COMP9417 Machine Learning Project

IEEE-CIS Fraud Detection

Alex Lee (z5207126)

School of Computer Science and Engineering
UNSW Sydney

July 2021

1 Introduction

The development of the internet and technologies has seen an exponential increase in e-commerce activity as our world becomes progressively interconnected. However, with this ever-increasing growth and activity in e-commerce, there has been a proliferation of fraudlent transactions in which sensitive financial information is obtained through criminal efforts. Although businesses have conventional systems in place to detect fraudulent activity, these systems are often outdated, inconvenient and inefficient. Thus, we propose a data driven machine learning approach to improve fraud detection and consequently prevent fraudulent transactions in the e-commerce industry.

The Kaggle competition IEEE-CIS Fraud Detection (2019) provides an opportunity to apply machine learning in this context. The purpose of this challenge was to predict the probability that an online transaction was fraudulent and thus correctly classify whether a transaction was fraudulent or not. Previous work on this topic has been limited to identifying if transactions were fraudulent using the best model; my approach will utilize not necessarily the best machine learning model but will explore how we can implement Random Forest specifically with hyperparameter tuning to achieve the best results. The data was provided by the Vesta Corporation through a Kaggle competition containing both training and testing sets.

This project is aimed at exploring how we can use supervised machine learning in classifying fraudulent transactions utilizing the Random Forest model. The scope of this problem is binary classification using the ROC AUC as the evaluation metric as specified in the rules of the Kaggle competition.

2 Related Work

R. Jhangiani, D. Bein & A. Verma (2019) proposes multiple supervised machine learning models to predict the probability of a transaction being fraudulent. The performances of algorithms such as Random Forest, Support Vector Machine, Gradient Boost and combinations of these are compared in terms of how well they are suited for this task. This report identifies issues such as concept drift and class imbalance providing potential solutions. They suggest that machine learning algorithms like Random Forest, Gradient Boost and AdaBoost are good choices for this task whereas other algorithms such as Support Vector Machine and Logistic Regression by themselves are not preferred for this task due to their poor performance in unbalanced binary classification problems[1].

G. E. Melo-Acosta, F. Duitama-Muñoz & J. D. Arias-Londoño (2017) is based on a Balanced Random Forest approach that can be used in supervised and semi-supervised settings based on a co-training approach. This report suggests that a co-trained Balanced Random Forest Approach has better performance compared to the standard Spark Random Forest model[2].

A. Saputra & Suharjito (2019) reviews suitable machine learning algorithms and subsampling techniques for fraud detection. This report proposes the use of Synthetic Minority Oversampling Technique (SMOTE) in order to balance the data along with Principal Component Analysis to extract features from high dimensional data. It is concluded that Random Forest is the most suitable algorithm for this task with suitable pre-processing[3].

3 Materials and Methods

3.1 Software

I have chosen to use python3 and a multitude of libraries to do my analysis on. These libraries include:

pandas, numpy, imblearn, collections.

3.2 Data Description

Kaggle provided 4 csv files (test_identity, test_transaction, train_identity, train_transaction) and a sample submission file (sample_submission.csv).

	test_identity.csv	***test_transaction.csv***	***train_identity.csv***	***train_transaction.csv***
Format	csv	csv	csv	csv
Size	25.8 megabytes 41 variables 141,908 observations	613.5 megabytes 393 variables 506,692 observations	26.5 megabytes 41 variables 144,234 observations	684.5 megabytes 394 variables 590,541 observations
Variables	TransactionID id-01 ~ id-38 DeviceType DeviceInfo	TransactionID TransactionDT TransactionAMT ProductCD card1 ~ card6 addr1, addr2 dist1, dist2 P_emaildomain R_emaildomain C1 ~ C14 D1 ~ D15 M1 ~ M9 V1 ~ V339	TransactionID id_01 ~ id_38 DeviceType DeviceInfo	TransactionID TransactionDT TransactionAMT ProductCD card1 ~ card6 addr1, addr2 dist1, dist2 P_emaildomain R_emaildomain C1 ~ C14 D1 ~ D15 M1 ~ M9 V1 ~ V339 isFraud
Data Types	Numeric Text	Numeric Text Datetime	Numeric Text	Numeric Text Datetime

3.3 Pre-Processing

We begin by importing the relevant libraries for pre-processing (numpy and pandas), naming these tables *train_transaction_df*, *train_identity_df*, *test_transaction_df*, *test_identity_df* respectively. The next step is to merge these two sets of tables together into *train_df* and *test_df* by "TransactionID". We reduced memory usage by down casting numeric features in the train and test data accordingly via the function mem_reduce(). For categorical features, we use label encoding to map each category to a number and also replace all missing values for numeric features with their respective medians. After completing all these steps for pre-processing, we export two data frames *train_reduced_memory* and *test_reduced_memory* to csv for further use[6].

