ALY 6040: DATA MINING AND APPLICATIONS

Assignment 6: Final Project

Submitted By

Siddharth Alashi

Mrityunjay Gupta

Kush Patel

Smit Parmar

NUID: 002728528

alashi.s@northeastern.edu

Submitted to

Prof. Andy Chen

05/21/2023

Project 6 Final Project: Credit Card Fraud Detection

Siddharth Alashi \\ Mrityunjay Gupta || Smit Parmar || Kush Patel
alashi.s@northeastern.edu || 002728528
College of Professional Studies, Northeastern University, Canada

I. Abstract

In this project, you will analyse customer-level data that has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group. In this project you will predict fraudulent credit card transactions with the help of Machine learning models. The data set is taken from the Kaggle website and has a total of 2,84,807 transactions; out of these, 492 are fraudulent. Since the data set is highly imbalanced, it needs to be handled before model building. In this model we've used metrics such as ROC-AUC SCORE, F1 score, precision, and recall We've predicted this machine learning models using Python and Juypter Notebook tools, Libraries and packages. In this project we have done Exploratory Data Analysis in which we've found the features of the data, Handled missing values, Outliers treatment, Observed distribution of classes, Splitting of data into train and test dataset, checking and treating skewness. Furthermore, we've also build the model using imbalance dataset, and created stratified cross validation schemas after narrowing the range of business models. Lastly, after getting final machine learning model observations on the dataset, we've created the same test on Balanced dataset.

II. Introduction

The dataset, obtained from Kaggle, comprises a total of 284,807 credit card transactions. Among these transactions, 492 are identified as fraudulent. It is crucial to address the significant class imbalance in the dataset before constructing a model.It is of utmost importance for credit card companies to be able to identify fraudulent credit card transactions to prevent customers from being charged for unauthorized purchases.

The dataset consists of credit card transactions made by European cardholders in September 2013. The transactions occurred over a span of two days, with 492 instances of fraud out of the total 284,807 transactions. The class distribution in the dataset is highly imbalanced, with the positive class (frauds) accounting for a mere 0.172% of all transactions.

The dataset includes only numerical input variables resulting from a PCA transformation. Unfortunately, due to confidentiality concerns, we are unable to provide the original features or additional background information about the data. The features V1, V2, ..., V28 correspond to the principal components obtained through PCA. The features 'Time' and 'Amount' are the only ones that have not undergone PCA transformation. The 'Time' feature represents the time elapsed in seconds between each transaction and the first transaction recorded in the dataset. The 'Amount' feature indicates the monetary value of each transaction and can be used for example-dependent cost-sensitive learning. The 'Class' feature serves as the response variable, taking a value of 1 in the case of fraud and 0 otherwise.Given the class imbalance ratio, it is advisable to assess accuracy using the Area Under the Precision-Recall Curve (AUPRC). The accuracy derived from a confusion matrix may not provide meaningful insights in the context of unbalanced classification.

III. Evaluation Metrics

ROC-AUC Score: The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score is a widely used metric for assessing the performance of binary classifiers. It quantifies a model's ability to differentiate between positive and negative classes by calculating the area under the curve of the receiver operating characteristic (ROC) curve.The ROC-AUC score is particularly suitable for imbalanced datasets as it is unaffected by class distribution. This means that even if the positive and negative classes are imbalanced in the dataset, the ROC-AUC score provides a reliable measure of the model's discriminatory power between the two classes.

F1 Score: However, it's important to note that the ROC-AUC score does not provide specific information about the performance of the model for each individual class. For instance, if one class is significantly smaller than the other, the model may perform well in the larger class but poorly in the smaller class, and this discrepancy may not be reflected in the ROC-AUC score. Therefore, it's crucial to consider other metrics like precision, recall, F1-score, or the confusion matrix when evaluating model performance on imbalanced datasets.The F1-score is a commonly used metric for evaluating the performance of binary classifiers. It combines precision and recall into a single score, providing an overall measure of model accuracy.

Precision: Precision is a measure of the correctness of a classification or prediction model. It is defined as the ratio of true positives (correctly predicted positive cases) to the sum of true positives and false positives (incorrectly predicted positive cases) in the model's output. In simple terms, precision quantifies the proportion of predicted positive cases that are actually positive.

