**CPS 803 Project Report**

**Credit Card Fraud Detection**

Khanh Linh Le

500895991

CPS803-F21

*Computer Science*

*Ryerson University*

*Toronto, Ontario*

Bradley Pahati

500829758

CPS803-F21

*Computer Science*

*Ryerson University* *Toronto, Ontario*

Viet Anh Phung

500895992

CPS803-F21

*Computer Science*

*Ryerson University*

*Toronto, Ontario*

Tusaif Azmat

500660278

CPS803-F21

*Computer Science*

*Ryerson University*

*Toronto, Ontario*

**Abstract**

**As the trend is moving towards digitization and cashless transactions, the need for credit cards usage also increases rapidly. Credit card fraud thus becomes a significant concern for both users and financial institutions around the world. To tackle this problem, the application of machine learning techniques is considered to be very useful in helping detect fraudulent transactions. Specifically, we apply seven different machine learning methodologies in this project: K-nearest neighbors, decision tree, support vector machine, logistic regression, naive Bayes, random forest, and neural network. We also further analyze the difference among these methodologies by looking at the logic behind each method and their performance results. We then conclude the paper with the technique that has the highest accuracy rate and lowest false negative rate.**

**I. Introduction**

*A. Overview*

Worldwide, payment fraud has reached $32.39 billion in 2020, tripled from $9.84 billion in 2011 [1]. With an increasing demand for credit card usage, also known as “buy now pay later,” the number of fraudulent transactions will continue to surge in the near future if institutions do not apply an effective fraud detection system.

An effective fraud prevention and protection plan should avoid thefts from executing fraudulent transactions while not preventing real card owners’ transactions. Having a high false positive rate could lead to negative customer experience, which could affect the credibility of the institution. Thus, it is critical to have a fraud detection system in which the true negative number is high and false negative number is low.

The rest of this paper is structured as follows: Section II discusses the problem statement and an overview of the dataset that we use in this project. Section III describes all the methods and models we apply in this experiment, after which the results and discussion are presented. Section IV briefly explains how we implement the code. Finally, a list of references are listed in Section V.

*B. Significance and Challenge*

Due to insufficient public credit card transaction databases, limited computer power, and limited time frame to run this experiment, we are only able to run this experiment on a dataset consisting of 284,807 transactions with a total of 31 features. In addition, because of confidentiality issues, most features are not in their original form but have been transformed using principal component analysis (PCA). Based on a realistic scenario, the fraud ratio in this dataset is minuscule - presenting only 0.172% of all transactions.

*C. Prior Work*

We choose this project from the dataset provided on Kaggle[1] website. There is some work done but we will use this dataset and apply the machine learning algorithms to further enhance the accuracy of our predictions.

**II. Problem Statement and Dataset**

*A. Dataset*

The credit card fraud dataset used in this paper is obtainable from Kaggle.com and contains a subset of online European credit card transactions made in September 2013 over a period of two days, consisting of a highly imbalanced 492 frauds out of 284 807 total transactions.

For confidentiality the dataset is simply provided as 28 unlabeled columns resulting from a PCA transformation. Additionally, there are three labeled columns: Class, Time, and Amount. The data dimension we use in this experiment is [Time, V1, V2 … V28, Amount, Class].

Note that the dataset is highly imbalanced towards legitimate transactions which have the label of “0”. Following is the result from running “*data.Class.value\_counts()”*:

0 284315

1 492

*B. Problem Statement*

The objective for this project is to build machine learning models to classify or identify fraudulent card transactions from a given set of card transaction data. We want to predict if the transaction is fraudulent based on the input attributes and identify it as fraudulent or genuine.

Each transaction is labeled either fraudulent or not fraudulent. Note that prevalence of fraudulent transactions is very low in the dataset. Less than 0.1% of the card transactions are fraudulent. This means that a system predicting each transaction to be normal can reach an accuracy of over 99.9% despite not detecting any fraudulent transaction. This will necessitate adjustment techniques.

We use all the techniques which we had learned during the course of machine learning. We had applied the supervised learning techniques for the classification algorithms. Following are the algorithms we use to predict the class results:

1. Naives Bayes classifier
2. Decision Tree.
3. Random Forest.
4. K-Nearest Neighbors (KNN).
5. Logistic Regression.
6. Support Vector Machine (SWM)
7. Neural Networks.