3.4 Exploratory Data Analysis

3.4.1 Data Imbalance and Resampling

Imbalanced datasets describe data in which there is a disproportionate ratio of observations in each class. This is a problem for classification and predictive modelling as most machine learning models were designed with the assumption of equal data. Therefore, strong data imbalance will result in poor predictive performance of the Random Forest model as decision trees are sensitive to class imbalance[5].

Nitesh Chawla, et al. In their research paper "*SMOTE: Synthetic Minority Oversampling Technique*", suggests that a combination of SMOTE with random under sampling of the majority class can solve data imbalance issues and achieve better classification performance in ROC space[9].

SMOTE essentially selects observations that are close in feature space and then draws a line between these observations, synthetically creating a new example from a point along this line. More specifically, a random observation is selected from the minority class. The k nearest neighbours are found for that example in which one of these neighbours are randomly chosen to create a synthetic observation between the two points in feature space[6]. (see 3.4.1.1 below[10])

Class Imbalance vs SMOTE Graphical Representation

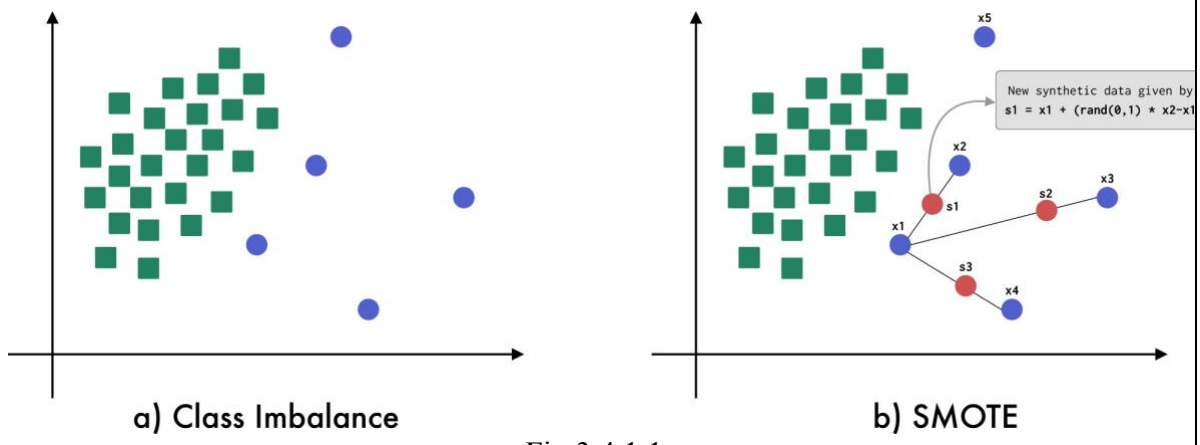

Fig 3.4.1.1

Random under sampling is a process whereby we randomly select examples from the majority class and delete them from the training dataset.

From initial exploratory analysis we identified a strong data imbalance in our training set with the minority class (isFraud = 1) only accounted for 3.5% of all observations (fig 3.4.1.2). Thus, we used SMOTE coupled with random under sampling to reduce the size of the majority class (isFraud = 0) and additionally increase the size of the minority class (fig 3.4.1.3) in the training data. We first oversampled the minority class to have 10% the number of observations in the majority class (56,987 observations), then we used random under sampling to reduce the number of observations in the majority class such that it has 50% more observations than the minority class (113,974 observations)[6].

figure 3.4.1.2

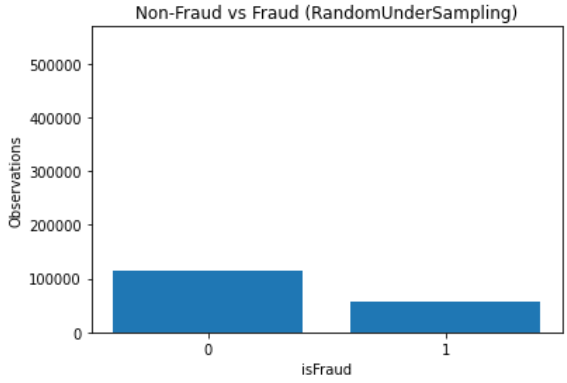

figure 3.4.1.3

3.4.2 Feature Selection

After re-sampling, we explored the 433 input variables in our dataset and their correlations to each other. Although the Random Forest model contains built-in feature selection, we implemented feature selection separately in order to reduce the computational cost of modelling and subsequently improve the performance of our model[7].

