal f1 score per feature amount

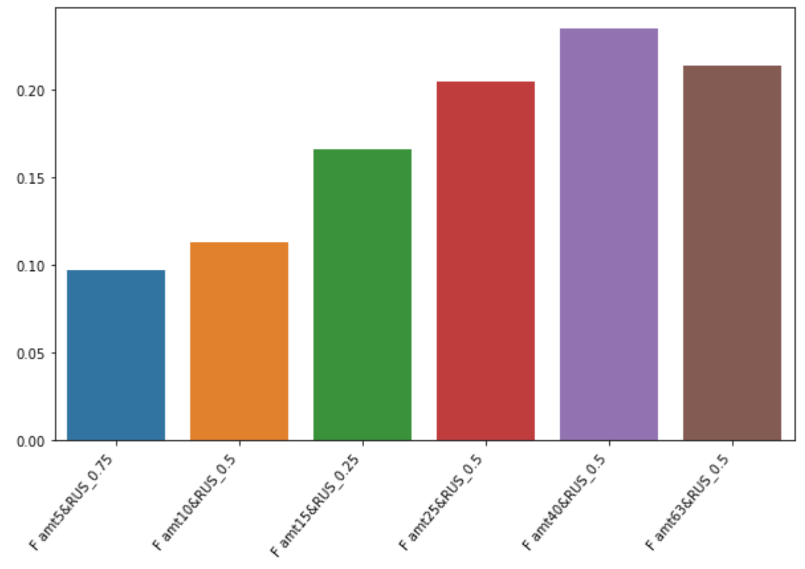


Figure 45: K Nearest Neighbors optimal recall per feature amount

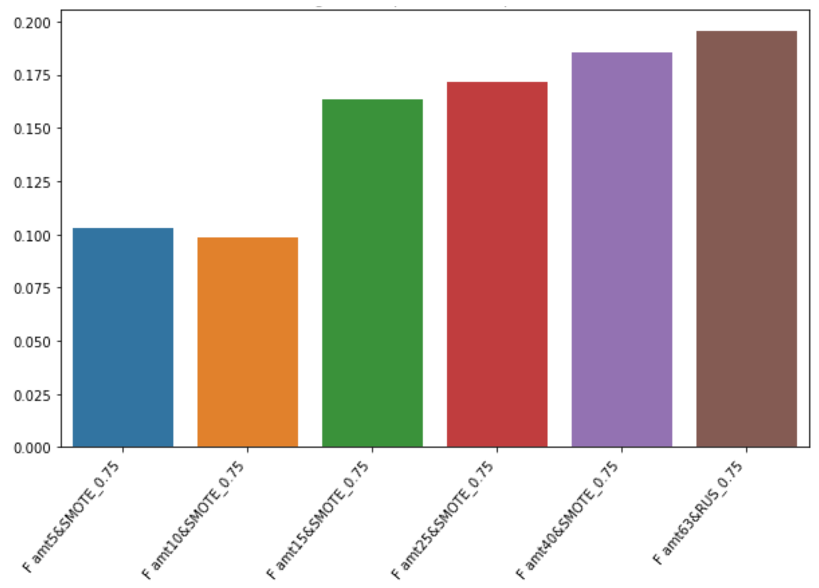


Figure 46: Multi-Layer Perceptron optimal cost savings per feature amount