We use the above machine learning algorithms to predict using both predictors and target values.

**III. Methods and Models**

We will start our project with data collection and for that we will use the dataset provided to us through Kaggle[1] dataset. After downloading the dataset we will be performing some functions such as preprocessing, analysis and evaluation based on our experiments. We will be performing the following methods:

1. ***Methods***

*1). Imbalance Learning:*

Standard decision trees use information gain as the splitting criterion for learning which results in rules biased towards the majority. Research also shows that imbalanced datasets pose a problem for kNN, neural networks (NN), and support vector machines (SVM). This problem is most pronounced when the two classes overlap as in the case of the Kaggle dataset; the majority of machine intelligence algorithms are not suited to handle both unbalanced and overlapped class distributions.

Fortunately, particular algorithms exist that can take class imbalance into account. In addition there are techniques at the data level and algorithm level which can reduce the negative effects of these biases.

*2). Concept Drift:*

Credit card fraud is prone to concept drift as consumer trends change due to changing preferences, seasonality and new products, as well as evolving fraud attack strategies [2]. The net effect of this is that the statistical properties of the underlying data change over time. Recent research has shown that it is possible to overcome this while still maintaining conventional machine intelligence techniques. For example, sliding windows approach where a classifier is trained everyday on the most recent samples, or an ensemble approach where the oldest component is replaced with a new classifier [2].

However, for the purposes of this paper the challenges associated with concept drift, as well as their resolutions, are not explored for two reasons. The first is that the dataset used is collected over a period of two days, which may not be sufficient for concept drift to actually take place. The second reason is that as mentioned, research shows that concept drift in FDS may be appropriately tackled by using conventional methods that are employed to maintain only a local temporal memory of learned attributes [2]. In other words, once a method is found that performs well for short periods of time,

i.e. sufficiently small periods of time where concept drift does not take place, its implementation can then be further improved to account for concept drift. Therefore the fraud detection methods explored in this paper would need further refinement before being applied to a data stream of longer than a handful of days. These refinements are discussed in more detail at the end of the paper.

*3). Sampling:*

Sampling methods are used to compensate for the un-balanceness of the dataset by reducing the classes to near equivalence in size. We use the oversampling technique which uses a bias to achieve this purpose. Complex algorithms such as synthetic minority oversampling technique (SMOTE) actually create new data points based on known samples and their features instead of simply replicating the minority class [2]. However this algorithm relies on assumption of the minority class and is generally computationally expensive. Particularly, the created data generally is an interpolation of prior data which may not actually provide a realistic approximation of if the classes were in fact, balanced. Despite this, sampling methods can provide a more robust approach to imbalance learning than other methods, for example cost-based techniques which penalize errors differently depending on class such that the minority class is favored [2]. For example, what is the cost to use?

Finally, the results will be compared to cost-based balancing methods, if applicable.

*4). Model Training:*

In this project, we implemented our classification algorithms both by using scratch coding and also a Python machine learning library called Scikit-learn. Scikit-learn is considered to be efficient, useful, robust, and consistent [4].

*5). Pre-Processing and Data Normalization*

The dataset we use in this project is taken from a Kaggle provided dataset that has been collected and analyzed during a research collaboration of Worldline and Université Libre de Bruxelles [5]. The dataset includes a total of 284,807 credit card transactions in which 492 transactions are fraudulent, which represents a fraud rate of 0.172%. There are 31 features in this dataset where features V1, V2, … V28 are transformed principal components obtained from PCA transformation. Features ‘Time’ and ‘Amount’ are kept original and they refer to the number of seconds elapsed between each transaction and the money amount paid in each transaction respectively. Lastly, feature ‘Class’ takes value 1 if it is a fraud and 0 otherwise.  In this experiment, we drop the time feature column.

*6). Dataset Split Strategy*

We will use the Holdout method where 70% sample size will be used for training and 30% (or 85443 samples) will be tested on models. Since the whole data set only contains 495 fraudulent transactions, which accounts for only 0.17% of all 284807 samples, the split will be stratified on the response variable.

*7). Feature Extraction*

We will use default parameters for the learning curve of the model and will also use the process of adjusting hyperparameters and the feature extraction methods to find the best fit for our model.

In a dataset the features V1, V2, to V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. Feature 'Class' is the target variable with value 1 in case of fraud and 0 otherwise.