The approach was to create correlation matrices for subsets of features that we aimed to reduce and then remove one of each pairs of features that had a correlation above 0.9. We explored groups of features rather than the whole dataset in order to replace each group with a subset to reduce the amount of information lost. The groups that we reduced are as follows: V1~V339, id_01 ~ id_38, D1 ~ D15, C1 ~ C14, M1~ M9.

The correlation matrices and the information on the reduced groups are shown below. After removing redundant features, our final dataset contained 259 input variables. Note that we also removed the same redundant features for our test dataset.

V1 ~ V339	id_01 ~ id_38	D1 ~ D15	C1 ~ C14	M1~ M9
Removed: 151 Retained: 181 Fig 3.4.2.1	Removed: 7 Retained: 31 Fig 3.4.2.2	Removed: 1 Retained: 14 Fig 3.4.2.3	Removed: 11 Retained: 4 Fig 3.4.2.4	Removed: 4 Retained: 5 Fig 3.4.2.5

Correlation Matrix V1~V339

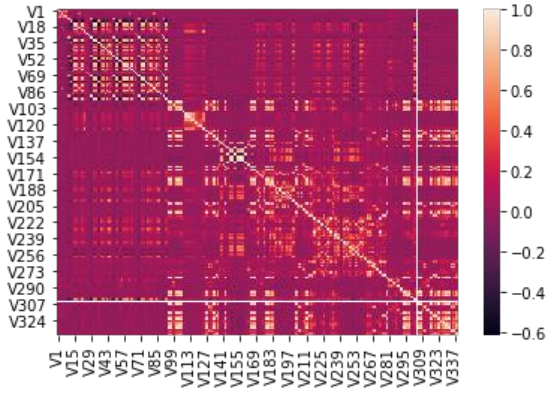

figure 3.4.2.1

Correlation Matrix id_01~id_38

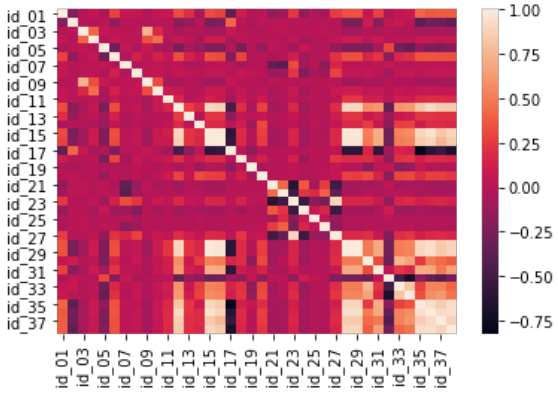

figure 3.4.2.2

Correlation Matrix D1~D15

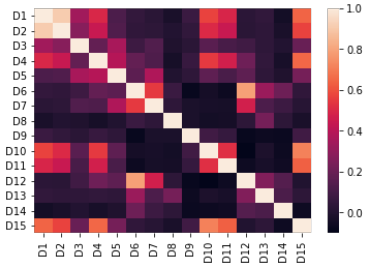

figure 3.4.2.3

Correlation Matrix C1~C14

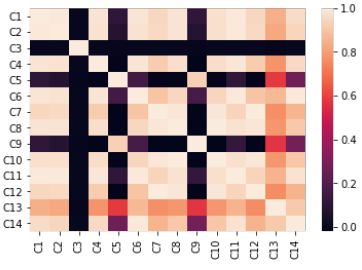

figure 3.4.2.4

Correlation Matrix M1~M9

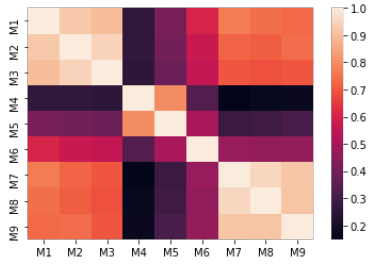

figure 3.4.2.5

3.4.3 Validation Split

A validation set is separate from the test set which is used to evaluate the performance of models and their hyperparameters whilst also identifying any overfitting. We created the validation set from the reduced training data by randomly selecting 33% of the data.