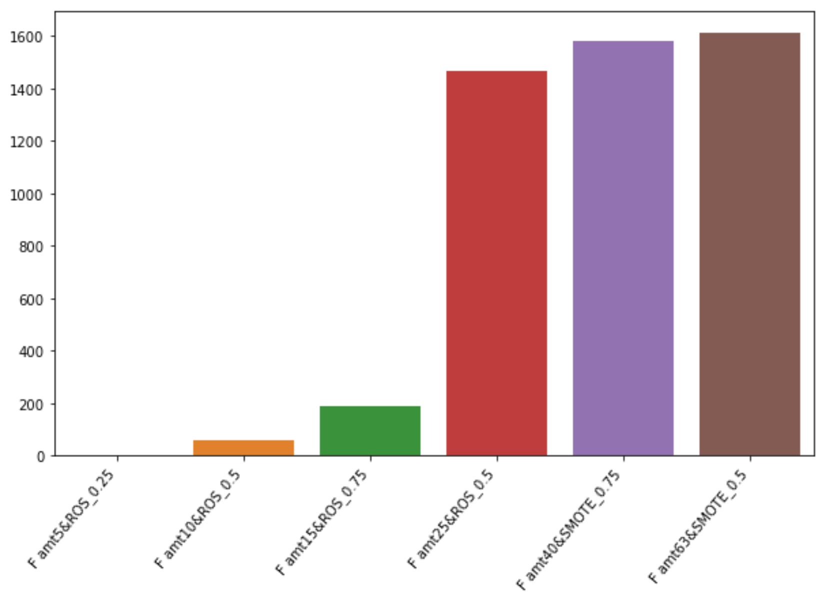


Figure 47: Multi-Layer Perceptron optimal f1 score per feature

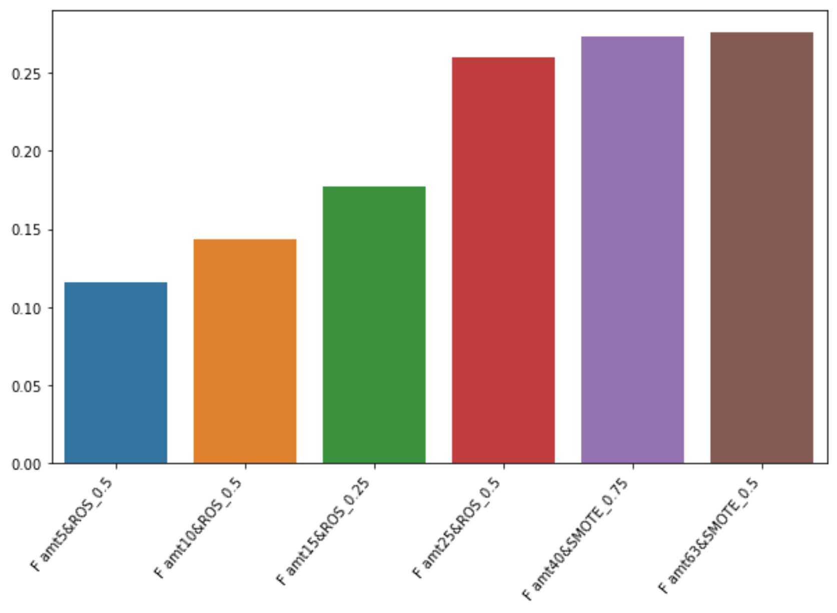


Figure 48: Multi-Layer Perceptron optimal recall per feature

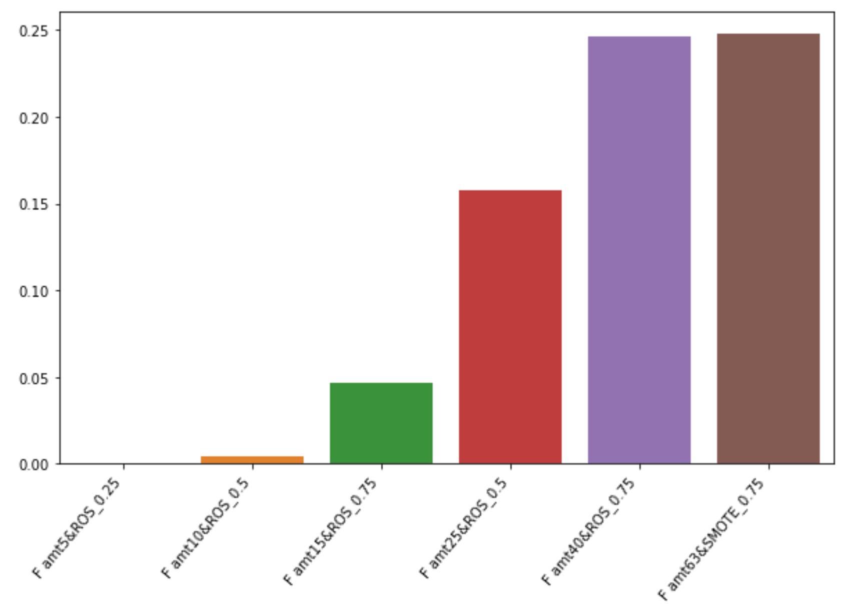


