Fraud Analytics

Capstone Project

Tanmay Parmar

16/09/2023

Link: Fraud Analytics Project
https://github.com/Tanmayss1/Credit_Card_fraud_Analysis/blob/master/CreditCard_Fraud_Analytics.ipynb

Executive Summary: In this project, we used machine learning approaches to address the crucial problem of credit card fraud detection. With 284,807 transactions in the dataset, which was carefully analyzed for our evaluation, it became clear that there was a substantial class imbalance because there were only 492 fraud occurrences.

Background Information: Credit card fraud is an inclusive term for fraud committed using a payment card, such as a credit card or debit card. The purpose may be to obtain goods or services or to make payment to another account, which is controlled by a criminal.
Credit card fraud can occur when unauthorized users gain access to an individual's credit card information in order to make purchases, other transactions, or open new accounts.
There are two kinds of card fraud: card-present fraud (not so common nowadays) and card-not-present fraud (more common). The compromise can occur in a number of ways and can usually occur without the knowledge of the cardholder. The internet has made database security lapses particularly costly, in some cases, millions of accounts have been compromised.

Economic Impact: Credit card fraud costs the global economy $5.127 trillion annually. In 2022, $8.8 billion was lost to credit card fraud. The Arab Monetary Fund estimates that global credit card fraud losses in 2021 were $32.3 billion, a 13.8% increase from the previous year.

Problem Statement:

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

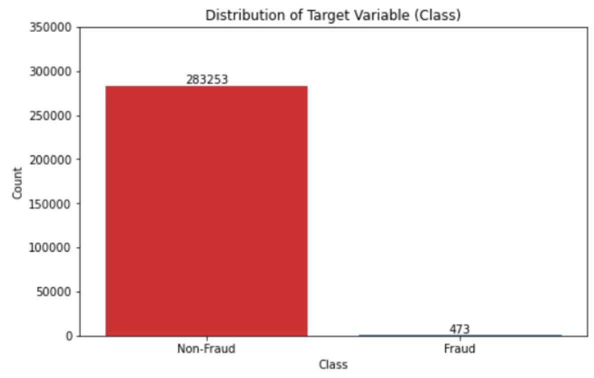
It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

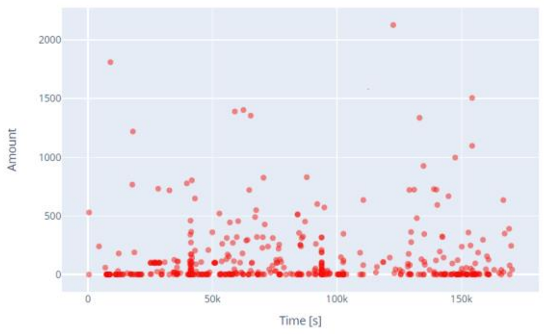
The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

We have to build a classification model to predict whether a transaction is fraudulent or not.

Objectives: To Develop an Effective Fraud Detection Model using, Machine Learning Models & Enhance Security for Financial Transactions.

Exploratory Data Analysis:





- There are 2 classes available 0 stands for No Fraud, whereas 1 stand for Fraud.
- The dataset is **highly unbalanced**, the **positive class (frauds)** account for **0.172%** of all transactions.
- It contains only numerical input variables which are the result of a **PCA (Principal Component Analysis) transformation**

Handling Missing Values: We began by identifying and addressing missing values in the dataset. Missing data points were found in various columns. Depending on the percentage of missing data in each column, we opted for strategies like removing rows with missing values, imputing missing values with a suitable measure (e.g., mean or median), or leaving certain columns untouched if the missing values were negligible.

Dealing with Duplicates: Duplicates in the dataset can distort model training. We addressed this issue by detecting and removing duplicate rows, ensuring each transaction entry is unique. This step reduced the chances of inadvertently assigning extra importance to repeated data points.

Outliers Detection and Treatment: Outliers, or extreme values in the data, can significantly affect model performance. We employed statistical methods to detect outliers. Depending on the context, we transformed these outliers to reduce their impact on the model.

