Kennesaw State University

DigitalCommons@Kennesaw State University

Master of Science in Computer Science Theses

Department of Computer Science

Fall 12-16-2020

Classifying Imbalanced Financial Fraud Data Utilizing Enhanced Random Forest Algorithm

Charles Gardner

Follow this and additional works at: https://digitalcommons.kennesaw.edu/cs_etd

Part of the Data Science Commons

Recommended Citation

Gardner, Charles, "Classifying Imbalanced Financial Fraud Data Utilizing Enhanced Random Forest Algorithm" (2020). *Master of Science in Computer Science Theses*. 40.
https://digitalcommons.kennesaw.edu/cs_etd/40

This Thesis is brought to you for free and open access by the Department of Computer Science at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Master of Science in Computer Science Theses by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact digitalcommons@kennesaw.edu.

Classifying Imbalanced Financial Fraud Data Utilizing Enhanced Random Forest Algorithm

A Thesis Presented to

The Faculty of the Computer Science Department

by

Charles Vincent Gardner, Jr.

In Partial Fulfillment

of Requirements for the Degree

Masters of Science, Computer Science

Kennesaw State University

December 2020

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

Charles Vincent Gardner, Jr.

Notice To Borrowers

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Charles Vincent Gardner, Jr.

1100 S Marietta PKWY, Marietta, GA 30060

The director of this thesis is:

Dr. Dan Lo

1100 S Marietta PKWY, Marietta, GA 30060

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Classifying Imbalanced Financial Fraud data Utilizing Enhanced Random Forest Algorithm

An Abstract of

A Thesis Presented to

The Faculty of the Computer Science Department

by

Charles Vincent Gardner, Jr.

In Partial Fulfillment

of Requirements for the Degree

Masters of Science, Computer Science

Kennesaw State University

December 2020

Abstract

Imbalanced datasets have been a unique challenge for machine learning, requiring specialized approaches to correctly classify the minority class. Financial fraud detection involves using highly imbalanced datasets with a class imbalance of up to .01% frauds to 99.99% regular transactions. It is essential to identify all frauds in financial fraud detection, even if some classifications' precision is low. I developed a random forest assembly that separates fraudulent transactions into tiers of precision. With this approach, 96% of fraudulent transactions are identified, showing an 8% increase in recall when compared to standard approaches. 59% of fraud classifications' precision increases by 10% up to 98% by optimizing several random forests on different fitness functions. These models are then combined to act as a sieve with increasing tolerance for low precision classifications. The effectiveness of random forest for financial fraud detection is also improved through feature extraction techniques. Random forest is weak at detecting patterns between interdepended features. This problem is address through unsupervised feature extraction. I will demonstrate a new random forest architecture PCA-embedded random forest, which increased random forest performance.

Acknowledgments

I want to thank my Adviser, Dr. Dan Lo, for his guidance and encouragement throughout my studies. I am very thankful for this experience, which would not have happened without his help.

Table of Contents

List of figures

List of tables

CHAPTER 1

Introduction

According to the Nilson Report, credit card fraud offers an important challenge to machine learning, with 31 billion dollars being lost each year [1]. Due to the number of transactions being processed each day, we must have robust machine learning algorithms to identify fraud. Financial fraud detection can be difficult due to the class imbalance of financial fraud data algorithm must not be biased by the majority class. The random forest has been shown to be effective at dealing with a class imbalance in [3]. Other methods, such as naive bays and logical regression, perform well with financial fraud detection. It has been demonstrated that random forest outperforms these other financial fraud classification methods in [2].

Random forest is a supervised learning technique created by Leo Breiman [9]. A random forest can be optimized to produce a wide range of results. Traditional fraud detection models are optimized to maximize both precision and recall using an F-score or ROC curve. This approach allows for most fraud to be caught and for the model to have high precision. The precision stratified random forest assembly seeks to identify fraud that is not identified by models optimized traditionally. Through optimizing multiple random forest models under different fitness functions, it was possible to classify 8% more fraud with no drop in precision.

The random forest algorithm can struggle with understanding patterns between interdependent features. Random forest makes separation linearly, allowing the random forest algorithm to make many separations, ultimately creating an accurate classification quickly. Linear separation can be problematic when a pattern only can be seen by looking at the relationship between multiple features. A new architecture PCA-imbedded random forest was made to address this problem. Feature construction can help improve the model's accuracy, but an automated approach would be helpful.

The PCA-embedded random forest automates the feature construction process by allowing each tree to have several unique features containing information from multiple features. PCA-embedded random forest increased the f-score of the model by 1% of the original random forest implementation. PCA-embedded random forest consistently provided high precision models with a tradeoff in the recall, which can be beneficial in some situations.

Chapter II

Related Work

Financial Fraud detection has been studied by many researchers over the past 25 years. Early studies in financial fraud such as [4] utilized Naive Bayes and the CART algorithm. CART is a decision tree classifier created in 1984, first published in [5]; an optimized version of the CART algorithm is used to create a decision tree when creating the random forest used in this paper. In [4], these algorithms tested against a balanced fraud dataset and produced 80% true positive rates. Early studies were often limited by the computation power requiring the use of efficient algorithms. The results of [4] demonstrated the potential strength of decision trees for classifying financial fraud.

Neural networks were used in financial fraud detection, starting in the late '90s. One of the early studies using neural networks compared Bayesian networks with neural networks [6]. In [6], imbalanced data created a problem due to the bias imbalanced data makes. The study resulted in finding that Bayesian produced better results. This trend has continued to be observed in [7] a survey of different classification techniques for financial fraud was conducted. This study found that neural networks were outperformed by most other approaches, including random forests, support vector machines, logistical regression, and variance Bayesian approaches. A recent study [8] explores new strategies to help neural networks deal with class imbalance. The improvement is promising but does not seem to be effective at the imbalance levels found in financial fraud datasets.