3.5 Implementation

3.5.1 Random Forest Model

The Random Forest machine learning classification model utilizes a large number of individual decision trees as an ensemble. Each decision tree within the random forest output a class prediction in which all the outputs of the decision trees are aggregated. The class with the most votes becomes the random forest model's prediction (see fig 3.5.1.1). Since random forests use ensemble predictions, this model is more likely to be robust, un-biased and accurate in comparison to a single decision tree implementation. Note that since random forests handle a large number of decision trees, it requires more processing power and thus takes more time for computation[11].

Random Forest Graphical Representation

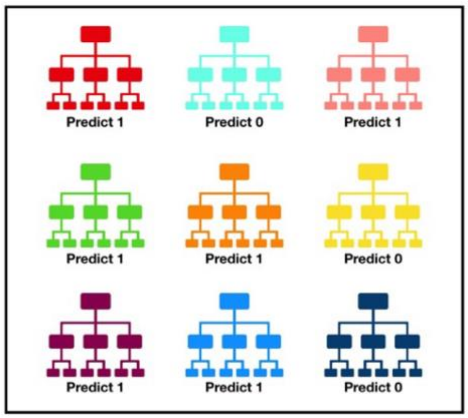

fig 3.5.1.1

3.5.2 Model Evaluation Metric

The roc_auc_score calculates the area under the curve (AUC) of the receiver characteristic operator (ROC). The ROC is a probability curve used for binary classification tasks plotting the True Positive Rates against the False Positive Rates at various threshold values. The AUC is the measure of how well a classifier can predict classes correctly. Therefore, the higher the AUC, the better the performance of the model[12].

We used the random forest classifier from sklearn to fit and train our model. We evaluated our model performance initially by predicting on our validation set using the roc_auc_score as our evaluation metric as directed by Kaggle.

3.5.3 Hyperparameter Tuning

Machine learning models such as random forest have a multitude of hyperparameters that can be tuned to improve model performance. There are countless numbers of combinations that you can set your model with however, there's a point where it is most optimal. The goal of this section is to determine the most optimal hyperparameters for our random forest model using scikit-learn tools from python.

Random forest has 19 hyperparameters that you can change, each with multiple options. We will be looking at 6 parameters that have the greatest impact on our model performance. Refer to the table below containing these 6 settings and a description of each.

Hyperparameters	**Description**
n_estimators	The number of trees in the forest
max_features	The maximum number of features considered for splitting a node
max_depth	The maximum number of levels in each decision tree
min_samples_split	The minimum number of samples placed in a node before the node is split
min_samples_leaf	The minimum number of samples allowed in a leaf node
bootstrap	The method for sampling datapoints (with or without replacement)

Due to the dimensionality of our data and the time complexities of building random forest models on such data, it is unfeasible to determine the best hyperparameters as computation will take too long to check every possible combination. Thus, we only tune a handful of parameters and pick the best combination out of a random selection of 10 different combinations.

K-fold cross validation is a method whereby we split the training data into K subsets or folds of equal size (K = 2, 3, ..., N). A machine learning model is then fit K times by training the model on the remaining K – 1 folds and validating that model with the Kth fold. Refer to fig 3.5.3.1 for a graphical representation of cross validation where there are 5 folds and thus a model is fitted 5 times and evaluated at each time.