Figure 49: Stochastic Gradient Descent optimal cost savings per feature

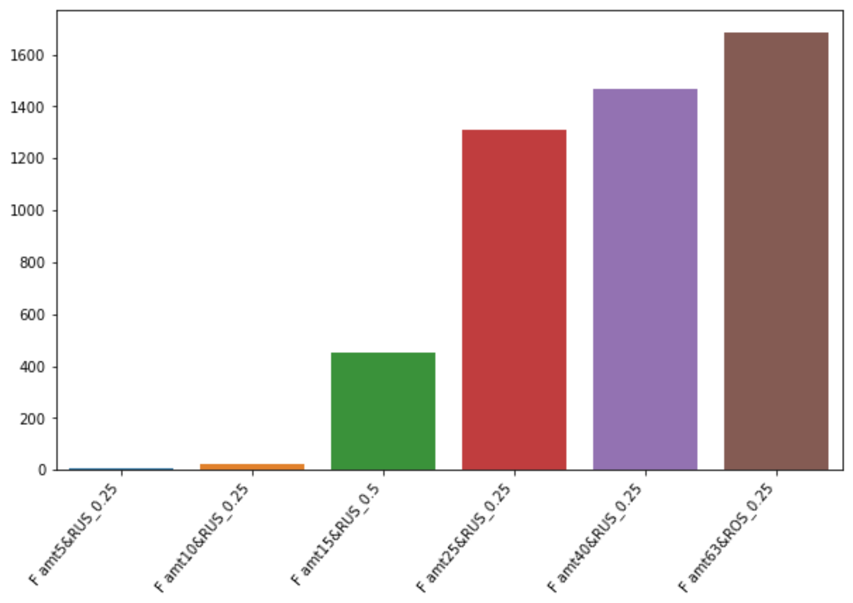


Figure 50: Stochastic Gradient Descent optimal f1 score per feature

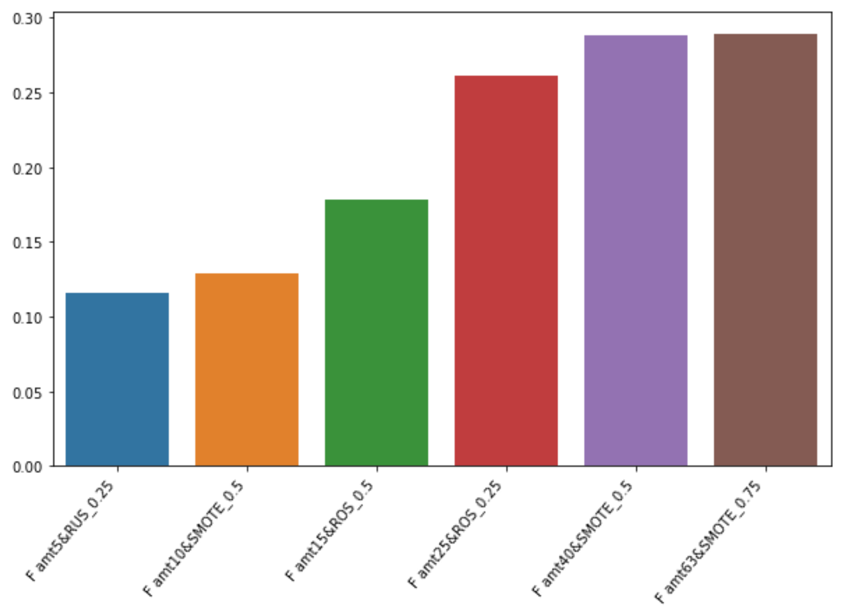


Figure 51: Stochastic Gradient Descent optimal recall score per feature