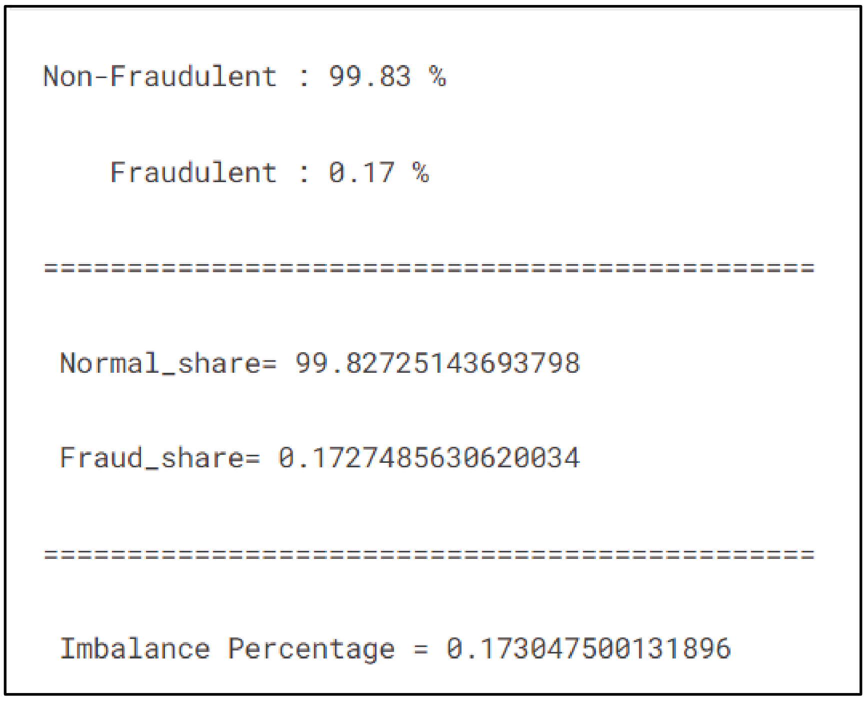
A high precision value indicates that the model excels at identifying positive cases, with only a few false positives in its output. On the other hand, a low precision value implies that the model has a higher rate of false positives, which can lead to incorrect or misleading outcomes.

Recall: Recall is a measure of the comprehensiveness of a classification or prediction model. It is defined as the ratio of true positives (correctly predicted positive cases) to the sum of true positives and false negatives (missed positive cases) in the model's output. In essence, recall measures the proportion of actual positive cases that the model correctly identifies.

A high recall value signifies that the model performs well in identifying positive cases, with only a few missed positive cases in its output. Conversely, a low recall value indicates that the model has a higher rate of missed positive cases, which can also result in incorrect or misleading outcomes.

IV. Exploratory Data Analysis

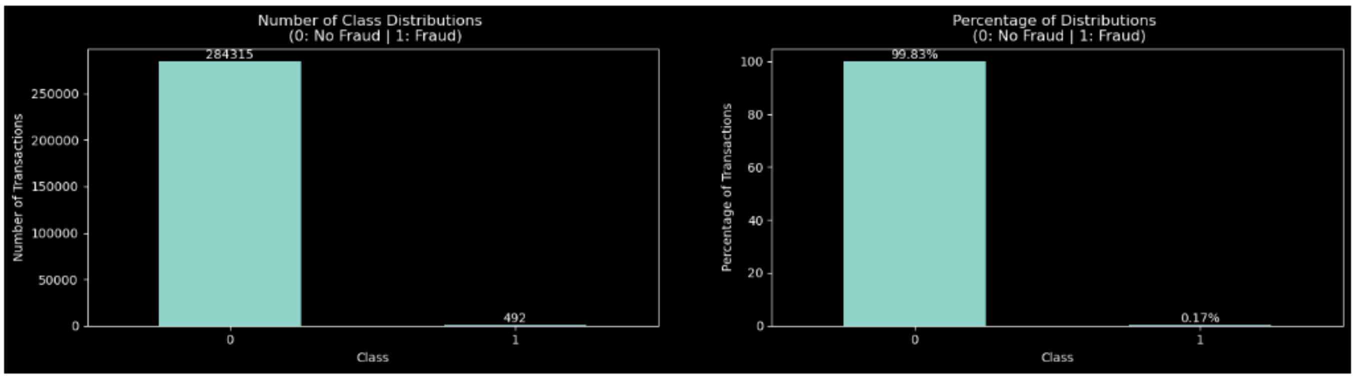
We have calculated the number of transactions which are fraudulent and non-fraudulent in a given dataframe. Afterward, it calculates the proportion of each class and saves the values in variables named "normal_share" and "fraud_share" correspondingly. Eventually, it displays the percentages of non-fraudulent and fraudulent transactions in the DataFrame by employing a formatted string.



```
Non-Fraudulent : 99.83 %

    Fraudulent : 0.17 %

==============================================

 Normal_share= 99.82725143693798

 Fraud_share= 0.1727485630620034

==============================================

 Imbalance Percentage = 0.173047500131896
```

