Unmasking Fraud: Applying Machine Learning to Detect Bank Drops

James Sohigian

March 15, 2024

1. Abstract

Bank drops are fraudulent bank accounts that are used to funnel and transfer illegal funds. Most fraudulent bank accounts often go undetected, exposing flaws in the banking system and require large teams of cybersecurity specialists to spend significant amounts of time and money trying to correct. This paper evaluates eight machine learning (ML) models that are capable of flagging bank accounts as fraudulent at the time of bank account registration. Each of these models process bank account registration information to instantly determine the legitimate versus fraudulent status of an account, expediting the bank account flagging process, saving banks internal resources and protecting bank clients from fraudulent activities. The highest performing models are ensemble models such as the gradient boosting classifier and the random forest classifier, each having accuracy scores above 90%. These results show that ML models are a valuable tool to detect fraudulent accounts and protect banks across the world, effectively and efficiently stopping crime before it begins.

2. Introduction

Bank drops are a central component of crime. In most cases of terrorism, drug trafficking, gang financial activity, and other forms of criminal activity, money is laundered through the international banking system through the use of bank drops. Bank drops, a term used to describe fraudulent bank accounts, are not only directly related to crime, but also incredibly difficult to detect. In order to detect fraud, teams of cybersecurity professionals are tasked with sorting through vast layers of internal bank data to find the bank drops. As a result, significant resources such as time and money are invested in the painstaking process of detecting bank drops. Despite the cybersecurity investments, dismal results prove that there must be a better way to deal with the bank drops that perpetuate bank fraud.

As artificial intelligence (AI) becomes more reliable and widespread, the financial world continues to integrate AI into their fraud detection processes, implementing many applications of AI systems to increase productivity. While AI is used by cybersecurity firms to enhance banking processes, there is a greater need to leverage AI for societal good in the banking industry.

This paper explores the efficacy of eight different machine learning (ML) models to detect bank drops. The models discussed in this paper can be integrated into the financial industry to identify fraudsters, stop fraudulent transactions like money laundering, and protect individuals from the theft that bank drops often exploit. With the use of ML models, it will take less time and effort than ever before to halt bank drops' illegal practices.

These ML models use supervised learning to classify bank accounts as legitimate accounts (labeled as 0) or fraudulent accounts (labeled as 1) at the point of bank account creation by using bank account registration information. While the point of bank account registration is only a snapshot in time, patterns in bank account registration information are high indicators of whether the bank account will be used for legitimate or fraudulent transactions. At the time of bank account registration, the account has no activity so it has not yet been used for legitimate or fraudulent purposes, but the accounts should be flagged as suspicious and monitored for possible fraud when the models label an account fraudulent. With banks' access to account specific transaction level data in real time, these models can be paired with other ML models that evaluate transactional activity over time and confirm whether or not the account was established for fraudulent purposes. The models in this paper are trained on 31 features: 26 of which are numerical data and 5 of which are categorical data converted to numerical labels using a label encoder that is later elaborated on in this research. Then to determine the models' accuracies, they are tested with 30 features (all features except the legitimate/fraudulent feature) and asked to predict if the account is legitimate or fraudulent. The prediction is then compared to the actual legitimate or fraudulent classification.

With only two label outputs of legitimate or fraudulent, these models serve as a detection tool for banks. Once an account is registered, banks can compile the registration information, plug it into the trained models, and see if the account is highly likely to be legitimate or fraudulent. Since ML models lack 100% accuracy, there is limitation to the process, but the models are the first layer of protection that then prompts the banks to further monitor or investigate the potentially fraudulent accounts.

3. Background

In the professional and academic fields applying machine learning to bank fraud, most of the previous research that implements AI has applied machine learning models to credit card fraud. Credit card fraud is only one type of financial fraud and does not relate to the problem of bank drops. As a result, ML is still newly applied to bank account fraud. While many cybersecurity firms are using AI in the workplace, research and development of next generation models are needed.

As it is geared toward credit card fraud, ML models such as logistic regression, decision tree, and random forest algorithms are proven successful in credit card fraud detection [1]. Even more noteworthy are these models' ability to stay accurate with imbalanced data sets [1], such as the one presented in this research.

Fine tuning of these models and the comparative study of model performances can make the difference to proper model selection focused on protecting people from fraud [2]. By quickly detecting fraud, potential risk is controlled by minimizing losses, even in the current state of globalization where a transaction can be completed from time zones separated hours away from each other [2].

While not every single issue relating to bank account fraud can be adequately addressed, an abundance of research has indicated that comparing ML models' ability to predict a certain aspect of the problem is a step in the right direction to stopping financial crime [3]. Creating the most useful models through hybridization, hyper parameter tuning, and transfer learning is the first step to leveraging the models for societal change [3].

In the case of bank drops, bank accounts can be applied for and registered online. In no time, fraudsters can wire illegal funds around the world without banks observing any indication of money laundering occurrences.

Therefore, there is certainly a need for bank account registration information to be checked by the proper ML models detecting potential fraud before it begins. Using a comparative approach, this paper explores the models with the highest indicators of future success in detecting bank account fraud in the financial sector and further discusses the models' potential role in stopping bank drops all together.

As shown by existing research in this literature review, it is expected that existing models will produce similarly positive results when used to detect bank drops. By using a large data set that rivals the data sets used in the literature review, this research plans to provide a greater population of banks with applicable models. The need for models in this field is necessary, and this research lays forth an outline for which models can do the best job.

4. Dataset

Bank Account Fraud Dataset Suite [10] is taken from kaggle data [11], a public data collection hub of various datasets for research purposes. While the data set has 6 variants, our research is conducted with the base data. The base dataset's description is "Sampled to best represent the original dataset" [10] while all of the variants have some type of modification in reference to the base.

This Bank Account Fraud (BAF) dataset suite was published at the 2022 NeurIPS (Neural Information Processing Systems) Conference [12] and is realistic, imbalanced, dynamic, privacy preserving, and has controlled biases [10].

> BAF was generated using feature selection — "features were selected based on 5 LightGBM models feature importance" — noise mechanisms — "added Laplacian noise to the data before the generative process … categorical features were changed according to the prior distribution … and were binned" — generative modeling — "GAN architecture adapted to the tabular dataset domain (CTGAN) … a total of 70 GANs were tested, with random sampling of hyper parameters" — as well as filtering and transformations — "generated datasets were sampled to not contain repeated instances with reference to the original dataset … transformations were applied to maintain original observed behaviors" [10].

The base dataset that this research works with is composed of 1 million instances, 30 realistic features "used in the fraud detection use-case", temporal information about the dataset, and protected attributes [10].

4.1 Dataset Description

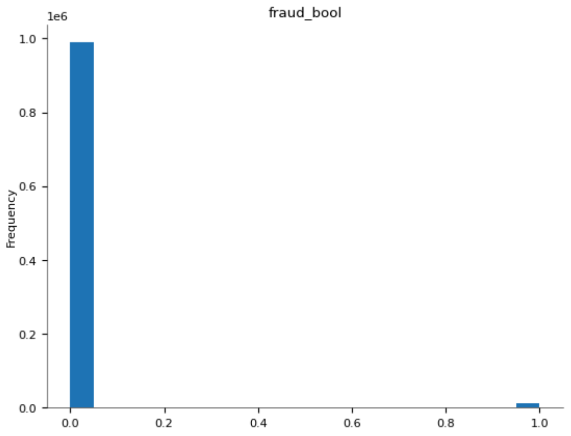
The purpose of this dataset is to link fraudulent and legitimate bank accounts to simple bank account registration information. In the base dataframe that this research works with, there are 1 million entries and 32 columns. The 1 million entries are a collection of registered bank accounts. The 32 columns contain the 31 features, consisting of individual categories of data such as income, name email similarity, device operating system, etc., and a column that provides temporal information about the data. The 32nd column is this temporal informative column that describes the dataset under the column name of month. This column is dropped when training and testing the models so that the models are trained on the 31 features and tested with 30 features to predict the 1st feature. The 1st feature is the fraud boolean (fraud_bool), indicating if the account is legitimate (labeled as 0) or fraudulent (labeled as 1). This fraud_bool is synonymous with the y-matrix in this research which is the output that the model trains to and is tested against.