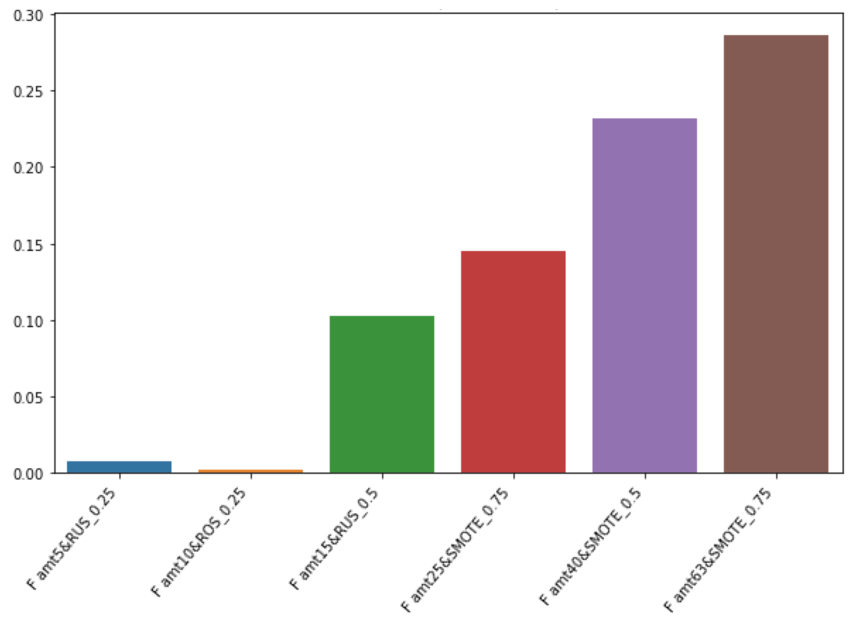


Figure 52: Gradient Boosting optimal cost savings per feature

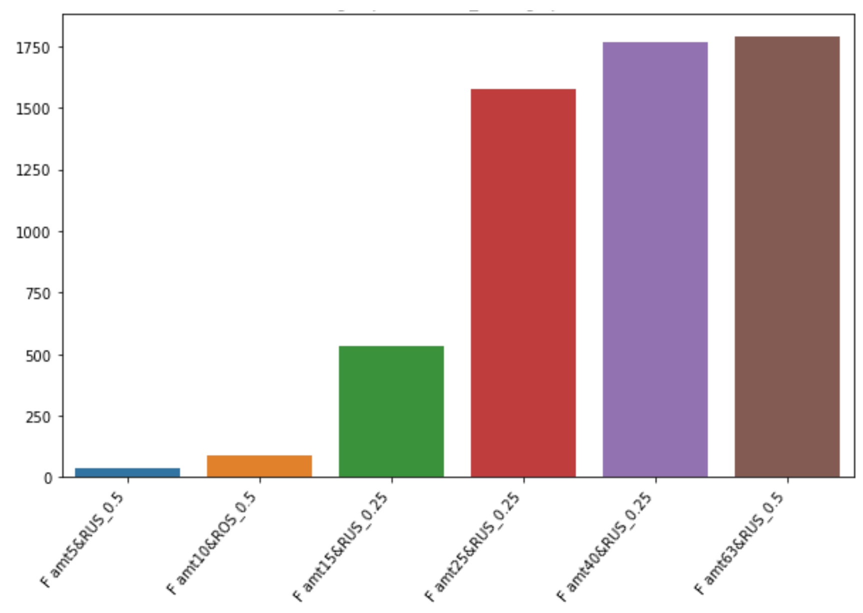


Figure 53: Gradient Boosting optimal f1 score per feature

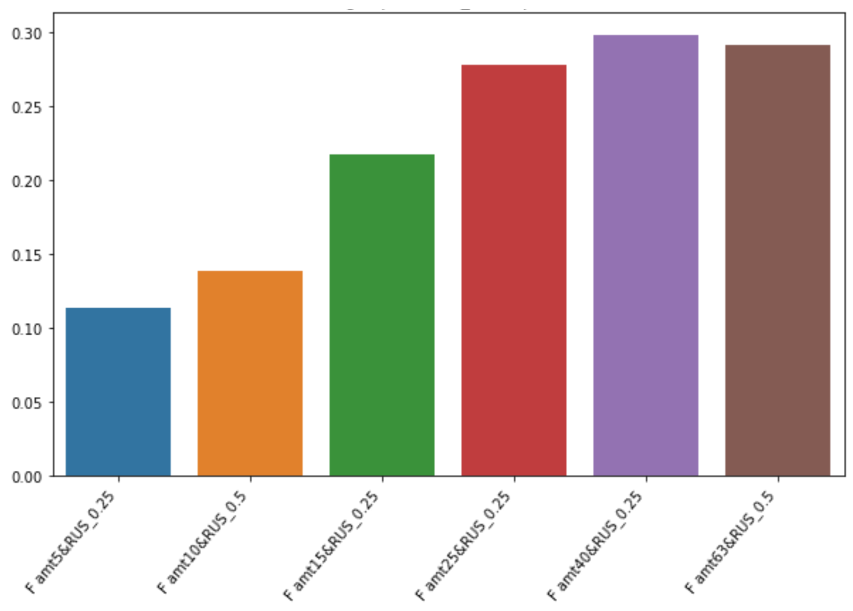