The figure besides states that there are total 99.83% of Non-fraudulent transactions but 0.17% are fraudulent. The total imbalance percentage comes to around 0.1730 %.

Therefore this states that there are very less fraudulent transactions accord.



The code generates a figure containing two subplots arranged side by side. The left subplot exhibits the count of transactions for each class (0 or 1), while the right subplot showcases the percentage of transactions in each class. The figure's face color is customized to magenta using the 'facecolor' parameter. Using the 'plot' method of the pandas DataFrame 'classes', bar charts are constructed for both subplots. The left subplot has its y-axis labeled as 'Number of Transactions', x-axis labeled as 'Class', and a title of 'Number of Class Distributions'. Similarly, the right subplot has its y-axis labeled as 'Percentage of Transactions', x-axis labeled as 'Class', and a title of 'Percentage of Distributions'. The code proceeds to add labels to the bars in both subplots using the 'bar_label' method. Lastly, the figure is displayed by employing the 'show' method of pyplot.

V. Model Building with Imbalance data

We will construct models using the algorithms listed below and compare them to determine the best model. However, we will not be building models using SVM and KNN. These algorithms are computationally intensive and require substantial computational resources, especially for SVM and KNN. Due to the computational complexity involved, we have decided to skip these models. Instead, we will focus on the following models:

1. Logistic Regression
2. KNN
3. SVM
4. Decision Tree
5. Random Forest

We are going to use ROC-AUC score as the evaluation metric for the model evaluation purpose. As the data is highly imbalanced and we have only 0.17% fraud cases in the whole data, accuracy will not be the right metric to evaluate the model.

Final Observation on Imbalanced Dataset

A. CROSS VALIDATION - ROC-AUC Score of the models and best hyperparameters on Imbalanced data

- **LogisticRegression** {'C': 0.01, 'penalty': 'l2'} =
  - Best Mean ROC-AUC score for val data: 0.9797969874466093
  - Mean precision val score for best C: 0.885478588591554
  - Mean recall val score for best C: 0.6295975017349064
  - Mean f1 val score for best C: 0.7341406860856002
- **KNeighborsClassifier** {'metric': 'manhattan', 'n_neighbors': 9} =
  - 0.9274613536399045
- svm.SVC {'C': 0.01, 'gamma': 'auto', 'kernel': 'rbf', 'probability': True} =
  - 0.9565173998635063

- **DecisionTreeClassifier** {'criterion': 'entropy', '': 3, 'min_samples_leaf': 1, 'min_samples_split': 2} =
  - Best Mean ROC-AUC score for val data: 0.9337472016466822
  - Mean precision val score for best max_depth: 0.8480952241800844
  - Mean recall val score for best max_depth: 0.71578379211967
  - Mean f1 val score for best max_depth: 0.7752315571186218
- RandomForestClassifier {'min_samples_split': 5, 'n_estimators': 500} =
  - 0.9646808744238831

After narrowing down the options, the chosen number of estimators is 400. However, it is important to note that the model's training process for each set of 3 folds is time-consuming, taking approximately 1517.3193807601929 seconds (or roughly 25 minutes).

Increasing the number of trees in a random forest can have several benefits. Firstly, it helps reduce the variance of the model, leading to more stable and reliable predictions. Additionally, a larger number of trees means more votes for the final prediction, which can increase the accuracy of the model. Furthermore, increasing the number of trees allows the model to capture complex relationships between features in the dataset, resulting in better generalization performance.

Considering the above factors, the range of the number of estimators is set as [500]. However, it is important to note that running a set of 3 folds with this range will require approximately 2000 seconds (or around 35 minutes).