The 31 features are: fraud boolean, income, name-email similarity, previous address months count, current address months count, customer age, days since request, intended balcon (balance consolidation) amount, payment type, zip count 4w, velocity 6h, velocity 24h, velocity 4w, bank branch count 8w, date of birth distinct emails 4w, employment status, credit risk score, email is free, housing status, home phone valid, mobile phone valid, bank months count, has other card, proposed credit limit, foreign request, source, session length in minutes, device OS, keep alive session, device district emails 8w, device fraud count.

In the base dataset, bank account application age and income were binned, and the number of decimal places stayed consistent with the original dataset. 5 of the 31 features were categorical with the remaining 26 all numerical. For the categorical features: payment type, source, device OS (operating system), employment status, and housing status, a label encoder function gave the categorical data numerical values that was used for model creation and evaluation.

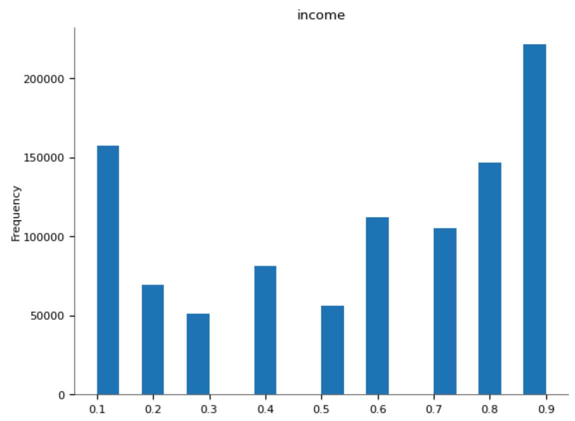
4.2 Dataset Visualization

To visualize some characteristics of the BAF Dataset Suite's base data that this research employs, 4 different charts are graphed below. These visualizations emphasize the distributions of the first 4 features in the data set.



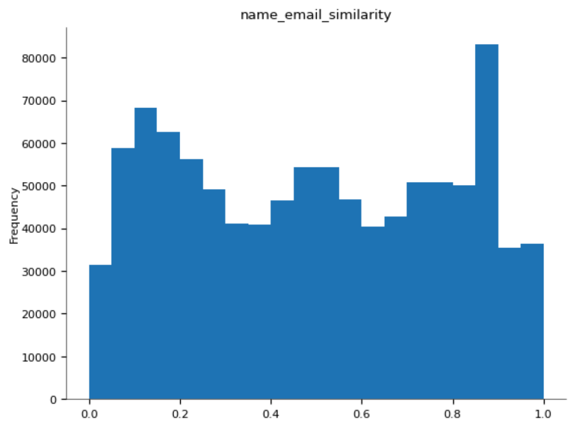
Visualization 1; chart of base dataset's fraud boolean distribution

Since the fraud boolean feature is a boolean value, there are only 2 possible values for fraud_bool (0 or 1). Clearly as represented in the visualization, the amount of legitimate accounts (labeled as 0) significantly outnumbers the amount of fraudulent accounts (labeled as 1). This research factors in this particular distribution when splitting the data before training and testing the models.



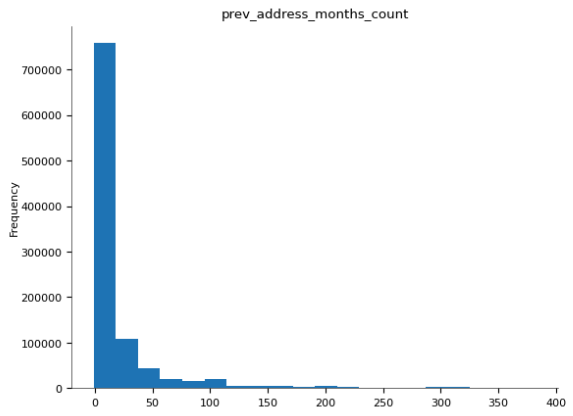
Visualization 2; chart of base dataset's binned income distribution

The reported income values that are collected during bank account registration are binned into groups, each with an upper and lower bound on a specific scale. With one million entries, binning the incomes into bounded groups rather than comparing every single specific account's reported income allows the models to find patterns of classifications based on the similarities of grouped incomes.



Visualization 3; chart of base dataset's name email similarity distribution

Name email similarity is measured from 0 to 1 with 0 indicating that there is no similarity between an account's registered name and email and 1 indicating that the name and email are very closely related. The models look at accounts with similar name email similarity to draw comparisons between their respective classifications.



Visualization 4; chart of base dataset's previous address months count distribution

Unlike other training dataset features like income and name email similarity, previous address months count depicts a strong skew with little variation in its distribution, making it harder for the models to use this feature to analyze an account and make a correct classification. However, paired with another feature or the entire training dataset, the previous address months count can be a significant factor in the models' decisions through its potential correlation with other variables.

In all these visualizations, the frequency of accounts, also referred to as the number of entries, is plotted on the y-axis and the x-axis is the value at which the feature is measured using a specific feature scale. With some correlated variables, features were charted in respect to one another, but the distributions showed no clear pattern in most cases since the large number of entries includes many outliers. Additionally, graphing individual values or time series created unclear data representations due to the size of the dataset. The research also included individual visualizations of just the legitimate or just the fraudulent accounts, but the comparative graphs often shared similar results due to a high number of outlining data points in the legitimate account large subset. Thus, this paper only provides individual visualizations of the first 4 features from the dataset, a small sample of graphs this research conducted but graphs with clear distributions.

4.3 Dataset Analysis & Methodology

Since the dataset consists of 988,971 legitimate accounts compared to the 11,029 fraudulent accounts, the models could assume that every bank account is legitimate and receive 98% accuracy scores in correctly classifying bank accounts.

To counteract this phenomenon, the data was split using size limits to have a roughly 1:1 ratio between legitimate and fraudulent accounts in training and testing our models.

For data preprocessing, the base dataframe was first split into two sub-dataframes — fraudulent accounts and legitimate accounts — using the fraud_bool feature. From the base dataframe, if the fraud_bool was equal to 1, the entry was sorted into the fraudulent subframe, and if the fraud_bool was equal to 0, the entry was sorted into the legitimate subframe.

With the numerical subframes, fraudulent and legitimate, the x and y matrices for each subframe were isolated. The x-matrix, used to predict the value of the y-matrix output, was the entire subframe minus the first feature, fraud_bool. The y-matrix, or output matrix, was just the fraud_bool feature, indicating if the bank account was legitimate or fraudulent.

Since the original number of fraudulent accounts was significantly less than the number of legitimate accounts, this research experimented with the optimal limit on the number of legitimate accounts that the models would be trained and tested on, finding that 15,000 worked consistently well as the new ratio of legitimate to fraudulent models was 15,000:11,029. This approximately 1:1 ratio created a more even distribution of account diversity while preserving the characteristics of the original dataset with a larger number of legitimate accounts. The more balanced ratio additionally forced the models to actually decipher if an account was legitimate or fraudulent rather than always predicting legitimate as the models would do with the 988,971:11,029 ratio.