Bhattacharyya, S [11] performed a study comparing regression, random forest, naive Bayes, and support vector machines for financial fraud detection. The study resulted in regression reaching the highest performance with a fraud detection rate of .999971. Random forest was the second best with a fraud detection rate of .999969. These two approaches out preformed Naive Bayes and support vector machines by a relatively large margin .003. These comparisons were useful in selecting which algorithm to use in creating an assembly model for fraud detection.

Chao, C [12] explored how to improve random forest performance when applied to imbalanced datasets. This study investigated the effects of weighted random forests (WRF) and balanced random forests (BRF) to improves classification accuracy. It was found that WRF and BRF outperformed other approaches for handling class imbalance and producing similar results. Utilizing BRF and WRF resulted in a 3.2% increase in performance, making these approaches useful when using imbalanced datasets. BRF was shown to be faster than WRF when applied to large datasets and proved useful in this study.

Svante Wold [13] was among the first to use principal component analysis (PCA) in computer science. In an unsupervised manner, PCA's goal is to extract information from a group of features and store it into new orthogonal variables. Through doing so, information from many features can be used in classification using a few PCA features.

PCA is widely used to perform feature reduction due to its ability to maintain information from the features it is reducing. PCA will be utilized in this study to allow linear separations to make classifications based on information from multiple features.

Campus, K [15] performed a study in 2018 comparing decision trees, random forest, support vector machines, and logical regression. This study is of interest due to how recently it was conducted. The dataset used had a similar class imbalance to the dataset used in this study at .173% frauds. The results were provided using accuracy, specificity, and precision, with the random forest being the more effective model at 98.6% accuracy. Random forest was similar in precision and several percent better at the specificity.

Abbasi, A [15] performed a study using meta-learning to carry over past models' knowledge to increase future models' performance. Meta-learning was a useful approach in financial fraud due to the ever-changing nature of financial fraud. Meta-learning was done by collecting the bias from many models and apply the correct bias to incoming data. Apply past bias to the future problems was done through stack generalization. Stack generalization utilizes a wide range of machine learning classifiers and seeks to extract each classifier's best parts. Stacking is an assembling method for machine learning that uses the knowledge gained from many iterations to assess each model's value within the assembly. The authors were able to significantly improve their model's performance using six years of sequential data from various sources. This study outlined the potential value of using an assembly approach to address financial fraud challenges.

Liu, C [17] Utilized the random forest algorithm to determine if a company had committed financial fraud. Their feature set was created through the construction features based on endemic knowledge of financial fraud. This indicator included a company's current assets ratios, past asset ratios, and more. For their model to perform well, constructed features had to be valuable, and feature created noise needed to be removed. It was found that their performance was at its peak when using the top 8 out of 30 features. This feature distribution shows that even if the random forest is good at handling noisy features, including bad features, it hurts performance. This information proved useful in deciding how many PCA features to include in the RF-PCA.

PCA was used in [18] to classify financial fraud without the need for endemic knowledge. PCA was used for outlier detection condensing information from multiple features to find the largest outliers. This approach is like what will be done in the creation of RF-PCA. PCA has been used for unsupervised feature construction in [19], which utilized PCA to extract meaningful information from an extensive feature set. This approach demonstrates PCA's ability to preserve valuable information from multiple features. In [20], PCA was used as a statistical analysis tool to detect financial fraud, validating PCA features' independence. Independence between PCA features is important for RF-PCA because low correlation forests produce better results. PCA was shown to effectively extract information in small features sets in [20] valuable features were extracted from only 11 original features. The features set used in this study is small and so it is helpful to know PCA can still produce strong results with a small feature set.

Chapter III

Concepts

3.1) *THE RANDOM FOREST ALGORITHM*

This chapter will discuss how random forest makes classifications better understand how it can be optimized to improve performance. We will later cover two optimization strategies being randomized grid search and evolutionary optimization. These optimization strategies focus on finding a robust set of hyperparameters in a search space, which is too large to be fully explored.

We can see the various hyperparameters used in optimization by performing a trace of how random forest models are trained. First, we start with a feature set P described in equations (1 and 2). In equation (1), F represents a transaction that contains some amount k of individual features. Below n is the number of elements in the dataset, and k is the number of features. y is defined as the class of each value element in P for financial fraud y hold 0 or 1 to represent if the transaction is fraudulent or not.

$$F = (f_{1,}f_2, \dots f_k) \tag{1}$$

$$P = (F_1, F_2, \dots F_n) \tag{2}$$

With a dataset P, we can train a random forest; this is done by creating decision trees. To train a decision tree at each split in the tree, we need to introduce feature randomness.

Feature randomness allows for trees in the forest to be uncorrelated, increasing the accuracy of the forest. In (3), creating a subsection S of F in P will be described.

For some m max features with each s_i by selected randomly without replacement from the K features. Below m is the number of features used in each split, and K is the feature list.

$$\text{S} = (s_1 \in \text{K}, s_2 \in \text{K}, \ldots, s_m \in \text{K})\ for\ m \leq K$$

$$F' = (f_{s1}, f_{s2}, \ldots, f_{sm})$$

$$P' = (F'_1, F'_2 \ldots F'_n) \qquad (3)$$

The features inside P' will be used to split the dataset into two separate branches. The decision tree will compare each feature in P' to determine which splits the data best. There are two widely used processes for finding which feature should be used to separate the dataset. We will be using entropy, which seeks to reduce the node's entropy or disorder created by splitting the data. Entropy is minimized, with its lowest state being when all values within a node at the same class produce an entropy of 0. The process of finding the entropy score for each possible split is described below; the entropy equation can be found in (4) and (5).