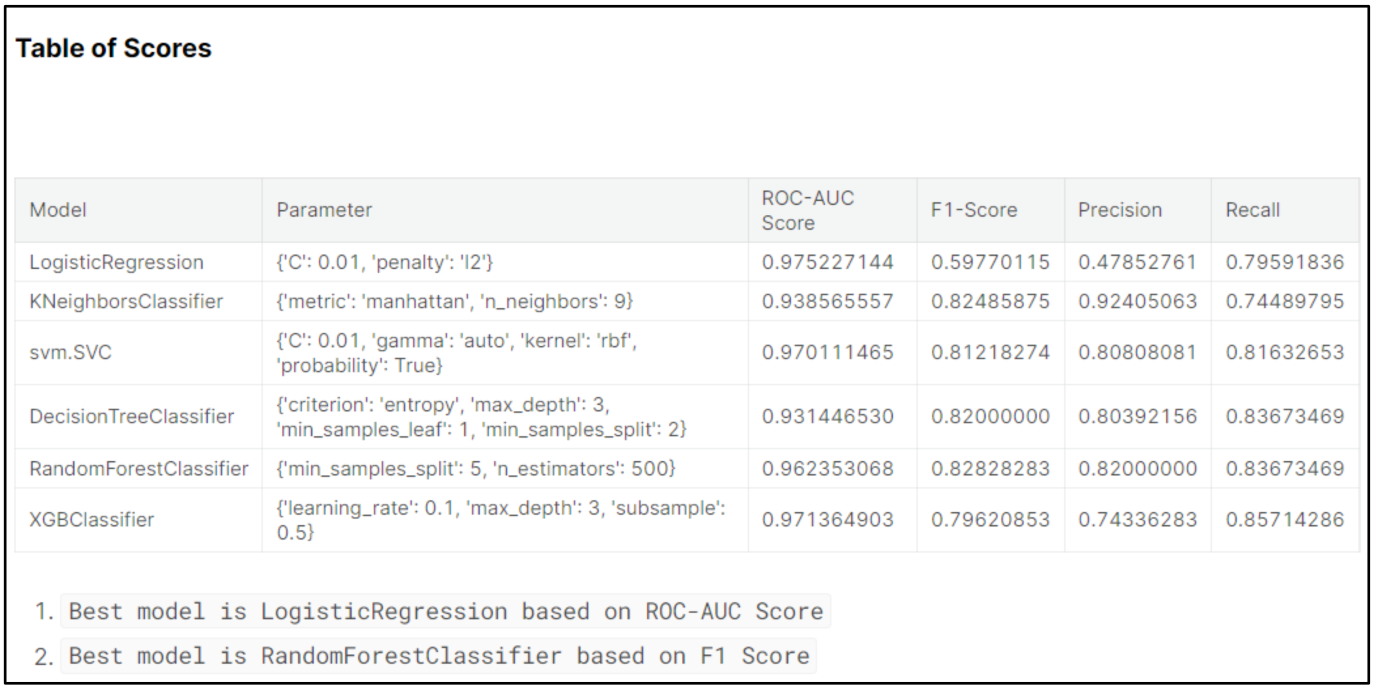


Table of Scores

Model	Parameter	ROC-AUC Score	F1-Score	Precision	Recall
LogisticRegression	{'C': 0.01, 'penalty': 'l2'}	0.975227144	0.59770115	0.47852761	0.79591836
KNeighborsClassifier	{'metric': 'manhattan', 'n_neighbors': 9}	0.938565557	0.82485875	0.92405063	0.74489795
svm.SVC	{'C': 0.01, 'gamma': 'auto', 'kernel': 'rbf', 'probability': True}	0.970111465	0.81218274	0.80808081	0.81632653
DecisionTreeClassifier	{'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2}	0.931446530	0.82000000	0.80392156	0.83673469
RandomForestClassifier	{'min_samples_split': 5, 'n_estimators': 500}	0.962353068	0.82828283	0.82000000	0.83673469
XGBClassifier	{'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.5}	0.971364903	0.79620853	0.74336283	0.85714286

1. `Best model is LogisticRegression based on ROC-AUC Score`
2. `Best model is RandomForestClassifier based on F1 Score`

VI. Model building with Balance data

Using accuracy as a metric for imbalanced datasets is not recommended. While a model may achieve a high accuracy score by correctly predicting the majority class, this can mask its poor performance in predicting the minority class. Relying on accuracy alone can be misleading and does not provide an accurate representation of the model's effectiveness.