Limiting the legitimate subframes' x and y matrices to 15,000, the new legitimate subframe's x-matrix and the fraudulent subframe's x-matrix were concatenated. After doing the same combination with the legitimate subframe's y-matrix and the fraudulent subframe's y-matrix, the final x-matrix and final y-matrix were formed. The final x and y matrices were then both split into 80% training data and 20% testing data. Randomizing the split, the X_train, X_test, y_train, and y_test matrices were created where the test data took 20% of the data and the train data took 80% of the data respectively. The raw data is then directly entered into the ML models.

5. Models & Methodology

This paper explores the efficacy of 8 different sklearn [13] supervised learning classification models that will be described below. Sklearn, sci kit-learn, is a machine learning framework in python for predictive data analysis [13]. Each of the models below was trained, hyperparameter tuned, and tested for accuracy using a series of model metrics.

For each model, the x array/vector holds the feature samples, and the y array/vector holds the class labels. The model visualizations are only two dimensional, but the base dataset has 31 features, so the models are trained with a multidimensional dataset and are therefore multi-layered. Additionally, each model is tuned in accordance with its hyper parameters. The specific tuning for each model is available for reference in the source code[1]. The tuning was completed in accordance with the dataset format and a multitude of other factors to ensure the highest performing models of each type.

Model 1: Logistic Regression

The logistic regression model is a linear model used for classification [13]. This model uses a logistic function to model the probabilities of the possible outcomes of a single trial [13]. The algorithm is trained with a one-vs-rest scheme and "implements regularized logistic regression" by default [13].

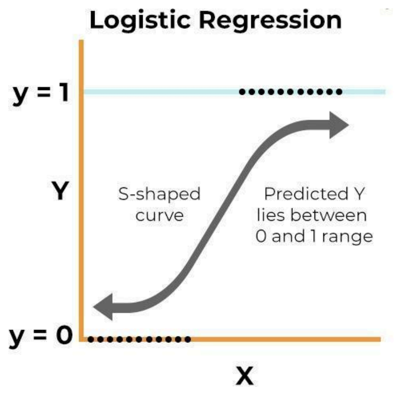


Figure 1; visualization of logistic regression [15]

Model 2: Ridge Classifier

The ridge classifier model is a variant of ridge regression, a linear model

[1] Source code access is referenced in the conclusion section

[13]. The model converts binary targets to {-1,1} and then uses regression to predict an outcome with the sign of the prediction corresponding to the sign of the predicted class [13]. The model penalizes Least Squares' large coefficients [13].

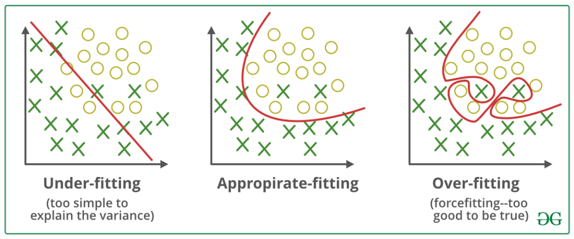


Figure 2; visualization of ridge classification fitting [16]

Model 3: Decision Tree Classifier

The decision tree classifier model is a non-parametric method "that predicts the value of a target variable by learning simple decision rules inferred from the data features" [13]. Piecewise functions with constant approximation make up a tree [13]. A deeper tree results in a more fitted model with complex decision rules [13].

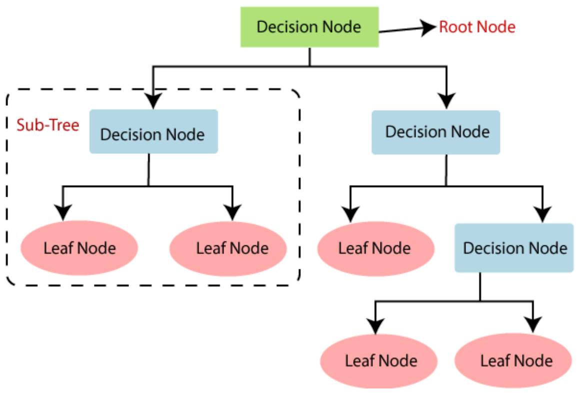


Figure 3; visualization of decision tree classifier [17]

Model 4: Random Forest Classifier

The random forest model is a collection of decision tree classifiers that uses averaging to make meta estimates [13]. Each tree is not built with the whole dataset as sub-samples of the dataset are used to build trees so as to not overfit [13]. By not overfitting, the model is less likely to make false predictions and is more applicable to larger sets of new data.

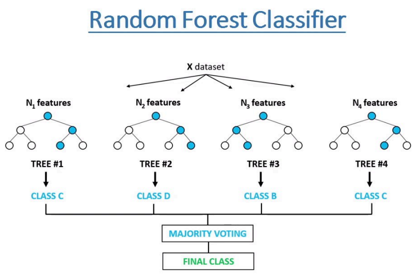


Figure 4; visualization of random forest classifier [18]

Model 5: Gradient Boosting Classifier

The gradient boosting model, also referred to as XGBoost, is a gradient boosting decision trees model that generalizes boosting "arbitrary differentiable loss functions" [13]. "The algorithm builds an additive model" allowing for optimized functions where additional trees "fit on the negative gradient of the loss function" with each stage [13].

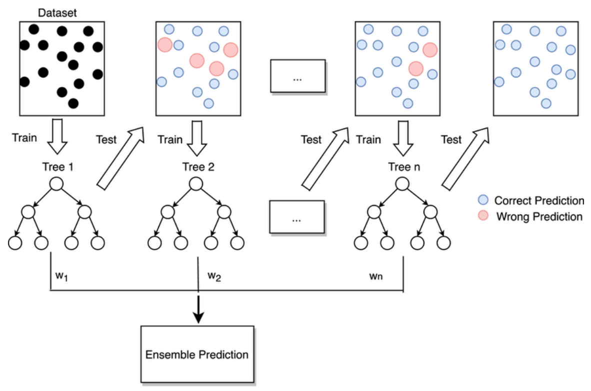


Figure 5; visualization of gradient boosting classifier [19]

Model 6: Nearest Neighbors Classifier

The nearest neighbors classifier is a form of instance-based learning and non-generalizing learning where the classifier does not construct an internal model but rather stores instances of the training data to be later compared [13]. This model is a k-neighbors classifier where the classifier learns based on the integer k nearest neighbor of the query point assigned, which is the point that is most representative of the nearest neighbor to the given data point [13].

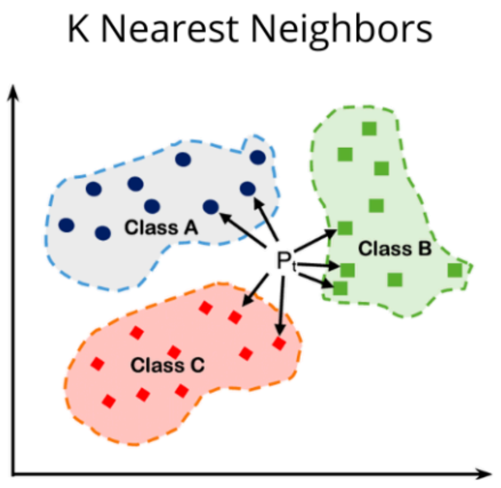


Figure 6; visualization of nearest neighbors classifier [20]

Model 7: Linear Support Vector Classifier

The support vector classifier, a type of support vector machines, uses kernel approximations and manipulations of the kernel functions for predictive analysis [13]. With many features, the classifier can multidimensionally model outputs with specific kernels [13].