For each $f_s \in F'$ create a list of possible split locations $X = (x_1, x_2, \ldots x_{n-1})$ with

$x \leq \max\left(P'_{F'_{fs}}\right)$ and $x \geq \min\left(P'_{F'_{fs}}\right)$. For each value in $x_i \in$ X, find the elements in P' with feature $f_s \geq x_i$ and store them in G, next store all values not included in G within L, $L = P' - G$. Now sum the member of G and L with a class value of 0, storing them in G_0

and L_0 and store the values with class value 1 in $G_1$and L_1. We can now find the entropy of the two-child nodes created by preforming a split at x_i.

$$e_G = -\frac{G_0}{|G|}\log_2\frac{G_0}{|G|} - \frac{G_1}{|G|}\log_2\frac{G_1}{|G|} \qquad (4)$$

The entropy of the child node of dataset L is given by

$$e_L = -\frac{L_0}{|L|}\log_2\frac{L_0}{|L|} - \frac{L_1}{|L|}\log_2\frac{L_1}{|L|} \qquad (5)$$

The best split for a feature F_{fs} will be determined by repeating (4) and (5) for each value in X. We now use e_G and e_L to find the total entropy loss from the split using (6) and storing it in e_t.

$$e_t = e_G * \frac{|G|}{|P|} + e_L * \frac{|L|}{|P|} \qquad (6)$$

The lowest value of e_t found after looking through all values in X will be the entropy for the given feature f_s. This process is repeated for all feature in F' until the feature which produces the lowest possible entropy is found. This feature will split the dataset if it is lower than its parent node's entropy. Optimization comes into play here as additional conditions can be placed on if a split should be accepted. Conditions such as the minimum number of elements allowed on a child node can stop a tree from splitting. Another parameter is the minimum number of elements on a leaf node. These parameters can significantly affect a tree's performance and, if optimized currently, can improve random forest performance.

3.2) Principle Component Analysis for feature construction

The principal component analysis algorithm is a feature reduction technique that is unsupervised. PCA focuses on projecting data from multiple features onto a new axis in a way that maximizes variance. PCA is excellent for condensing many invaluable features into a few new features. PCA will preserve information from multiple features giving a PCA feature value by magnifying many bad features into one useful feature. PCA adds value by eliminating useless features; random forest already does well at ignoring useless features. Thus, the new feature can be added to the feature set as a random forest already handles the unnecessary features well. These additional features allow the random forest to make classification using information from more than one feature simultaneously. Creating a new one-dimensional feature containing information from multiple sources features random forest's linear separations that can utilize information beyond a single feature.

Chapter IV

Model Evaluation and Dataset

Financial fraud datasets can be challenging to find due to the confidential nature of financial transactions. Often synthetic financial datasets are created to address this problem, which attempts to be analogous to real data. A synthetic dataset created Paysim mobile money simulator [14] was used in this study. The Paysim dataset contains approximately 6,300,000 transactions and nine features with 8,213 fraudulent transactions making the class imbalance 99.87% nonfraudulent to .13% fraudulent. The Paysim dataset was created by training an ai agent on real financial data then having that agent simulate transactions over 30 simulated days. The high-class imbalance in the Paysim dataset mirrors the imbalance in real financial data providing an excellent challenge to test our approaches against.

Due to the high-class imbalance found in our dataset model, evaluation can not be done using accuracy. If the model simply predicted every transaction as not fraudulent, it would have an accuracy of 99.87%. For this reason, it was necessary to use recall, precision, and f-score to evaluate model performance. These evaluators are biased to the positive class, and so the model is evaluated on how accurately it can identify fraud. Equations (7), (8), and (9) show how precision, recall, and f-score are calculated. F-score provided the harmonic mean between precision and recall, which encourages both being increased together. A model with a precision of .8 and a recall of .8 will have a higher f-score than a model with a precision of .7 and a recall of .9.

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{7}$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{8}$$

$$fscore = \frac{2 * precision * recall}{precision + recall} \tag{9}$$

Optimization of models can come with the risk of overfitting to the training dataset. The random forest was chosen to be used in this study because it is resistant to overfitting. Cross-validation is used to avoid overfitting during the optimization process of the precision stratified random forest assembly. Cross-validation is executed by using different sections of the training set as the test set over several iterations. A model must perform well using multiple training and testing datasets, which will make a model perform poorly if overfitting has occurred.

Chapter V

Optimization of the Random Forest Algorithm

By following the random forest trace in chapter 3, we can see that a decision tree will select its best possible move until entropy can no longer be reduced by finding a new split. We can make the final decision tree vastly different from one from a model containing no optimization by controlling the hyperparameters. The effects of changing the maximum depth, minimum samples per split, and minimum samples on each leaf are not independent. Interdependent parameters mean that the search space for the most optimum parameters is extensive, requiring a guided search.

Two optimization methods were used in the study a randomized grid search and evolutionary optimization. Randomized grid search takes the search space shown in table 1 and selects random assortments of parameters. The parameters are then used to build a random forest using cross-validation. After each model has been trained, its performance will be compared to a fitness function. Three fitness functions of maximum f-score, maximum precision, and maximum recall were used. Each produced vastly different models with fraudulent class recalls from 59% to 98% and a range of precision from 4.2% to 95.60%.

Parameters	Values					
Bootstrap	True	False				
Criterion	Gini	Entropy				
max depth	10	15	20	30	40	50
Max features	Auto	Sqrt				
Min sample leaf	1	2	4	8		
Min samples per split	2	5	10			
Forest size	10	15	20	30	40	

Table 1, The search space of randomized grid search

The evolutionary approach uses a larger search space representing max depth, minimum sample per split, minimum samples per leaf, and forest size as binary strings. This approach allowed for faster optimization with higher confidence. Evolutionary optimization starts with a population of randomly generated feature sets. A random forest is created using each feature set and then tested against its fitness function. The top half of the population will survive to the next generation. The other half will be replaced with new, randomly generated feature sets. The nodes in the population then crossover information creating slightly different offspring for the next test. Finally, at a probability of 2% each for each bit in a binary feature string, there will be a chance the bit will flip. This process is repeated until it converges by no longer being able to find better sets of parameters. Evolutionary results did not produce a substantial increase in model accuracy, but it did converge quicker. A comparison between the two approaches will be shown below in table 2.