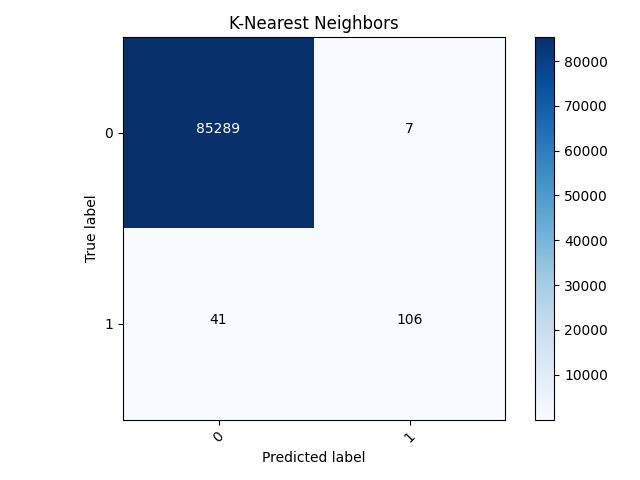
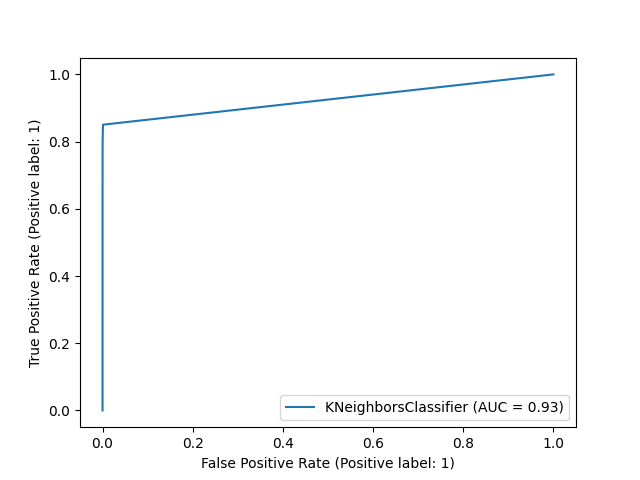
1. ***Models (Classifications)***

Most Fraud Detection Systems (FDS) use supervised classification techniques to discriminate between fraudulent and legitimate transactions. Research on similar credit-card fraud datasets shows Random Forest (RF) having superior performance than NN and SVM when using oversampling to correct for class imbalance [3]. This finding is verified by exploring three general classes of techniques against the Kaggle dataset: linear methods, ensemble methods, and neural networks. Specifically, linear SVM, RF, and Neural Networks (NN) with training data subjected to SMOTE using oversampling are explored in this paper.

*1). K-Nearest Neighbors (KNN):*

After running the KNN algorithm on the dataset our model prediction performance was as below:

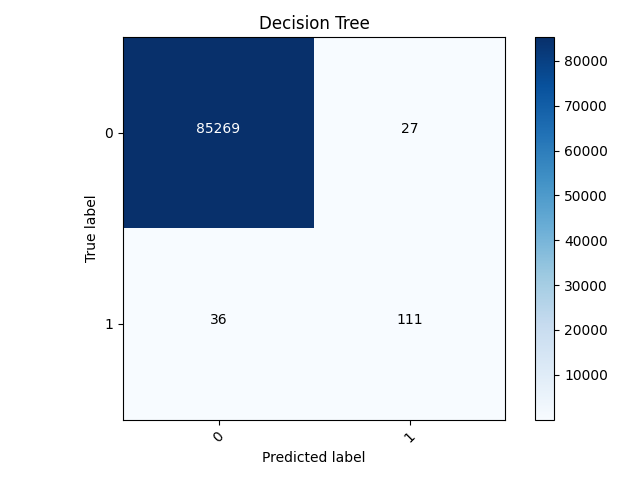
| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9994 | 0.2790 | 0.7211 | 0.9381 | 0.8154 |