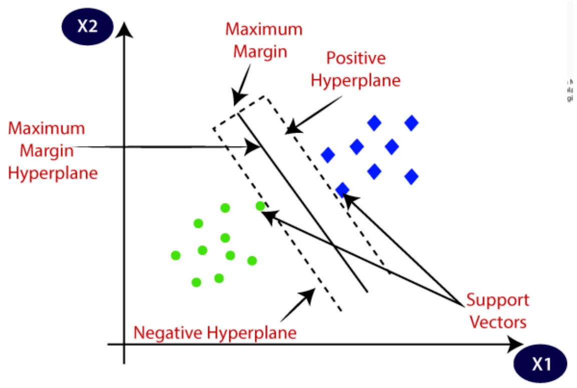


Figure 7; visualization of linear support vector classifier [21]

Model 8: MLP Classifier

The MLP Classifier is a type of Neural Network for supervised learning classification [13]. The model "implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation" [13]. This neural network trains on floating point feature vectors which consist of the training samples [13]. The model is self-modifying, using some forms of loss functions and gradients, cycling through the training data while changing the weights of each variable within the model to correctly detect a fraudulent or legitimate bank account. With every layer, the model is learning from the past layer and self-training to correctly value the weights to make optimal output predictions.

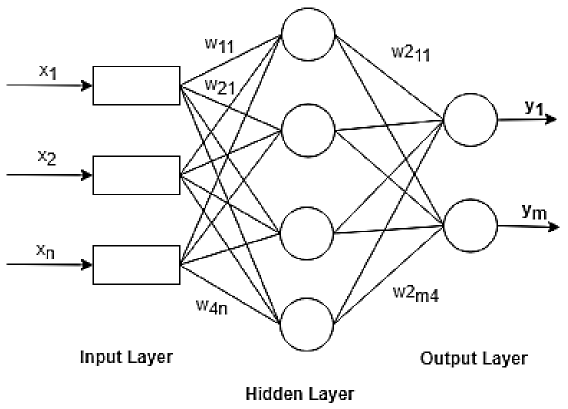
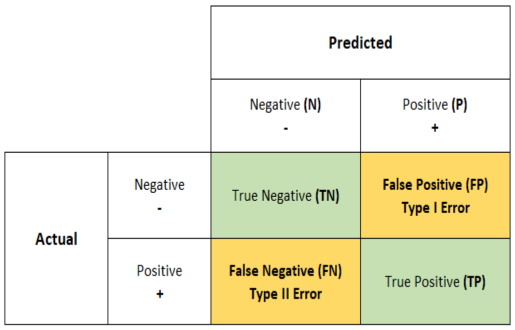


Figure 8; visualization of MLP classifier [22]

6. Results and Discussion

To evaluate the models, the research uses a series of metrics and scoring tools to see how the models performed on the testing data. In the table explained in this section, accuracy score, recall, precision, F1 score, and AUC are all explored. In this section, each models' confusion matrix and ROC curve is also depicted.

A confusion matrix is a visualization of the distribution of classifications a model makes. If the model predicts an entry to be legitimate based on the entry's x-matrix containing the many features of bank account registration information, the entry is tagged with a predictive label of 0. If the model predicts the account to be fraudulent, the entry is tagged with a predicted label of 1. These predicted labels are then compared with the true label, contained in the y-matrix which holds the actual label of the bank account as 0 (legitimate) or 1 (fraudulent). The confusion matrix is a visual representation of the amount of bank accounts the model predicts correctly and incorrectly and how the models' predictions are distributed. The sections of the matrix that are correct classifications are the top left box, a true negative or TN, meaning that the model predicted a legitimate account, and the actual/true state of that account is legitimate, and the bottom right box, a true positive or TP, meaning that the model predicted a fraudulent account and the actual/true state of that account is fraudulent. A false positive or FP, known as type I error, is the top right box where the model predicts a fraudulent account, but the account is actually legitimate. A false negative or FN, known as type II error, is the bottom left box where the model predicts a legitimate account while the account is actually fraudulent. Since the research only has two output possibilities with a potential 0 (legitimate) or 1 (fraudulent), the models have a binary classification confusion matrix.



		Predicted	
		Negative **(N)** -	Positive **(P)** +
Actual	Negative -	True Negative **(TN)**	**False Positive (FP) Type I Error**
	Positive +	**False Negative (FN) Type II Error**	True Positive **(TP)**

Figure 9; description of confusion matrix [23]

When tuning ML models, researchers can favor one type of error over another. There exist many arguments to train the models so that they value a certain type of error over the other type of error. The argument to support type I error in this case is that since the models are used as the first layer of defense against fraud, this research would rather tag a true/actual legitimate account as fraudulent for further review by a cybersecurity professional team than let a fraudulent account slip through the models' detection. On the other hand, the argument to value a type II error is that this research should never shut down a legitimate account by mistaking it as fraudulent, similar to the criminal justice system's belief that it is better to let ten guilty individuals free than to have one innocent individual suffer.

In this research, none of the models are tuned or trained to value a type of error because of these two arguments. The decision to favor a type of error is based on how these models are implemented into a bank's system. If the models are used in the manner as recommended in this paper, as a flagging device, then it is reasonable for banks to value a type I error. However, many banks can use these models to detect the potentially fraudulent accounts, immediately blocking them once identified. Those banks should value type II error to minimize mistakes in shutting down accounts. Type II error is a much more conversative approach towards the detection of fraudulent accounts in comparison to type I error. Because this research does not know how the models will be used, no type of error is preferred.

Every model is also evaluated with the ROC curve, short for receiver operating characteristic curve. The ROC curve is created by plotting the true positive rate against the false positive rate. In essence, the ROC is plotting the model's recall (a useful metric that will be explained later in this section that is synonymous with sensitivity in other fields) on the y-axis and the specificity of the model on the x-axis. While accuracy, recall, precision, F1, and AUC scores are used in this research to compare the models, it is important to note ROC's use of sensitivity (TP/TP+FN) on the y-axis and specificity (TN/TN+FP) on the x-axis. The shape of the curve is an important indicator of the models' ability to correctly classify as the more extreme the curve bends the better, and the AUC, area under the curve, is an important metric that the research also observes when evaluating the models.

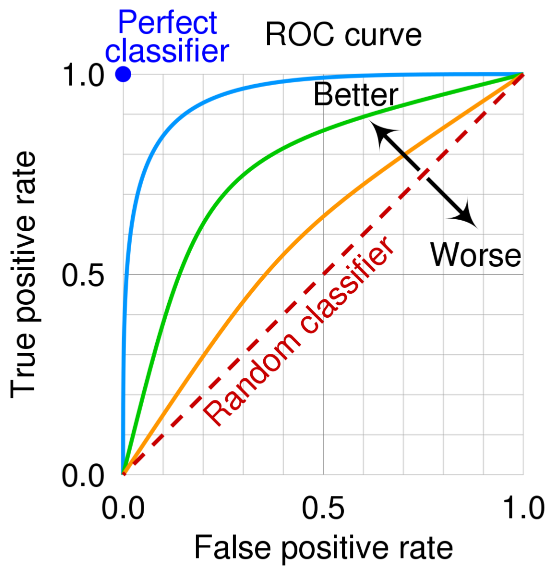
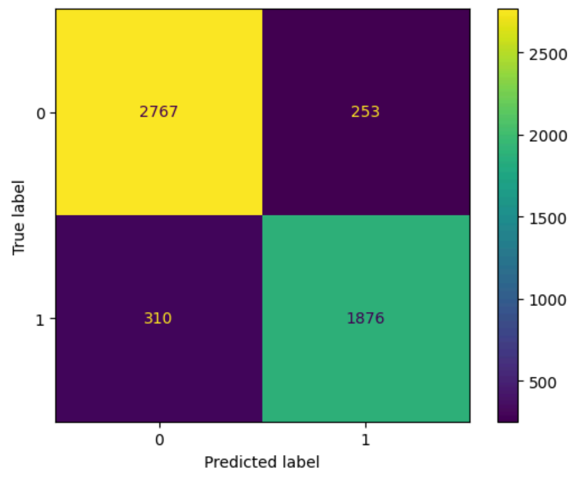