Figure 54: Gradient Boosting optimal recall per feature

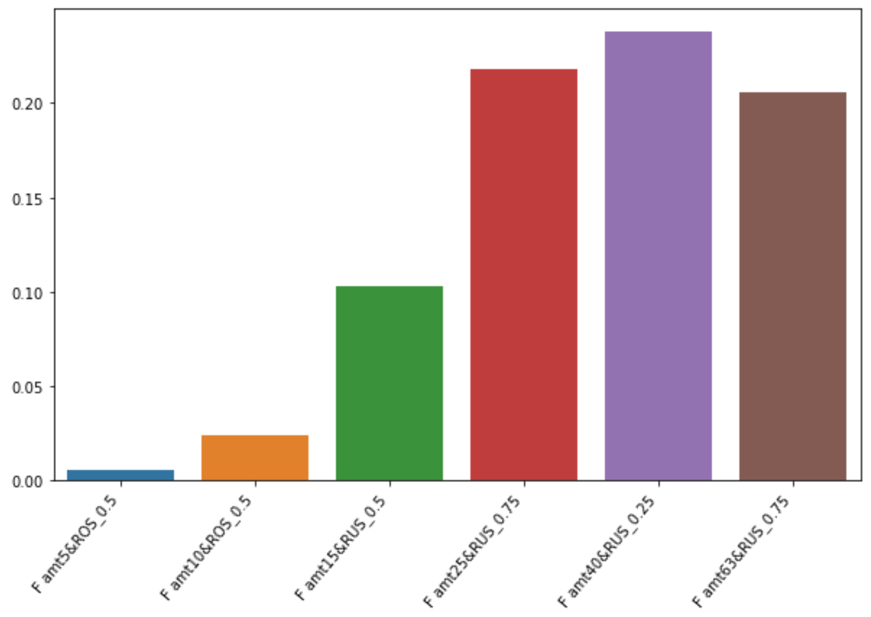


Figure 55: AdaBoost Classifier optimal cost savings per feature

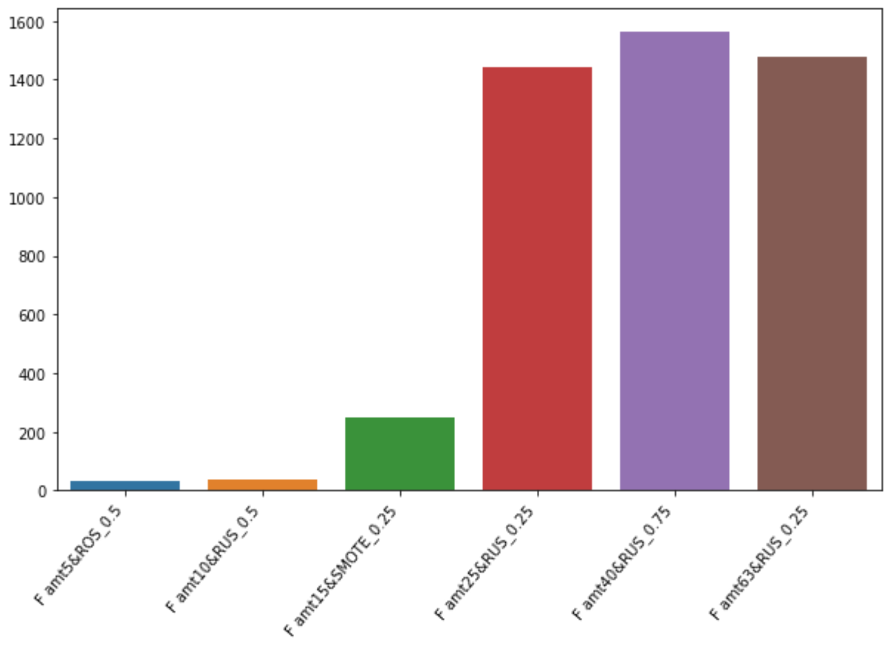


Figure 56: AdaBoost Classifier optimal f1 score per feature