Impact on the Model: Proper data cleaning and outlier treatment are critical for model training. These steps help ensure that the model learns patterns and relationships in the data accurately, without being skewed by noise or inconsistencies. Inaccurate data pre-processing can lead to overfitting or underfitting, reducing the model's generalization performance on unseen data. These techniques enhance model performance by preventing certain features from dominating others during training. Outliers Detection and Treatment: Outliers, or extreme values in the data, can significantly affect model performance.

Class Imbalance: Class imbalance is a situation in which there are significantly more instances of one class than the other in the context of fraud detection (or any other binary classification problem). For example, in the detection of credit card fraud, the vast majority of transactions are legitimate (Class 0), but only a very tiny percentage are genuinely fraudulent (Class 1).

Challenges with Imbalanced Data: This class imbalance poses challenges during model training. Models tend to be biased towards the majority class, achieving high accuracy simply by predicting the majority class most of the time. However, this is detrimental for fraud detection because we are more interested in accurately identifying the minority (fraud) class.

Techniques to Balance Data: Several techniques address this issue, broadly falling into two categories: under sampling and oversampling.

1. **Under sampling**: Removing instances from the majority class to balance the class distribution. However, this can lead to a loss of valuable information.
2. **Oversampling**: Duplicating or generating new instances for the minority class to balance the class distribution. Oversampling is often achieved through techniques like Synthetic Minority Over-sampling Technique (SMOTE), which generates synthetic samples based on existing minority samples.

Why Used Oversampling: Oversampling is a commonly used technique to handle imbalanced data because it allows the model to learn from the existing minority class instances while generating new data points. By creating synthetic samples, the model can better understand the features and patterns associated with the minority class, improving its ability to predict the rare events accurately. This helps prevent the model from being biased towards the majority class and leads to a more balanced and effective machine learning model for fraud detection.

Explanation of the chosen models:

In this Fraud detection, selecting the right machine learning models was crucial for achieving accurate and reliable results. Here's an explanation of the chosen models and the reasons for their selection:

1. **Logistic Regression:** Logistic Regression is a simple yet effective linear classification algorithm. Why Chosen: It serves as a baseline model due to its simplicity and interpretability. It helps understand the importance of each feature in predicting fraud.
2. **Random Forest:** Random Forest is an ensemble learning method that combines multiple decision trees. Why Chosen: Random Forest is robust, handles non-linearity, and can capture complex interactions between features. It also provides feature importance, aiding in understanding which features contribute most to fraud detection.
3. **XGBoost (Extreme Gradient Boosting):** XGBoost is a gradient boosting algorithm known for its high performance. Why Chosen: It excels in capturing intricate patterns in data, making it suitable for fraud detection. XGBoost often outperforms other algorithms and is widely used in competitions and real-world applications.
4. **K-Nearest Neighbours (KNN):** KNN is a simple instance-based learning algorithm. Why Chosen: KNN can be effective when detecting local anomalies. It's selected to explore whether proximity-based methods can be useful in identifying fraud patterns. Reasons for Model Selection:

Features Used for Data Splitting: When splitting the data into training and test sets, all features except the "Class" and "Time" features are used. The "Class" feature, which indicates whether a transaction is fraudulent or not (0 for non-fraud, 1 for fraud), is the target variable we want to predict, and "Time" feature is not beneficial for training as it doesn't mean anything to training and testing. All other features, such as "V1," "V2," ..., "V28,", and "Amount" are used to train the models.

Model Training and Hyperparameter Tuning: The models are trained using the training dataset, which comprises a subset of the original data. During the training process, each model learns to identify patterns and relationships between the features and the target variable. Hyperparameter tuning is performed to optimize the model's parameters, ensuring it performs at its best.