	Maximum precision	Maximum recall	Maximum f1 score
Randomized grid search	.980	.956	.881
Evolutionary algorithm	.960	.990	.883

Table 2, Randomized grid search compared to evolutionary optimization

We can see in table 2 the performance of a randomized grid search, and the evolutionary algorithm is comparable. The evolutionary algorithm was more efficient due to it being a guided approach. The evolutionary algorithm was also able to search a large search space that could impact the optimization's performance in some datasets. Ideally, the whole search space could be explored, so multiple iterations were done until the best results could be found. The optimization process was done using cross-validation to ensure there was no overfitting.

Chapter VI

Precision Stratified Random Forest Assembly

6.1) *Assembly Approach Compared to Single Model Approach*

The traditional single model approach focuses on creating a single balanced model utilizing f-score or roc curves. A single model approach allows for most but not all fraudulent transactions to be caught. Transactions have different tiers of risk-based on the nature of the transaction. This risk is displaying in an assembly approach as identifying transactions with varying levels of precision. If we look at two examples, the first being a low-risk transaction that could be fraudulent. A low-risk transaction could be a person using their credit card in a different geographical location than where they usually use it. This transaction could indicate that their card has been stolen, and so this transaction should be investigated. A high-risk transaction could be all the money in a person's account being withdrawn to an account they do not own. These two scenarios would call for different responses. The low-risk example may only be fraudulent 5% of the time, whereas the high-risk example might be fraud 99%. When using a single model, the low-risk example is often ignored as it decreases the precision at which the high-risk transactions can be identified. By creating multiple models, these transactions can be separated and treated with different levels of response. These low-risk transactions might require only a call from the bank to verify the transaction, while the high-risk transaction needs to be frozen before it goes through.

6.2) *Assembly Creation*

To create a precision stratified random forest, three models M1, M2, and M3, are made, maximizing precision, recall, and f1 score, respectively. Each model is optimized using the strategy discussed in chapter 2, with 50 models being created in each optimization. The three-models created had the following results in isolation displayed in tables 3, 4, and 5. Table 3 shows the model trained for maximum precision; it achieves a precision of .98. M1's high precision comes at the cost of having a recall of .59, resulting in a low f1 score. The frauds detected in this model will make up the first tier of our model, allowing us to separate 59% of the dataset into a model tier with a precision of 98%.

	Predicted Fraud	0	1
True Fraud			
0		1,397,356	22
1		731	1,056

Table 3, Model M1 optimized for maximum precision

Table 4 shows the result of M2, which was optimized to maximize the f-score. This optimization is what is traditionally used when creating fraud detection models. It will be compared to M1 to make our second tier of fraud classifications. We will see later that there is nearly 100% crossover between these two models. Crossover says that only four transactions were found in M1, which were not detected by M2. High crossover is essential; the tiers would not be valid if the higher precision tiers could not identify the transaction in lower precision tiers. The observation that this crossover does exist means that some transactions can more confidently be identified as fraud.

	Predicted Fraud	0	1
True Fraud			
0		1,397,316	62
1		322	1,465

Table 4, model M2 optimized for maximum F1-score

Table 5 shows the results of M3, which was optimized to maximize recall. M3 achieves a recall of 95.6%; this is 8% higher than what could be found by M2 meaning. This 8% of frauds would not be identified in a traditional random forest because the loss in precision would too large to validate, including them in the model. M2 only identified three frauds that were not included in M3. As a result, we can separate the transactions identified M2 from M3, allowing these transactions' precision not to be pulled down by the 8% of low precision fraud in M3. Identifying these additional frauds is important as these transactions should still be investigated further despite only a 4.2% chance that a tier 3 fraud is fraudulent.

	Predicted Fraud	0	1
True Fraud			
0		1,391,915	5,463
1		83	1,704

Table 5, model M3 optimized for maximum recall

We can see that there is a wide range of results under different optimization strategies. To utilizes models M1, M2, and M3 in an assembly model, we need to establish if lower-level models always found transactions identified in high precision models. To do this, we need to satisfy equation (10) to ensure that a higher precision model finds all fraud found in a lower precision model.

$$M1(tp, fp) \cap M2(tp, fp) = M1(tp, fp) \qquad (10)$$

We must also ensure that equation (10) is satisfied and equation (11), then we can separate frauds found in a higher tier model from the frauds found in a low tier model.

$$M2(tp, fp) \cap M3(tp, fp) = M2(tp, fp) \qquad (11)$$

We test equations (10) and (11) in Tables 6 and 7, respectively.

	Predicted Fraud	Transactions Include in tier 1 but not in tier 2	Transactions not exclusive to M2	Transactions only found in M2
True Fraud				
0		12	1397314	52
1		4	1370	413

Table 6, Results of M2 – M1

We can see in figure 5 that four fraudulent transactions were correctly identified by M1 that could not be found in M2. We can also see that 12 transactions that were not fraudulent were correctly identified by M1, which M2 could not identify. These misidentified transactions resulted in a small error resulting in only four transactions that will be included in tier 1, which cannot truly be shown to be of high precision. In the fifth column, we see the transactions predicted fraud found only in M2 these transactions make up our second tier of frauds. We can see that these transactions have lower precision than transactions found in M1. By performing this separation, transactions in tier 1 have a precision of 98%, while transactions in tier 2 have 88% precision. If these two tiers had not been separated, then the 1056 fraud in tier one would have only been identified at a precision of 96%.

Tier three is created by comparing M3 with M2 in the manner described in equation (11). Similarly, in figure 6, we will subtract the set of classification in M2 from

M3 to determine what M3 found which had not been seen in M2. Figure 6 details this comparison's results; we find that equation 11 is satisfied, and a new tier can be created.

	Predicted Fraud	Transactions Include in tier 3 but not in tier 2	Transactions not exclusive to M3	Frauds only found in M3
True Fraud				
0		2	1397314	5403
1		3	1542	242

Table 7, Results of M3 – M2

We can observe that M3 is much more likely to identify a transaction as fraudulent falsely. M3 incorrectly classified 5403 transactions as frauds, which were not classified as frauds in M2. This high false-positive rate is advantageous because higher precision classifications are separated away from M3. The final tier classifications identified an additional 242 fraud, which was not included in M1 or M2.