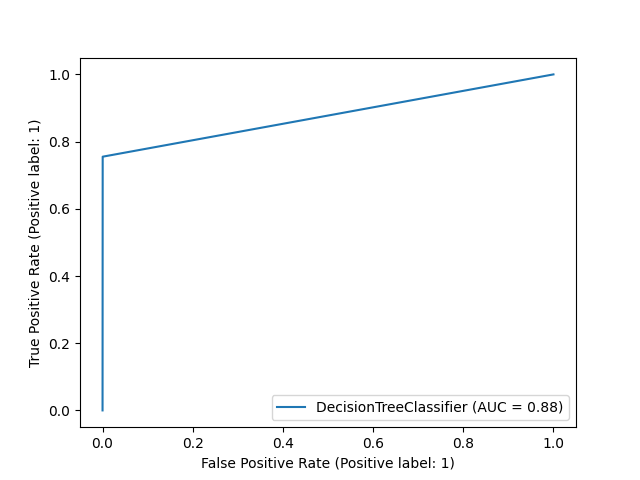
 

*2). Decision Tree (DT):*

After running the DT algorithm on the dataset our model prediction performance was as below:

| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9993 | 0.2450 | 0.7551 | 0.8043 | 0.7790 |

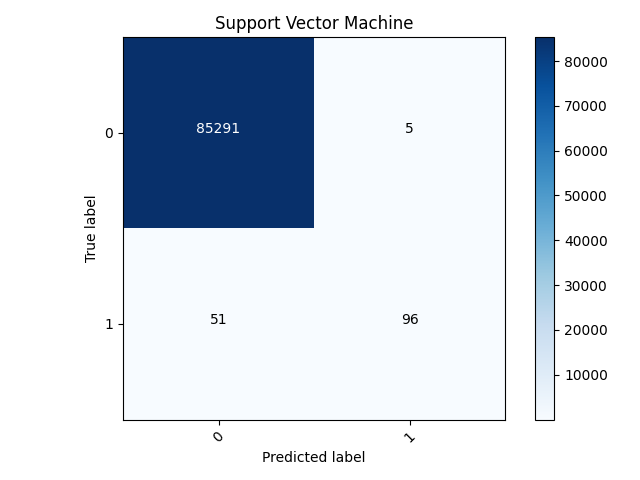
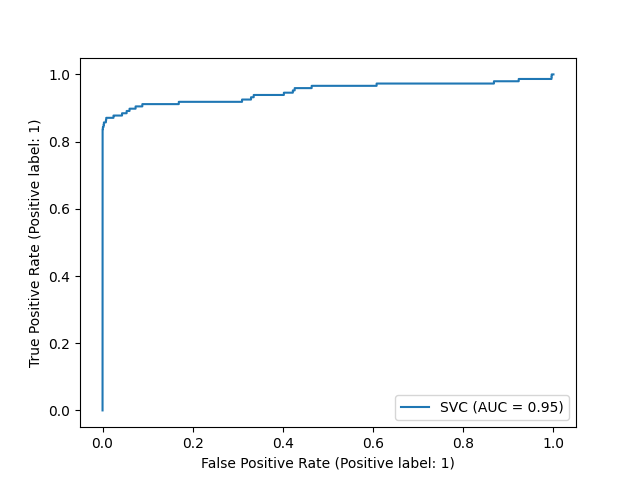




*3). Support Vector Machine (SVM):*

After running the SVM algorithm on the dataset our model prediction performance was as below:

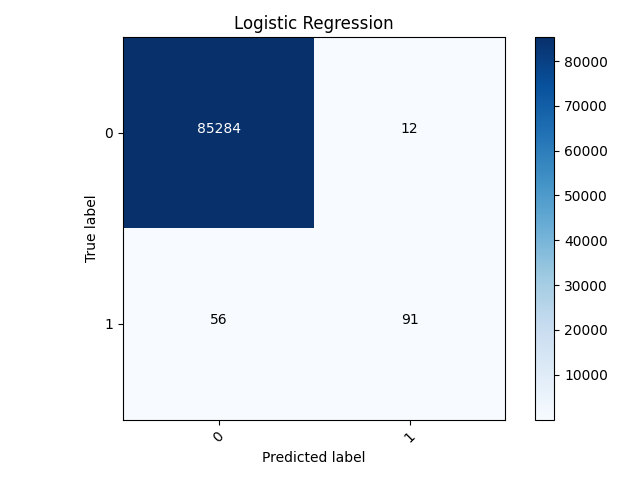
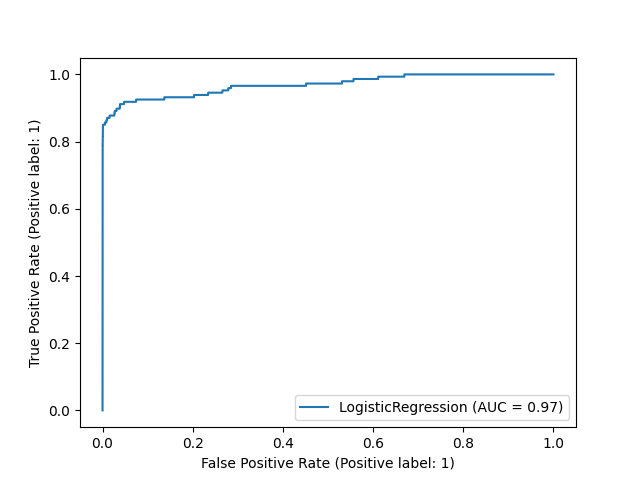
| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9994 | 0.3470 | 0.6531 | 0.9510 | 0.7750 |