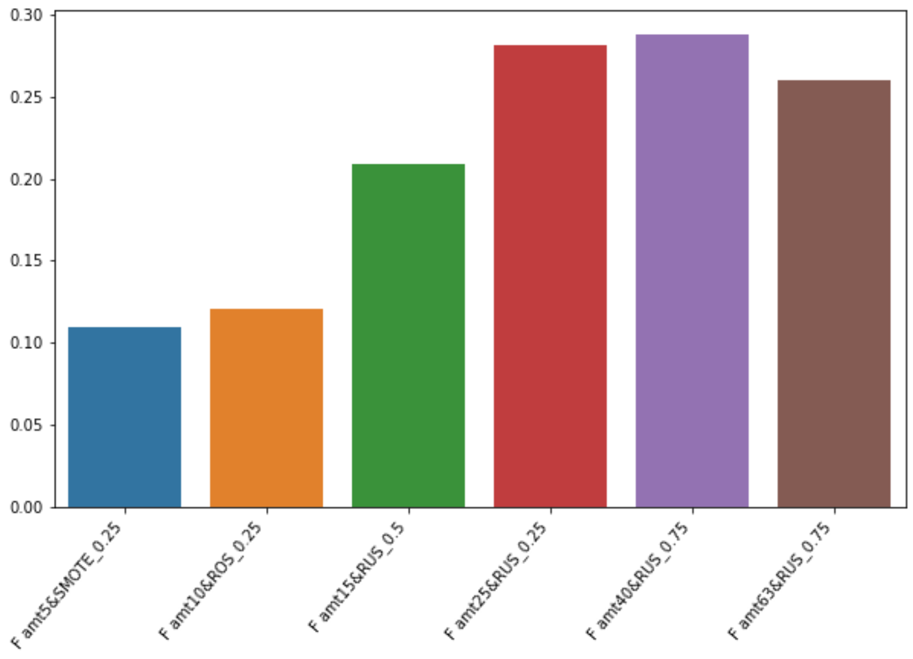
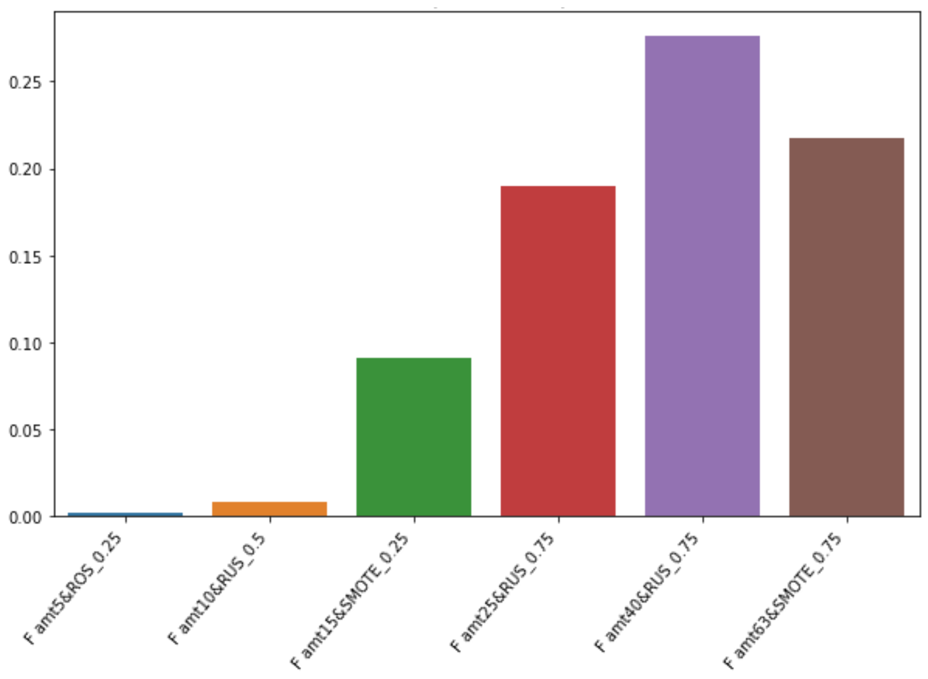


Figure 57: AdaBoost Classifier optimal recall per feature


F amt5&ROS_0.25
F amt10&RUS_0.5
F amt15&SMOTE_0.25
F amt25&RUS_0.75
F amt40&RUS_0.75
F amt63&SMOTE_0.75