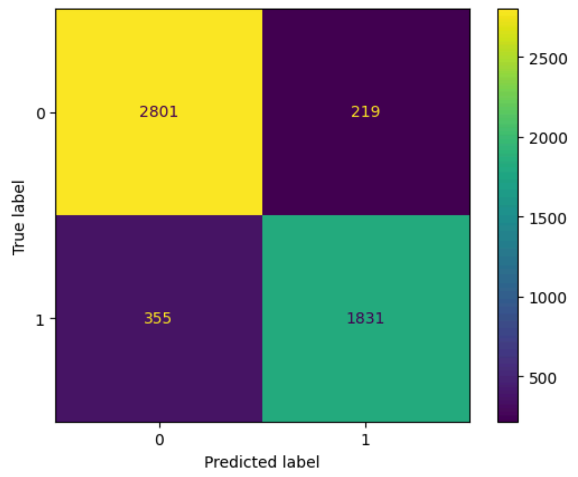
Figure 10; description of ROC curve [24]

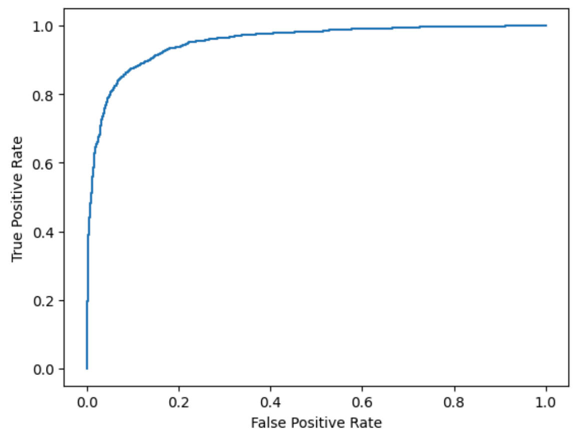
6.1 Model Results

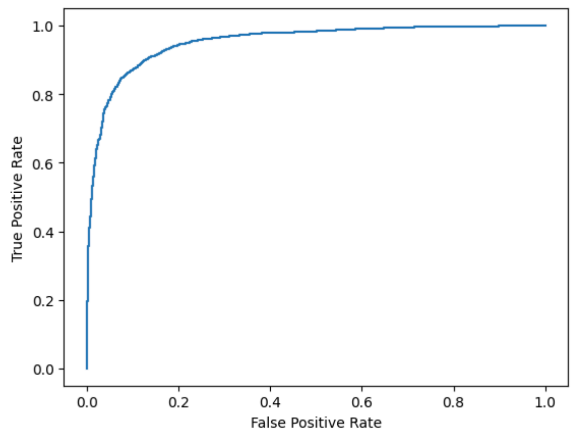
Model 1: Logistic Regression

Model 2: Ridge Classifier

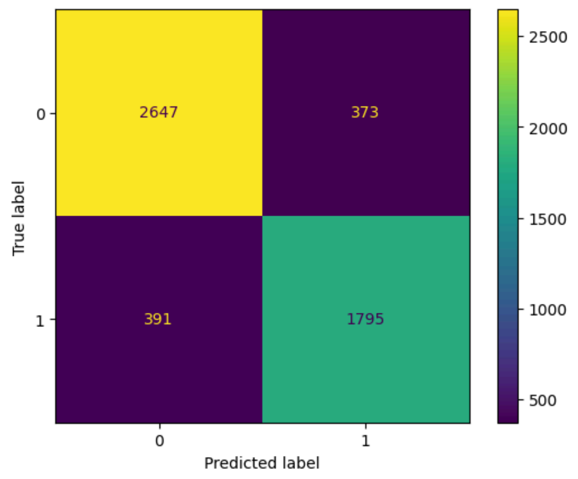


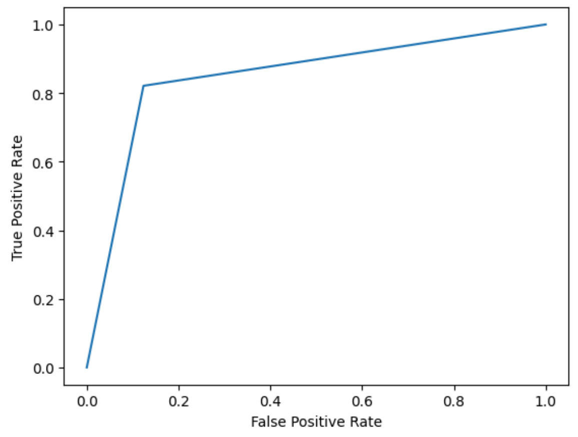


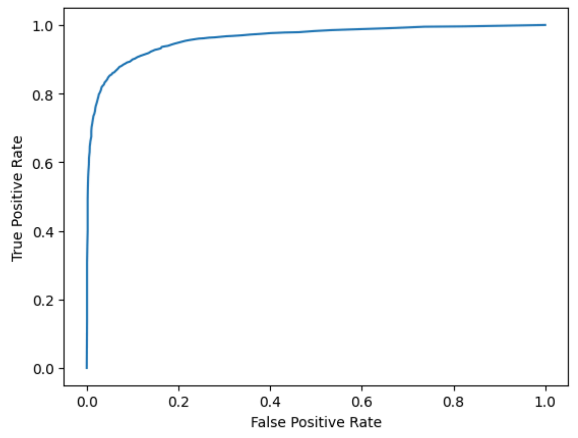




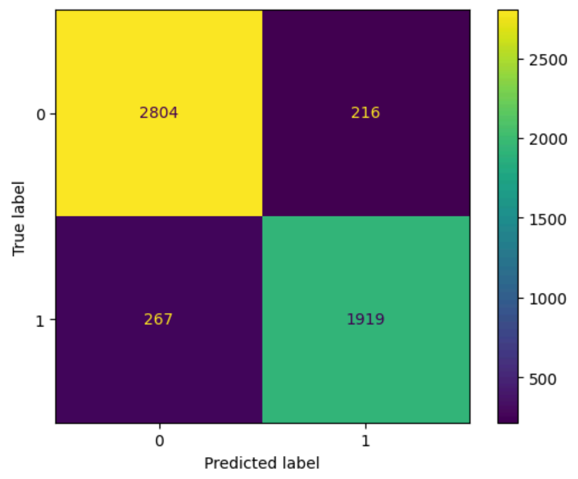
Model 3: Decision Tree Classifier



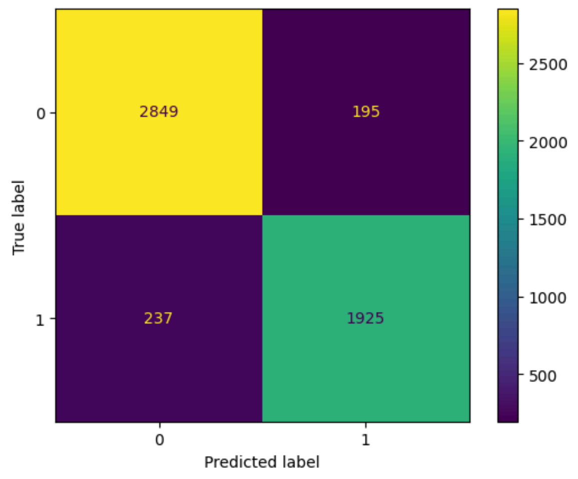


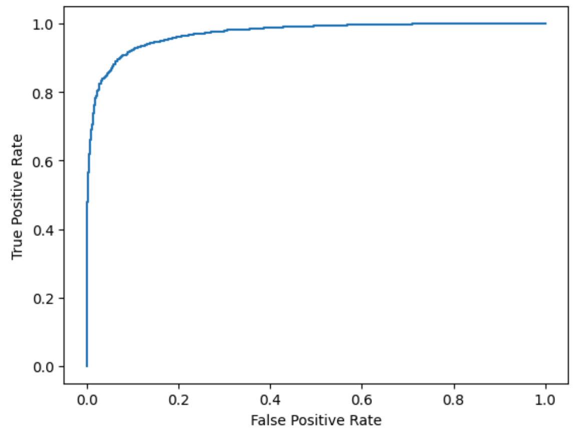


Model 4: Random Forest Classifier

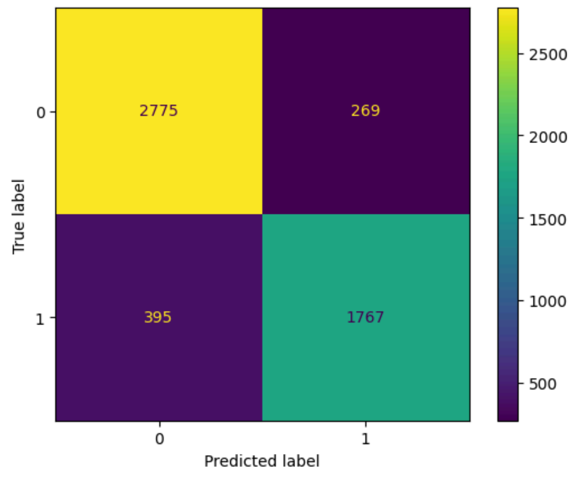


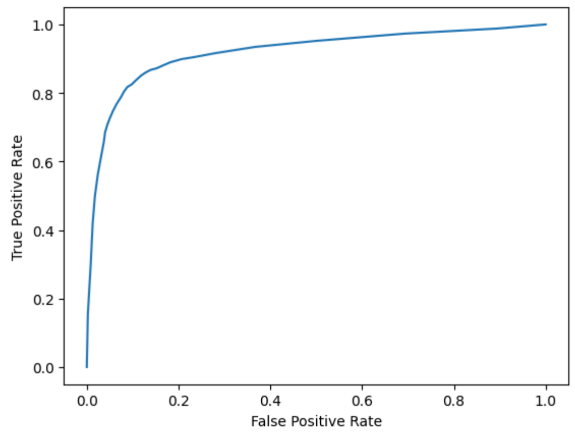
Model 5: Gradient Boosting Classifier



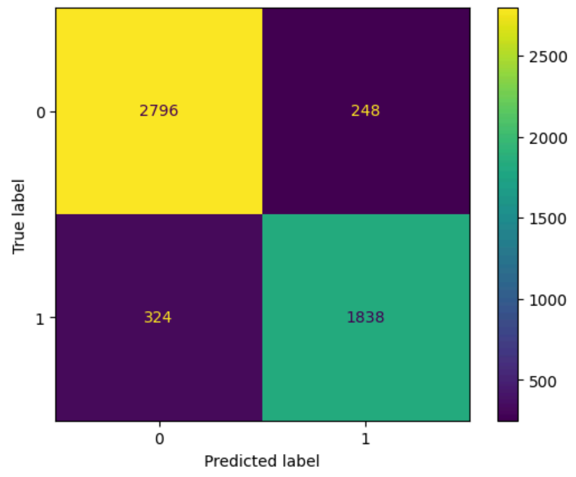


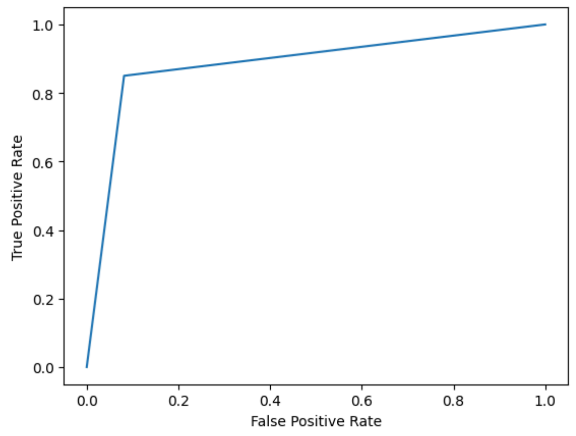
Model 6: Nearest Neighbors Classifier



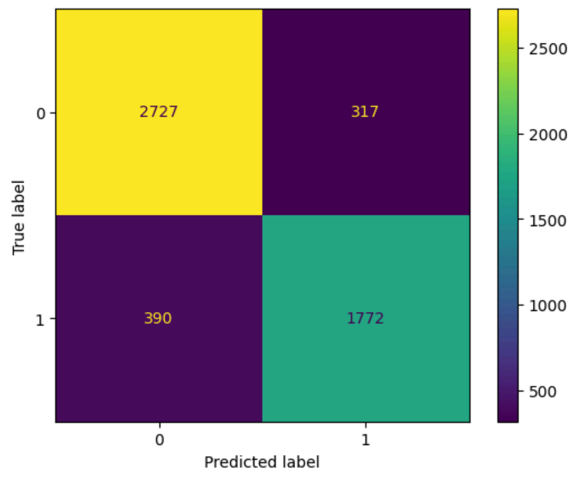


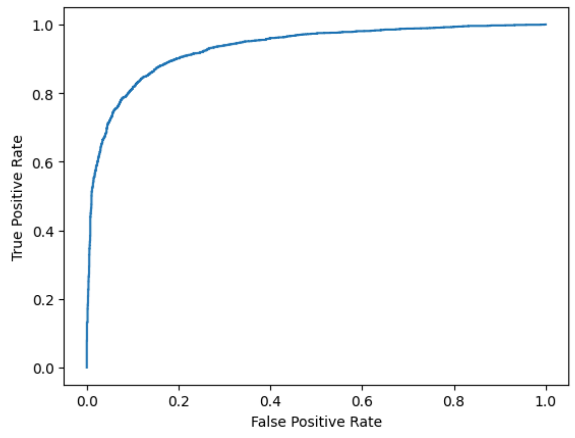
Model 7: Linear Support Vector Classifier





Model 8: MLP Classifier





6.2 Explanation

As seen in all the models' confusion matrix to a varying degree, a pattern appears where the bottom left box has more classified entries than the top right box. This is due to the tuning of the models' hyperparameters. While tuning the models, this research did not value an error type over the other, so some of the model parameters were left in their default state. This resulted in this dominant false negative pattern indicative of a type II error heavy phenomenon where this research suspects model overfitting. The pattern indicates a higher level of false negatives which is due to the models' tendency to fine tune its weights to match the training data in a complex way. By overfitting to the training data, the weights often fail to translate properly to new data as seen in the testing dataset. However, if this research were to combat this phenomenon, it would have to weigh the error types unequally or limit the models' ability to fine tune itself, each resulting in a loss of the total model accuracy. In this case, the models slightly overfit by over cycling as seen with the higher false negative rate, but the weights still hold strong when applied to the testing data and net high accuracy scores. Thus, this research does not retest different variants of the models because any additional tuning to change false negative and false positive rates would decrease accuracy scores. This paper addresses this observable pattern and considers it for model recommendation.