*4). Logistic Regression (LR):*

After running the LR algorithm on the dataset our model prediction performance was as follows:

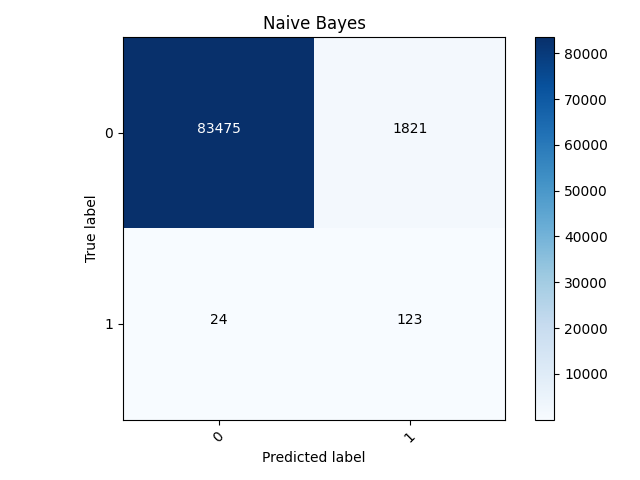
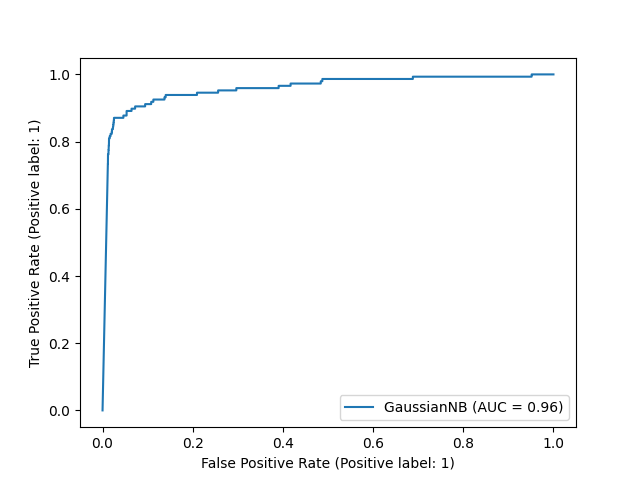
| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9992 | 0.3810 | 0.6200 | 0.8850 | 0.7280 |

*5). Naïve Bayes (NBG):*

After running the LR algorithm on the dataset our model prediction performance was as below:

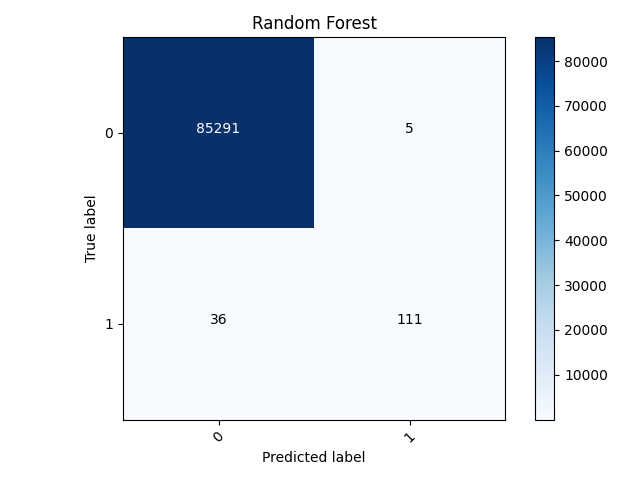
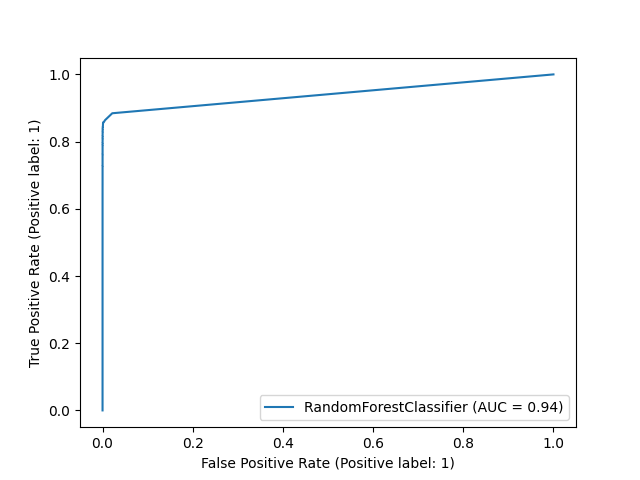
| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9784 | 0.1633 | 0.8368 | 0.0633 | 0.1177 |