6.3) results of the precision stratified random forest

Tier 1 and 2 account for 1,469 or 82.8% of the true positives in the test dataset; the transactions in Tiers 1 and 2 are not affected by the false positives in M3. Using an assembly approach allows us to identify fraudulent transactions, which are normal ignored to maintain high precision. M3 identifies an additional 242 frauds increasing the recall of the precision stratified random forest to 95.7%. The inclusion of M3 in our random forest assembly allows for an extra 13% of frauds to found at a precision of 4.2%. This precision is significant in a dataset with only .13% of transactions being fraudulent; 4.2% precision greatly increases our ability to identify frauds. Table 8 will show the precision and recall at each tier of classification. An f-score optimized random forest has a precision of 96% and a recall of 82.8%.

Tiers	Frauds detected	Cumulative recall	Tiers precision
Tier 1	1056	.59	.98
Tier 2	413	.828	.88
Tier 3	242	.957	.042

Table 8, precision and recall at each tier of precision stratified random forest assembly

Utilizing the information provided by the precision stratified random forest assembly, differing response levels can be taken depending upon the risk posed by a transaction. A transaction found in the first tier likely requires immediate action due to the high likelihood of fraud. On the other hand, the transactions found in tier 3 might just require a warning to be sent on to a client asking if the transaction was them. Separation of precision can be useful in some situations, for example, if it can sometimes indicate a credit card is used out of town or on a strange purchase. The client might not want their purchase to block every time they leave town while also wanting protection if their card is stolen. Knowing the level of confidence of a transaction being fraudulent allows for better responses to potential frauds. Figure 1 shows the process of data being separated as it passes through each tier.

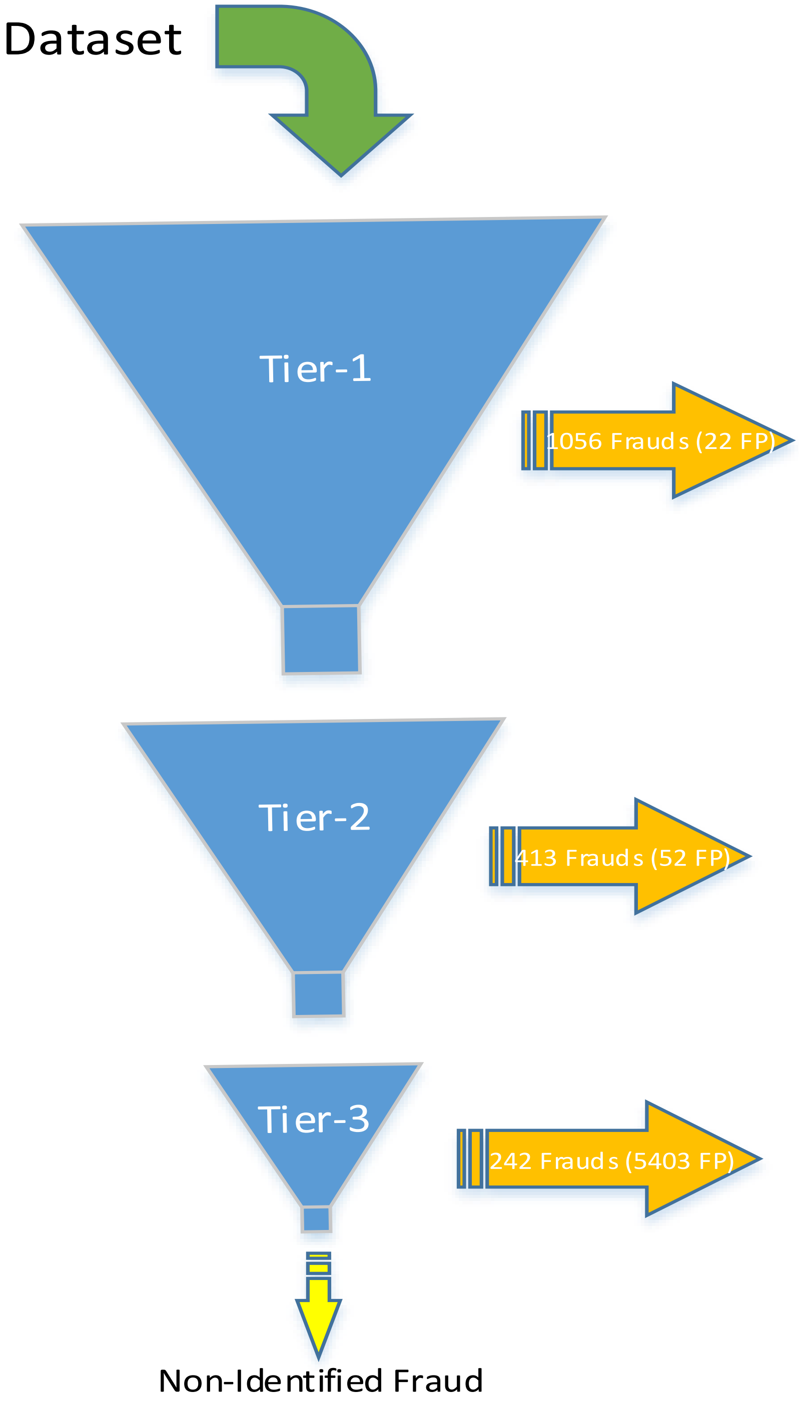

Figure 1, filtering of fraudulent transactions using a tiered model

CHAPTER VII

PCA embedded Random Forest (RF-PCA)

7.1) Architecture of RF-PCA

PCA and Random Forest are combined through to the creation of a PCA model within each tree. The internal PCA model is trained during the random forest training phase. This model then remains assigned to its tree and is used to transform the test data. Although PCA is unsupervised and could be trained on test data, it is not. Not retraining the PCA model ensures the forest features have been trained on are the same as the test set features. The new PCA features contain information from more than one feature allowing for new patterns to be classified. These new features also create a less correlated forest as each tree has new and unique features. Uncorrelated trees have been shown to increase the accuracy of the random forest [10]. Below in figure 2 is the training architecture for the RF-PCA. Decision trees are created in the same way they are made in a traditional random forest described in chapter 2. The RF-PCA is unique in the way new features are created and stored. Creating new features using the entire dataset was not successful resulted in a higher precision but overall lower f-score. The precision was increased while the recall was substantially less than from an f-score optimized model. We see the differences below in Tables 9 and 10.