6.3 Scores

In this section, this research measures the models with a series of metrics and scores: accuracy, recall, precision, F1, and AUC. In the table below, each model and its respective scores are listed. Accuracy score is the percentage of entries correctly labeled, which is the number of correctly classified entries over the total number of entries. Recall is the ratio of true positives over the sum of true positives and false negatives. Recall is a measure of the model's ability to find all the positive entries. Precision is the ratio of true positives over the sum of true positives and false positives. Precision is a measure of the model's ability to not classify a negative entry as positive. F1 is 2 times the product of precision and recall all over the sum of precision and recall. F1 is the harmonic mean of the recall and precision of the model. Finally, AUC is the area under the curve of the ROC (receiver operating characteristic) curve from classification scores.

$$R = \frac{T_p}{T_p+F_n}$$

$$P = \frac{T_p}{T_p+F_p}$$

$$F_1 = \frac{2T_p}{2T_p+F_p+F_n}$$

```
accuracy = (TP + TN) / (TP + TN +
FP + FN)

recall = TP / (TP + FN)

precision = TP / (TP + FP)

F1 = 2 * TP / (2 * TP + FN + FP)


F1 = 2 * (precision * recall) /
(precision + recall)
```

Each score reports something different about the models' performance. Each score is calculated with unique formulas that compare different ratios for unique contextual results. This research does not value a single metric over the rest, but rather looks at them in collection. A model with a high score in one metric will most likely be a strong model across all of the evaluating factors, and this paper makes its recommendations based on the total model performance.

6.4 Comparing Models:

Model *Score*		*Accuracy*	*Recall*	*Precision*	*F1*	*AUC*
Logistic Regression		89.19%	88.72%	89.02%	88.86%	95.26%
Ridge Classifier		88.97%	88.25%	89.03%	88.58%	95.20%
Decision Tree Classifier		85.32%	84.88%	84.96%	84.92%	84.88%
Random Forest Classifier		90.72%	90.32%	90.59%	90.45%	96.14%
Gradient Boosting Classifier		91.70%	91.32%	91.56%	91.43%	97.09%
Nearest Neighbor Classifier		87.25%	86.45%	87.16%	86.75%	91.87%
Linear Support Vector Classifier		89.01%	88.43%	88.86%	88.63%	88.43%
MLP Classifier		86.42%	85.77%	86.16%	85.95%	93.18%

6.5 Discussion

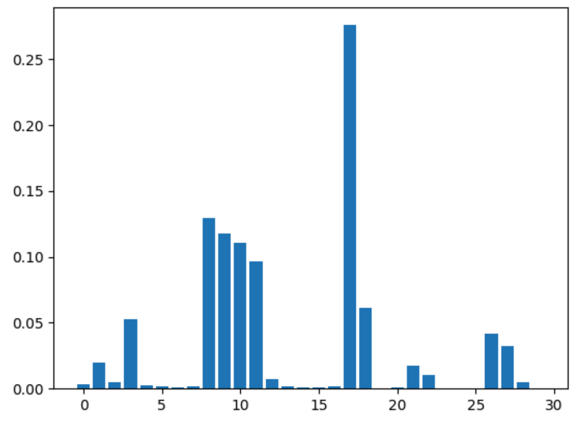
As expected, one specific model did not incredibly outperform the others. Rather, there exists a bundle of top performing models with similarities between them. The strongest bundle of models are the ensemble models with the Gradient Boosting Classifier and Random Forest Classifier with the second strongest bundle of models being the linear models with the Ridge Classifier and Logistic Regression. Ensemble models, using multiple ML algorithms collectively, work well for this classification because of the multidimensional nature of the dataset. With 30 features taken into account to make a binary classification, the best models are those that are created from multiple decision tree-based algorithms working together to cipher the information into a result. Linear models similarly perform well due to their ability to take the features and assume a linear combination with the outputs of the binary classification. By individualizing every single feature in its own relationship with the output, the linear models can compare a single feature's indication of an output and overlay it with all the remaining features. With all ML models, these models' ability to self-modify weights at every decision point or feature relationship allow it to self-learn for optimal performance.

6.6 Feature Importance

Examining this research's most successful models, this paper analyzes the feature importance for each one. For each of the top 4 highest performing models, the feature importance is graphed. In each graph, the 30 different features were assigned a respective value that indicated how important that individual feature was in making the classification.

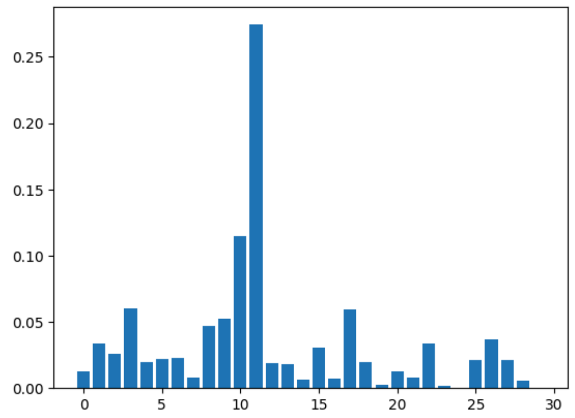
In the feature importance graphs, the y-axis measures the assigned value for every feature on a certain scale while the x-axis marks the features. The models used 30 features to predict the fraud_bool, and those 30 features are plotted in the same order they exist in the dataset. However, these graphs use their index numbers (0-29) to denote the features. So, the first feature in the dataset that the model looks at, which is income, is graphed at the 0 position because the income feature's index number is 0.

Gradient Boosting Classifier:



Graph 1; gradient boosting classifier feature importance

Random Forest Classifier:

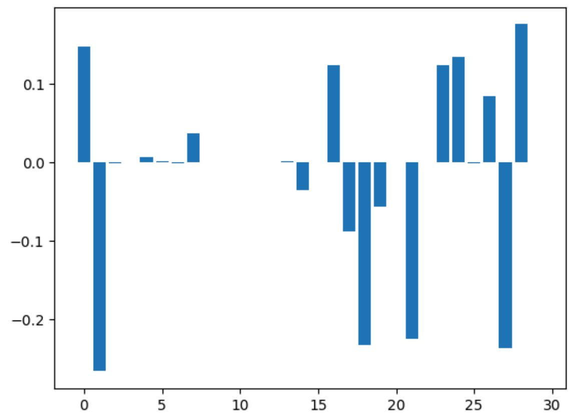


Graph 2; random forest classifier feature importance

For the gradient boosting classifier and random forest classifier, each feature is assigned either a 0 or positive value. A 0 indicated that the feature had no effect on the model's classification while the larger positive values show features that were large indicators of a certain classification. The sum of all feature importance values is 1.0, meaning that 100% of the classification decision was made by all the features. Using this logic, both classifier models have one feature with a feature importance value of over 0.25 meaning that 25% of the classification was derived from one single feature. Interestingly, this dominating feature was different for these models. For the gradient boosting classifier model, the most influential feature was the 18th (index number 17), housing status. The random forest classifier's most influential feature was the 12th (index number 11), velocity 4w (velocity of total number of bank account applications submitted in the past 4 weeks).