*6). Random Forest (RF):*

After running the RF algorithm on the dataset our model prediction performance was as below:

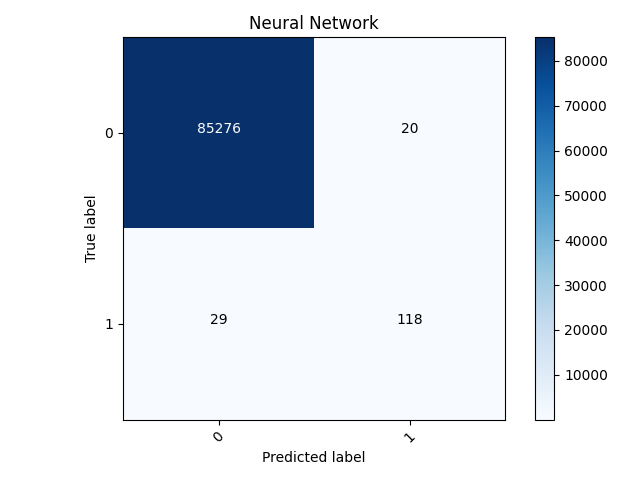
| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9995 | 0.2450 | 0.7551 | 0.9569 | 0.8441 |

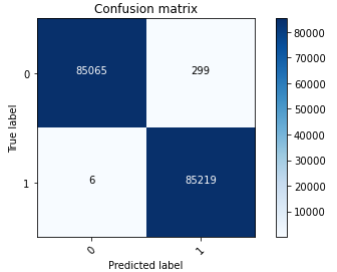
*7). Neural Networks (NN):*

After running the NN algorithm on the dataset our model prediction performance was as below:

| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9994 | 0.1973 | 0.8027 | 0.8551 | 0.8281 |

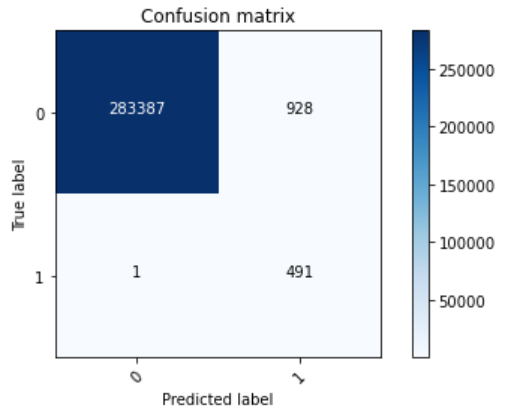


Training Neural Networks with oversampling SMOTE Below:



After training Neural Networks SMOTE, we run on the whole dataset again and we got amazing results:

| Accuracy | FalseNegRate | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| 0.9986 | 0.0011 | 0.9989 | 0.9982 | 0.9986 |



Note the almost zero value of False Negatives with SMOTE. The model is able to detect all fraudulent transactions on the full dataset. Note the limited number of False Positives which means a lot less verification work (on legitimate transactions) for the fraud department.

*8). Model Evaluation and Results:*

Overall, the accuracy rates for all models are extremely high. However, looking at this metric alone does not indicate that the predicted results are perfect. We should also inspect the other 4 applied metrics, which are false negative rate, recall, precision, and F1 score, to have a better view and deeper understanding of how these results are truly interpreted by each model.