	Predicted Fraud	0	1
True Fraud			
0		1396252	353
1		235	1625

Table 9, The f-score optimized model tested using original data.

	Predicted Fraud	0	1
True Fraud			
0		1396438	167
1		702	1158

Table 10, Three PCA generated features added to the dataset before training.

We can see from 9 and 10 that PCA features did add information to the model but ultimately inhibited the model recall. The RF-PCA was made to ensure that the new PCA features did not increase the correlation between trees. In Table 10, the features m used to create new PCA features are randomly selected. The number of original features used to create our PCA features was optimized to find the best solution. Utilizing feature randomness in creating PCA features allows each tree to be making classification on unique patterns. Feature randomness resulted in a marginal increase in performance over a traditional random forest.

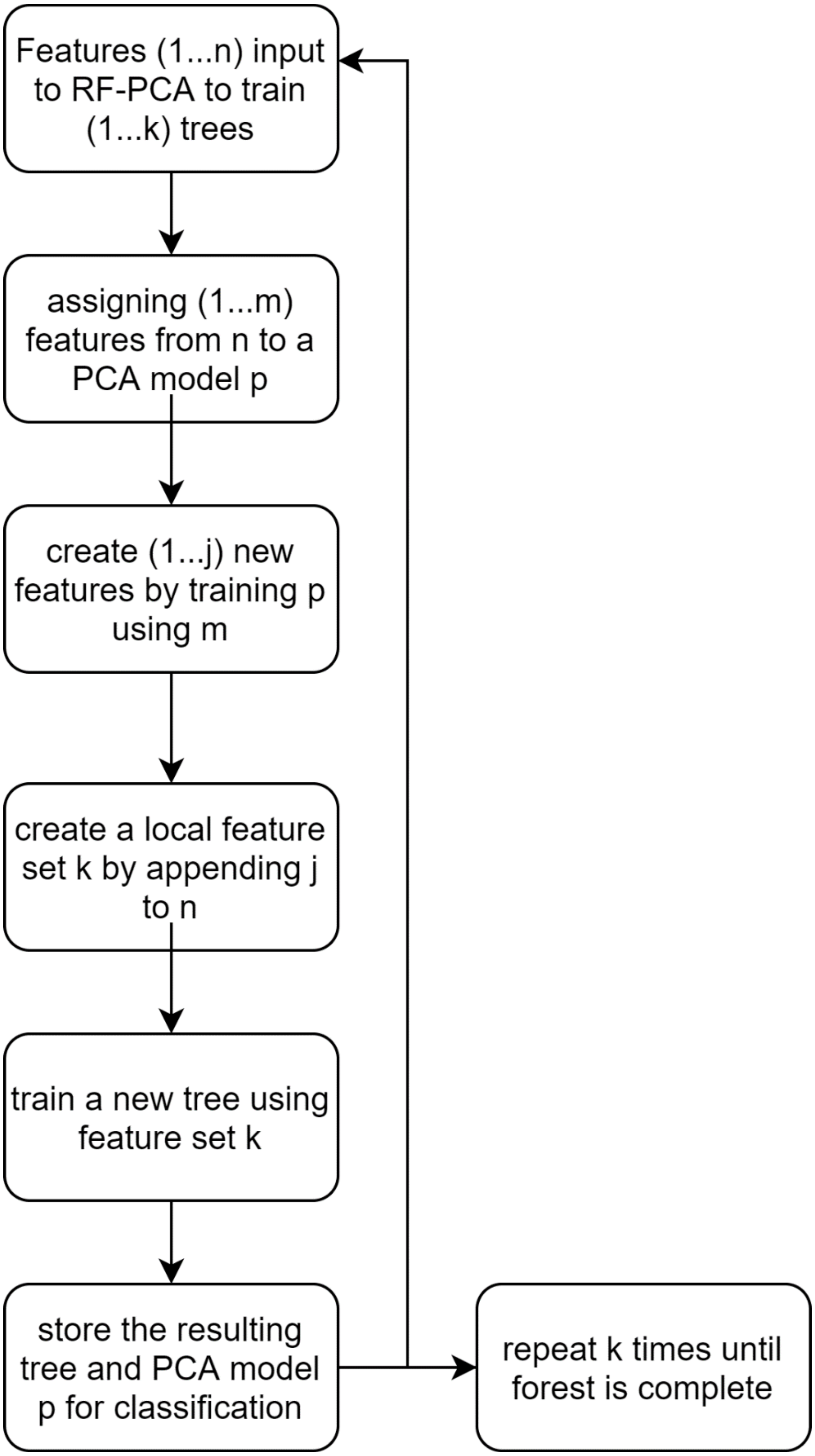
Features (1...n) input to RF-PCA to train (1...k) trees
assigning (1...m) features from n to a PCA model p
create (1...j) new features by training p using m
create a local feature set k by appending j to n
train a new tree using feature set k
store the resulting tree and PCA model p for classification
repeat k times until forest is complete

Figure 2: Training architecture for RF-PCA

7.2) Results of RF-PCA

RF-PCA was tested by optimizing both the standard random forest and RF-PCA and optimized using a randomized grid search and an evolutionary algorithm. These two approaches were primarily the same, with maximum f1 scores of .881 and .883 using a traditional random forest. Using two optimization strategies adds confidence that an optimal parameter set is found. RF-PCA was able to produce an f1-score of .895, showing a 1.2% increase in performance. Our increased performance is small but does show the potential of RF-PCA. RF-PCA models were consistently more accurate. Through giving each tree its own unique PCA feature, new information was able to reveal to the model without creating a bias towards the new features. When PCA features were extracted from the entire feature set and given to every tree, precision increased while the overall f1 score went down. The increased precision shows that some useful patterns were being overshadowed by the new PCA features resulting in a low recall. Using RF-PCA, the forest became less uncorrelated because each tree was trained on slightly different datasets, resulting in increased recall and precision. Figure 3 compares the performance of each model created in the optimization of RF-PCA and the standard random forest. RF-PCA has a higher variability and, on average, a lower performance but demonstrates a higher maximum performance. The variability of RF-PCA's performance makes it essential to optimize the RF-PCA thoroughly. Table 11 shows the results of the best model from each optimization.