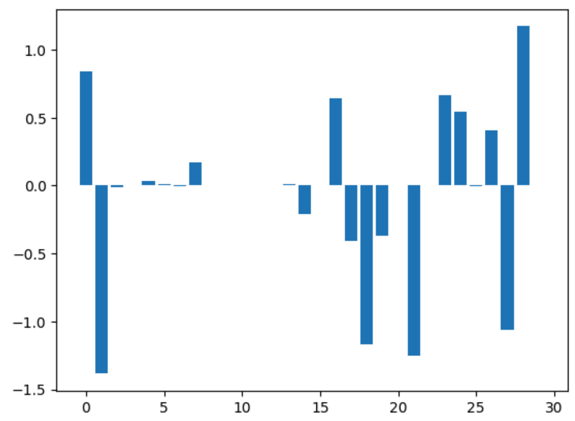
While the feature importance graphs share several similarities like general shape, the individual feature importance spikes occur at some different features on the x-axis, and the gradient boosting classifier takes less features into account than the random forest classifier. In the gradient boosting classifier feature importance graph, there are many more features with assigned values that are roughly 0 with only a few features making up most of the decision. In the random forest graph, many features have low assigned values that aren't zero meaning more features are a part of the decision just with less influence.

Ridge Classifier:



Graph 3; ridge classifier feature importance

Logistic Regression:



Graph 4; logistic regression feature importance

For the ridge classifier and logistic regression feature importance graphs, features are assigned with 0, positive, or negative values depending on their individual relationship with the classification decision. If the feature has a direct relationship with the classification, it has a positive value. If the feature has an inverse relationship with the classification, it has a negative value. If the feature has no relation to the classification, it is assigned a 0 value. The linear models' feature importance graphs are almost identical. The only noteworthy difference between the graphs is a different scale to measure feature importance on the y-axis. Other than the scale, the way the features are related with the classification within the model is almost the exact same.

In both the ensemble models' successful bundle and the linear models' successful bundle, the classifiers that make up the bundle share similar feature importance graphs. This is not surprising as top performing models of the same suite share similar decision pathways and decision characteristics. However, there exists many interesting differences between the feature importance graphs of the ensembles and linear models. This is due to the nature of the models. Since the models use different algorithmic techniques to make the classification, they have different decision styles and therefore weigh the features at different degrees.

7. Conclusion

To optimally detect potential bank drops from bank account registration information, this research compared 8 different supervised learning classification models for this predictive binary classification case: logistic regression, ridge classification, decision tree classifier, random forest classifier, gradient boosting classifier, nearest neighbor classifier,linear support vector classifier, and MLP classifier. Each model was hyperparameter tuned, trained, and tested on the 1 million entries by 31 features dataset, and evaluated using a series of algorithmic metrics. This research concludes that the most successful models this paper recommends for the financial industry to implement into their systems are the gradient boosting classifier, random forest classifier, ridge classifier, and logistic regression models. Ensemble models, such as the gradient boosting classifier and random forest classifier, are well fit for multidimensional use that is seen in this research's large number of features. Linear models, such as the ridge classifier and logistic regression, are strong for analyzing individual feature importance in making the classification prediction.

7.1 Future Considerations

At the time of bank account registration, no fraud has occurred on a bank account. However, these models prove that ML can be successful in detecting future fraudulent bank accounts from just the registration information. As a result, these models are recommended for the financial industry as detection tools for banks to use to monitor potential bank drops. With the integration of these models into a larger network of cybersecurity tools, technology will have the power to detect and prevent bank fraud before it begins, creating a safer financial environment for all. The models' potential role in safeguarding banks by detecting fraud has no limit as larger, real-time datasets and increased testing with banking clearance will improve the models' functionality.

7.2 Limitations

The models proved to be very effective, with the highest achieving bundle of models consistently performing above 90% accuracy in correctly classifying bank accounts as either fraudulent or legitimate. However, with all ML algorithms, limitations do exist in this research. Due to privacy reasons that exist in the financial industry, this research does not have access to additional sources of potential data, current ML models that are used in the cybersecurity field, or more information about this dataset. Additionally, the models were not programmed to value one type of error over another. Finally, the models were trained and tested on only a specific set of features. While the features generally match bank account registration information, different features could also apply to this classification for better results. Despite this, the models were accurately tuned, trained, tested, and measured using properly certified data.

7.3 Source Code

Source code available upon request at email address js.research248@gmail.com.

8. Acknowledgements

I first want to thank my family for all their support. I additionally extend a huge thanks to the InspiritAI team for introducing me to machine learning and its applications. Finally, I want to thank and recognize my mentor Ben Wettstein for monitoring my progress throughout this research and answering any questions I had about the research process.

9. References

[1]
Vaishnavi Nath Dornadula, S Geetha. Credit Card Fraud Detection using Machine Learning Algorithms, Procedia Computer Science, Volume 165, 2019, Pages 631-641, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.01.057. (https://www.sciencedirect.com/science/article/pii/S187705092030065X)

[2]
Mishra, M.K., & Dash, R. (2014). A Comparative Study of Chebyshev Functional Link Artificial Neural Network, Multi-layer Perceptron and Decision Tree for Credit Card Fraud Detection. *2014 International Conference on Information Technology*, 228-233.
[3]
Minastireanu, Elena & Mesnita, Gabriela. (2019). An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection. Informatica Economica, 23. 5-16. 10.12948/issn14531305/23.1.2019.01.
[10]
kaggle datasets download -d sgpjesus/bank-account-fraud-dataset-neurips-2022
Jesus, S. (2023, November 29). *Bank Account Fraud Dataset Suite (neurips 2022)*. Kaggle. https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022/data
[11]
Goldblum, A., & Howard, J. (2010). *Find open datasets and Machine Learning Projects*. Kaggle. https://www.kaggle.com/datasets
[12]
NeurIPS 2022 The Thirty-sixth Annual Conference on Neural Information Processing Systems. NeurIPS 2022. (n.d.). https://nips.cc/Conferences/2022
[13]
Cournapeau, D. (2024, January). *1. supervised learning*. scikit learn. https://scikit-learn.org/stable/supervised_learning.html
[15]
Kanade, V. A. (2022, April 18). *Logistic regression: Equation, assumptions, types, and best practices*. Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/
[16]
Fiorentine, C. (2020, September 24). *Intro to Ridge and lasso regression*. Medium. https://medium.com/@chrisfiore13/intro-to-ridge-and-lasso-regression-e8b3271b5fd0
[17]
Decision tree algorithm in Machine Learning - Javatpoint. www.javatpoint.com. (n.d.). https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
[18]
Khushaktov, M. F. (2023, August 26). *Introduction random forest classification by example*. Medium. https://medium.com/@mrmaster907/introduction-random-forest-classification-by-example-6983d95c7b91
[19]
Zhang, T. (2020, October 29). *Flow diagram of Gradient Boosting Machine Learning method.* ResearchGate. https://www.researchgate.net/figure/

Flow-diagram-of-gradient-boosting-machine-learning-method-The-ensemble-classifiers_fig1_351542039

[20]
Sachinsoni. (2023, June 11). *K nearest neighbours - introduction to machine learning algorithms*. Medium. https://medium.com/@sachinsoni600517/k-nearest-neighbours-introduction-to-machine-learning-algorithms-9dbc9d9fb3b2

[21]
Saini, A. (2024, January 23). *Guide on Support Vector Machine (SVM) algorithm*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[22]
Iwendi, C. (2020, April). *Structure of MLP classifier. | download scientific diagram - researchgate*. ResearchGate. https://www.researchgate.net/figure/Structure-of-MLP-classifier_fig3_340629569

[23]
Suresh, A. (2024, January 18). *What is a confusion matrix?*. Medium. https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

[24]
Everton Gomede, P. (2023, October 28). *The ROC curve: Application and interpretation in the health context*. Medium. https://medium.com/@evertongomede/the-roc-curve-application-and-interpretation-in-the-health-context-cc06af05a6ec