With the Holdout method, it seems that KNN and Random Forest are the best two models for fraud detection on the test dataset in this case. Both of these models got higher accuracy and F1 score, which is the combination of precision and recall. Precision in our case provides us with the False Positive Rate, which means out of all the items labeled as fraud, what percentage of them is actually fraud. Recall on the other hand calculates the True Positive Rate, which means the percentage of all the fraudulent cases captured. So, the KNN and Random Forest were able to achieve the required results.

The Naive Bayes algorithm, on the other hand, does not predict fraudulent transactions as well because the precision value was quite low. That means the model could not predict the fraud case correctly.

By oversampling the dataset, it is obvious that the Neural Network model can improve its prediction by significantly lowering its false negative rate and increasing the recall, precision, and F1 score metrics. The obtained results from the oversampled Neural Network model could show an accuracy rate of up to 99.85% with a false negative rate of less than 0.2%, which is outstanding, compared to not applying the oversampling technique. The model is actually much better at detecting fraudulent cases. We have a lower False negative rate which is the key criteria for our purpose (detect a fraud when there is one). The model is able to detect all the fraudulent transactions on the unseen test set. All the metrics are excellent for the neural networks by using the oversampling technique.

1. ***Hyper Parameter Tuning and Cross fold***

Cross Validation is a very useful technique for assessing the performance of machine learning models. It helps in knowing how the machine learning model would generalize to an independent data set and to estimate how accurately your model will predict in practice.

| KVC | LR | DT | SVM | RF | NB |
| --- | --- | --- | --- | --- | --- |
| Testing Model | 0.9991 | 0.9983 | 0.9990 | 0.9995 | 0.9777 |

As you could see all the models perform similarly but random forest (RF) did the better prediction with 5-folds.

1. ***Conclusions***

The best results are achieved by oversampling the under-represented class using SMOTE (synthetic minority oversampling technique). With this approach, the model is able to detect 100% of all fraudulent transactions in the unseen test set. This fully satisfies the primary objective to detect the vast majority of abnormal transactions. Please note that the technique and model used are simple to implement, easy to use and can be updated in real-time. In addition, the number of false positives remains acceptable. This means a lot less verification work (on legitimate transactions) for the fraud department compared to some other approaches which failed on this aspect. Key results are shown in the table below:

| **Model** | **Accuracy** | **False**  **NegRate** | **Recall** | **Precision** | **F1 Score** |
| --- | --- | --- | --- | --- | --- |
| **KNN** | 0.9994 | 0.2789 | 0.7211 | 0.9381 | 0.8154 |
| **DecisionTree** | 0.9993 | 0.2245 | 0.7755 | 0.8201 | 0.7972 |
| **SVM** | 0.9994 | 0.3470 | 0.6531 | 0.9505 | 0.7742 |
| **Logistic Regression** | 0.9992 | 0.3810 | 0.6191 | 0.8835 | 0.7280 |
| **Naive Bayes** | 0.9784 | 0.1633 | 0.8367 | 0.0633 | 0.1177 |
| **Random Forest** | 0.9995 | 0.2517 | 0.7483 | 0.9402 | 0.8333 |
| **PlainNeural**  **Network** | 0.9994 | 0.2313 | 0.7687 | 0.8692 | 0.8159 |
| **OverSampled**  **NeuralNetwork** | 0.9982 | **0.0001** | 0.9999 | 0.9965 | 0.9982 |

**IV. Implementation and Code**

We use two approaches in the project, one creating the evaluation models from scratch and the other using the Python libraries, such as sklearn. We will be loading the code with this project. We also have compared the results of both code models created from scratch and Python libraries.

**V. References**

[1] “Credit Card Fraud Detection.” https://www.kaggle.com/mlg-ulb/creditcardfraud

[2] A. D. Pozzollo, “Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information” in *International Joint Conference on Neural Networks (IJCNN)*, 2015 © IEEE. doi: [10.1109/IJCNN.2015.7280527](https://doi.org/10.1109/IJCNN.2015.7280527)

[3] A. D. Pozzollo, *et. al*., “Learned lessons in credit card fraud detection from a practitioner perspective”, submitted for publication in Expert Systems with Applications, Feb 2014.

[4] “Scikit Learn Tutorial.” <https://www.tutorialspoint.com/scikit_learn/index.htm>

[5] “Global Payment Fraud Statistics, Trends & Forecasts.” https://www.merchantsavvy.co.uk/payment-fraud-statistics/