	Standard Random Forest	RF-PCA
f-score optimized over 50 forests	.883	.895

Table 11, The results of the best model in the optimization of 50 models

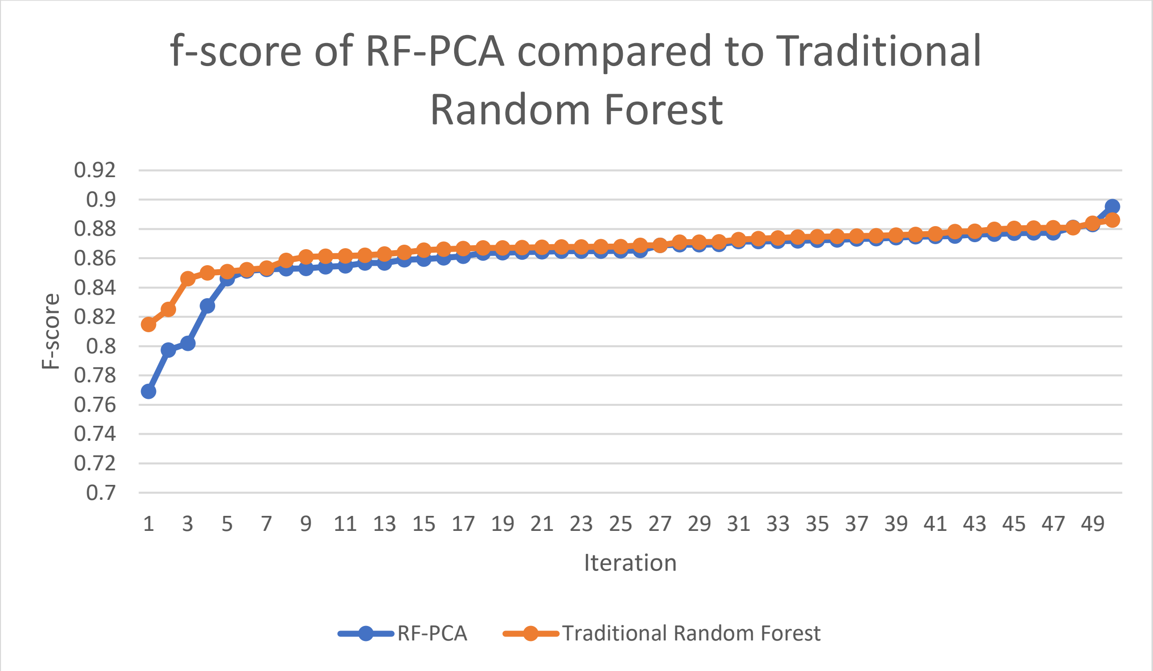

Figure 3, optimization of 50 models using PCA-RF and traditional RF.

Due to the two models' close performances, the parameters used to create the highest performance RF-PCA model were used with a regular random forest model. The results of this test are shown in Tables 12 and 13. This test was done using a new random sample of fraud transactions. We can see below in Tables 12 and 13 that the performance using the best feature set was nearly the same, with RF-PCA having a higher precision and the standard approach having a higher recall. For this reason, RF-PCA is best suited for when a model needed to be precision focused.

	Predicted Fraud	0	1
True Fraud			
0		1,397,904	119
1		274	1621

Table 12, RF-PCA tested using the best-found parameters.

	Predicted Fraud	0	1
True Fraud			
0		1,397,904	141
1		254	1641

Table 13, Standard random forest tested using best RF-PCA parameters.

Twenty new models were optimized and compared based on precision scores shown in figure 4. We can see that RF-PCA consistently produces higher precision when compared to a standard random forest. This high precision could be useful when creating the precision stratified random forest presented in chapter 6. RF-PCA provided an option for models with favor high precision over high recall; this can be useful when using the assembly approaches described previously in this study.