Instead of accuracy, it is advisable to use metrics such as the F1-score, precision/recall score, or the confusion matrix when dealing with imbalanced datasets. These metrics take into account the performance of the model for both classes, providing a more comprehensive evaluation and avoiding misleading interpretations.

Undersampling involves selecting a reduced number of data points from the majority class during the model building process to achieve a balanced representation of both classes.

On the other hand, **oversampling** assigns weights to randomly selected data points from the minority class. This is done to emphasize the importance of this class while optimizing the loss function during training.

SMOTE (Synthetic Minority Over-sampling Technique) is a technique that generates new data points lying vectorially between two existing data points belonging to the minority class. This helps to augment the representation of the minority class in the dataset.

ADASYN (Adaptive Synthetic Sampling) is similar to SMOTE but with a slight modification. It not only generates synthetic samples but also considers the density distribution of the minority class. The objective of ADASYN is to create synthetic data for minority examples that are harder to learn, giving more emphasis to those examples rather than easier ones.

VII. Observation

A. CROSS VALIDATION - ROC-AUC Score of the models and best hyperparameters on Imbalanced data

- **LogisticRegression** {'C': 4, 'penalty': 'l2'} =
  - Best Mean ROC-AUC score for val data: 0.9884840531068964 [Before Oversampling {'C': 0.01, 'penalty': 'l2'} = 0.9812052138770543]
  - Mean precision val score for best C: 0.9719371184117677 [Before Oversampling {'C': 0.01, 'penalty': 'l2'} = 0.885478588591554]
  - Mean recall val score for best C: 0.9294177647053652 [Before Oversampling {'C': 0.01, 'penalty': 'l2'} = 0.6295975017349064]
  - Mean f1 val score for best C: 0.950201240931848 [Before Oversampling {'C': 0.01, 'penalty': 'l2'} = 0.7341406860856002]
- **KNeighborsClassifier** {'n_neighbors': 9} =
  - 0.9998373276002304 [Before Oversampling {'metric': 'manhattan', 'n_neighbors': 9} = 0.9274613536399045]
- **svm.SVC** {'C': 0.01, 'gamma': 'auto', 'kernel': 'rbf', 'probability': True} =

        - Not computed due to very large training time [Before Oversampling {'C': 0.01, 'gamma': 'auto', 'kernel': 'rbf', 'probability': True} = 0.9565173998635063]
- **DecisionTreeClassifier** {'criterion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 4, 'min_samples_split': 2} =
    - Best Mean ROC-AUC score for val data: 0.9981460788751075 [Before Oversampling {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2} = 0.9337472016466822]
    - Mean precision val score for best Max Depth: 0.9736125554386047 [Before Oversampling {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2} = 0.8480952241800844]
    - Mean recall val score for best Max Depth: 0.9558410382895657 [Before Oversampling {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2} = 0.71578379211967]
    - Mean f1 val score for best Max Depth: 0.964631942249586 [Before Oversampling {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2} = 0.7752315571186218]
- **RandomForestClassifier** {'min_samples_split': 5, 'n_estimators': 500} =
    - 1.0 [Before Oversampling {'min_samples_split': 5, 'n_estimators': 500} = 0.9646808744238831]
- **XGBClassifier** {'learning_rate': 0.6, 'max_depth': 5, 'subsample': 0.7} =
    - Best Mean ROC-AUC score for val data: 0.9999960678244962 [Before Oversampling {'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.5} = 0.9848866713890976]
    - Mean precision val score for best Learning Rate: 0.9995517600748944 [Before Oversampling {'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.5} = 0.9233400094242072]
    - Mean recall val score for best Learning Rate: 1.0 [Before Oversampling {'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.5} = 0.779204256303493]
    - Mean f1 val score for best Learning Rate: 0.9997758279719619 [Before Oversampling {'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.5} = 0.8448234879500908]

References

1. *Credit Card Fraud Detection*. (2018, March 23). Kaggle. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
2. Mazumder, S. (2022). 5 Techniques to Handle Imbalanced Data For a Classification Problem. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/