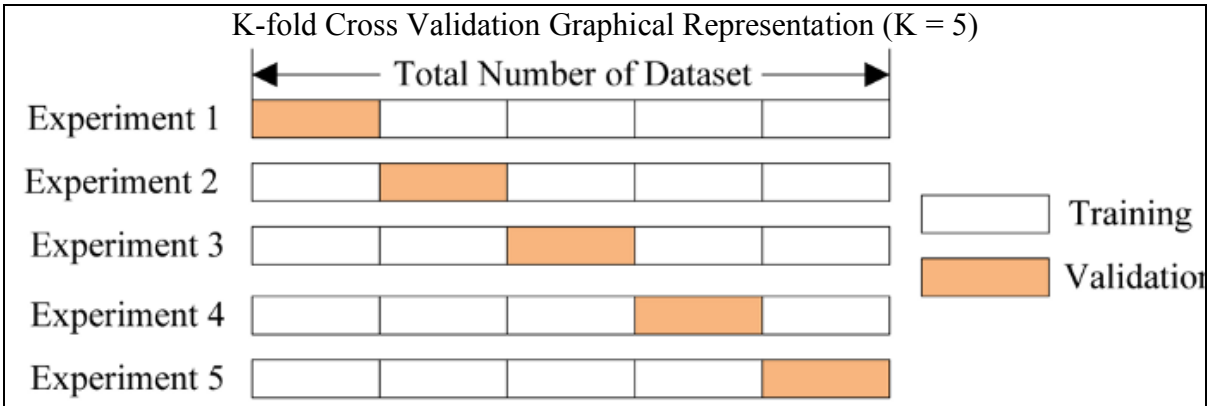

Fig 3.5.3.1

For each combination of hyperparameter settings, we use K-fold cross validation to compare average model performance on each of the folds. After comparing all of the models, we select the best one to train on the whole training set and evaluate on the validation set that we created previously[13].

4 Results

Due to initial pre-processing, we were able to reduce memory usage by 1.4 GB for our training dataset and by 1.2 GB for our testing dataset. We then resampled the training data using SMOTE combined with Random Under Sampling to alleviate the data imbalance. Fraudulent data points which only accounted for 3.5% of the whole dataset now accounted for 50% of the dataset. Due to the high dimensionality of our data, we utilized feature selection to remove any redundant input variables using correlation. This resulted in the reduction of our dimensions from 393 input features to 259. Additionally, we further split this data using 33% of the training set as the validation set. Once we had our final datasets, we were able to reduce memory usage again for our training and validation sets.

In order to optimize our model performance, we utilized K-fold cross validation for hyperparameter tuning, using the ROC_AUC_SCORE as the evaluation metric. According to our search, the optimal hyperparameters were at these settings:

Hyperparameters	**Optimal settings**
n_estimators	= 400
max_features	= auto
max_depth	= 80
min_samples_split	= 2
min_samples_leaf	= 4
bootstrap	= False

This model provided us with an accuracy on the validation set of 99.99%.

5 Conclusion

Therefore, we can see that supervised machine learning algorithms such as random forest is suitable for this task of credit card fraud classification as reflected by our performance scores. SMOTE and random under sampling was effective in mitigating the issues of data imbalance and memory usage reduction was essential for improving processing speeds. Not all features were important for our model with a handful of input features being redundant for our model. There were a few drawbacks, namely being the inability to exhaustively search through every combination of hyperparameter settings for the random forest model due to computer processing capacity issues. Additionally, the re-sampling process also took a significant amount of time. Furthermore, our model may be sensitive to concept drift whereby consumer transaction patterns change over time resulting in poor predictive performance in the future as our predictive model assumes a static relationship between the variables. In order to address concept drift, this model requires an update with more recent historical data periodically at each month or year[14].

6 References

[1] R. Jhangiani, D. Bein and A. Verma, Machine Learning Pipeline for Fraud Detection and Prevention in E-Commerce Transactions, *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019

[2] G. E. Melo-Acosta, F. Duitama-Muñoz and J. D. Arias-Londoño, Fraud detection in big data using supervised and semi-supervised learning techniques, *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2017

[3] A. Saputra, Suharjito, Fraud Detection using Machine Learning in e-Commerce, *International Journal of Advanced Computer Science And Applications (IJASCA)*, 2019

[4] S.Dogga, THE ROLE OF RANDOM FOREST IN CREDIT CARD FRAUD ANALYSIS, *Bournemouth University*, 2020

[5] J.Brownlee, *A Gentle Introduction to Imbalanced Classification*, *Machine Learning Mastery*, 2019, Last updated 2020

[6] J.Brownlee, SMOTE for Imbalanced Classification with Python, *Machine Learning Mastery*, 2020, Last updated 2021

[7] L.Ferreira, Extensive EDA and Modelling, *Kaggle*, 2019

[8] R.Vishal, Feature Selection – Correflation and P-value, *towards data science*, 2019

[9] N.Chawla, K.Bowyer, Lawrence.Hall, Philip.Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research 16*, 2002

[10] F.Lopez, SMOTE: Synthetic Data Augmentation for Tabular Data, *towards data science*, 2021

[11] T.Yiu, Understanding Random Forest, *towards data science*, 2019

[12] A.Bhandari, AUC-ROC Curve in Machine Learning Clearly Explained, *Analytics Vidhya*, 2020

[13] W.Koehrson, Hyperparameter Tuning the Random Forest in Python, towards data science, 2018

[14] J.Brownlee, A Gentle Introduction to Concept Drift in Machine Learning, *Machine Learning Mastery*, 2017, Last Updated 2020