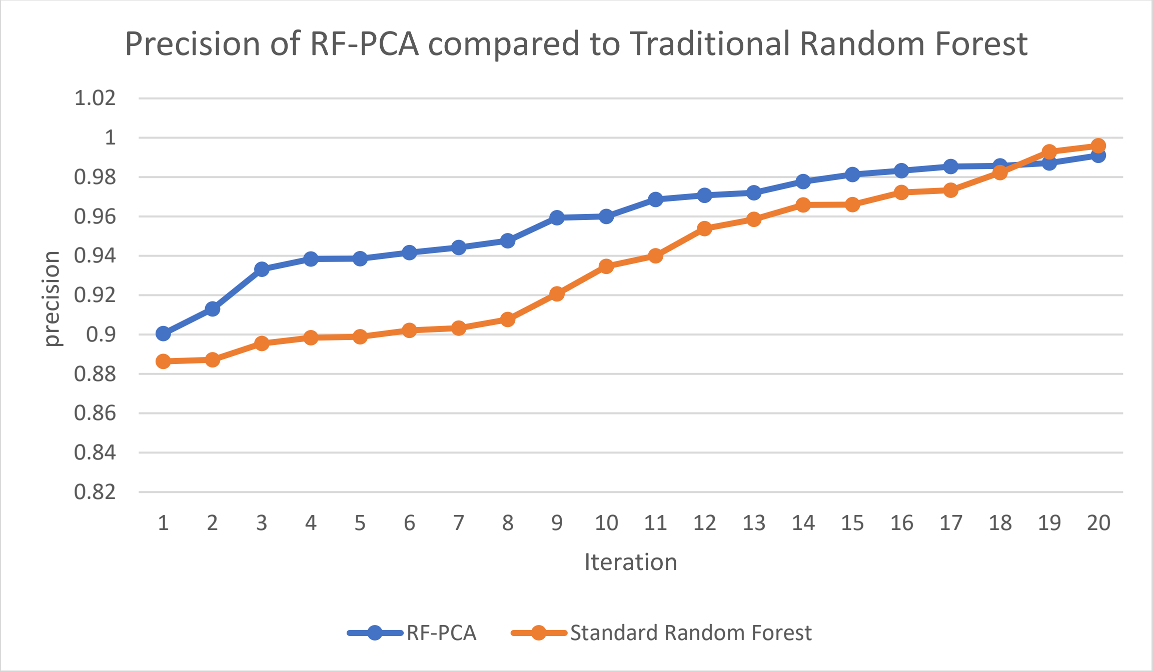

Figure 4, RF-PCA precision compared to standard random forest.

CHAPTER VIII

Conclusion

In this paper, two novel approaches were presented to increase classification accuracy for imbalanced financial datasets. The precision stratified random forest produced a 13% higher recall while more accurately assessing classification precision. The increase in recall represents a significant increase over the performance of a standard random forest. The assembly process used was able to show lower precision models almost always found transactions that were found by higher precision models. This information allowed for creating an assembly that could separate high precision transactions from low precision transactions. Utilizing this approach could protect many consumers from financial fraud while also providing the information necessary to make an appropriate response to potential fraud.

PCA-RF provides a new architecture for random forest focused on allowing the random forest algorithm to detect patterns found between interdepended features. The most considerable advantage to the RF-PCA is its consistently high precision, which comes at a tradeoff to recall. PCA-RF has potential problems as its average model did not outperform a traditional random forest during the optimization process. We have shown that depending on the number of features used in making new PCA features, the overall performance can be decreased while precision is increased. Finding the right balance requires optimization and will be different for every dataset, but PCA-RF did outperform a standard RF once optimum features were found in some aspects.

References

[1] Lothson, Anna. "The State of Card Fraud - and the Impact on Financial Institutions." *The State of Card Fraud - and the Impact on Financial Institutions*, RippleShot, 25 Aug. 2017, info.rippleshot.com/blog/card-fraud-impact-financial-institutions.

[2] West, J., & Bhattacharya, M. (2015, September 5). Intelligent financial fraud detection: A comprehensive review, Computers & Security, Volume 57, page 47-66. Retrieved September 10, 2019, from www.sciencedirect.com https://doi.org/10.1016/j.cose.2015.09.005.

[3] Chen, Chao, et al. "Using Random Forest to Learn Imbalanced Data." *Using Random Forest to Learn Imbalanced Data*, statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

[4] Stolfo S, et al. Credit card fraud detection using meta-learning. Issues and initial results. AAA-97 workshop on fraud detection and risk management. 1997

[5] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. ISBN 978-0-412-04841-8.

[6] Maes, Sam, et al. "Credit card fraud detection using Bayesian and neural networks." Proceedings of the 1st international naiso congress on neuro fuzzy technologies. 2002.

[7] West, J., & Bhattacharya, M. (2015, September 5). Intelligent financial fraud detection: A comprehensive review, Computers & Security, Volume 57, page 47-66. Retrieved

September 10, 2019, from www.sciencedirect.com https://doi.org/10.1016/j.cose.2015.09.005.

[8] Raj V., Magg S., Wermter S. (2016) Towards Effective Classification of Imbalanced Data with Convolutional Neural Networks. In: Schwenker F., Abbas H., El Gayar N., Trentin E. (eds) Artificial Neural Networks in Pattern Recognition. ANNPR 2016. Lecture Notes in Computer Science, vol 9896. Springer, Cham. https://doi.org/10.1007/978-3-319-46182-3_13

[9] Breiman, L. 2001. Random Forests. Machine Learning, Vol. 45 Issue 1, pp. 5-32.

[10] S.Bharathidason, C.Venkataeswaran, Improving Classification Accuracy based on Random Forest Model with Uncorrelated High Performing Trees, International Journal of Computer Applications (0975 – 8887) Volume 101– No.13, September 2014

[11] Bhattacharyya, S., Jha, S.K., Tharakunnel, K.K., & Westland, J.C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50, 602-613.

[12] Chen, Chao, et al. "Using Random Forest to Learn Imbalanced Data." Using Random Forest to Learn Imbalanced Data, statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf. 2004.

[13] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2.1-3 (1987): 37-52.

[14] E. A. Lopez-Rojas , A. Elmir, and S. Axelsson. "PaySim: A financial mobile money simulator for fraud detection". In: The 28th European Modeling and Simulation Symposium-EMSS, Larnaca, Cyprus. 2016

[15] Campus, K. (2018). Credit card fraud detection using machine learning models and collating machine learning models. International Journal of Pure and Applied Mathematics, 118(20), 825-838.

[16] Abbasi, A., Albrecht, C., Vance, A., & Hansen, J. (2012). Metafraud: a meta-learning framework for detecting financial fraud. Mis Quarterly, 1293-1327.

[17] Liu, C., Chan, Y., Alam Kazmi, S. H., & Fu, H. (2015). Financial fraud detection model: Based on random forest. International journal of economics and finance, 7(7).

[18] Pawar, Amruta D., Prakash N. Kalavadekar, and Swapnali N. Tambe. "A survey on outlier detection techniques for credit card fraud detection." *IOSR Journal of Computer Engineering* 16.2 (2014): 44-48.

[19] Rosipal, Roman, et al. "Kernel PCA for feature extraction and de-noising in nonlinear regression." *Neural Computing & Applications* 10.3 (2001): 231-243.

[20] Mironiuc, Marilena, Ioan-Bogdan Robu, and Mihaela-Alina Robu. "The fraud auditing: Empirical study concerning the identification of the financial dimensions of fraud." *Journal of Accounting and Auditing* 2012 (2012): 1.