Performance Metrics Used for Evaluation: Several performance metrics are used to evaluate the models' effectiveness in fraud detection:

ROC AUC (Receiver Operating Characteristic - Area Under the Curve): ROC AUC measures the model's ability to distinguish between classes. A higher ROC AUC indicates better discrimination between fraud and non-fraud cases.

Accuracy: Accuracy measures the proportion of correctly classified transactions. However, accuracy may not be the most appropriate metric for imbalanced datasets.

Precision: Precision calculates the ratio of true positive predictions to all positive predictions. It assesses the model's ability to avoid false alarms (i.e., classifying non-fraud transactions as fraud).

Recall (Sensitivity): Recall calculates the ratio of true positive predictions to all actual positive cases. It assesses the model's ability to detect all fraudulent transactions.

F1-Score: The F1-Score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is useful when there's an uneven class distribution.

*** ROC AUC Score: A measure of how well the model distinguishes between fraud and non-fraud cases.**
*** Accuracy Score: The proportion of correctly classified transactions.**
*** Precision Score: The model's ability to avoid false alarms.**
*** Recall Score: The model's ability to detect fraudulent transactions.**
*** F1-Score: A balanced measure of precision and recall.**

Tools and Technologies Used in the Project:

IDE (Integrated Development Environment): Visual Studio Code (VS Code): Used as the primary code editor for writing and managing the project code.

Programming Language: Python: The core programming language used for data analysis, machine learning, and model development.

Data Manipulation and Analysis Libraries: Numpy: Used for numerical and mathematical operations on data arrays. Pandas: Used for data manipulation, data cleaning, and exploratory data analysis. Scipy: Used for advanced scientific and technical computing tasks.

Machine Learning Libraries: Scikit-learn (sklearn): Utilized for building, training, and evaluating machine learning models.

Data Visualization Libraries: Matplotlib: Used for creating static, animated, and interactive visualizations in Python. Seaborn: Built on top of Matplotlib, used for creating aesthetically pleasing statistical graphics.

Sampling Techniques: Sampling: Refers to techniques used for handling imbalanced datasets. This may include oversampling the minority class or undersampling the majority class to balance the dataset.

Microsoft Excel: Used for data pre-processing tasks, such as initial data exploration and cleaning.

The different classification reports for different machine learning models (Logistic Regression, XGBoost, Random Forest, and k-Nearest Neighbours) used for fraud detection, here are the key findings:

Logistic Regression (logc):

- Precision for class 1 (fraudulent transactions) is quite low (0.11), indicating that it has a high rate of false positives.
- Recall for class 1 is high (0.90), which means that it captures a significant portion of actual fraud cases.
- The F1-score for class 1 is relatively low (0.19), indicating that there is room for improvement in balancing precision and recall.

XGBoost (xgbc):

- Precision for class 1 has improved significantly compared to Logistic Regression, now at 0.77.
- Recall for class 1 remains high at 0.83.
- The F1-score for class 1 has also improved to 0.80.
  XGBoost shows a good balance between precision and recall for class 1.

Random Forest (rfc):

- Precision for class 1 is even higher at 0.88.
- Recall for class 1 is slightly lower at 0.82 but still strong.
- The F1-score for class 1 is relatively high at 0.85.
  Random Forest provides a good trade-off between precision and recall for class 1.

k-Nearest Neighbors (knnc):

- Precision for class 1 is the lowest among the models at 0.04.
- Recall for class 1 is very high at 0.93, indicating a low rate of false negatives.
- The F1-score for class 1 is relatively low at 0.07, mainly due to the low precision.

General Observations:

- All models achieve high accuracy (close to 1.00) but differ significantly in precision, recall, and F1-score for class 1.
- XGBoost and Random Forest outperform Logistic Regression and k-Nearest Neighbours in terms of precision, recall, and F1-score for class 1.
- For minimizing false positives (precision) is crucial, then XGBoost or Random Forest might be preferred.
- For capturing more fraud cases (recall) is the priority, k-Nearest Neighbours might be considered.
- Further evaluation, including cross-validation and hyperparameter tuning, could help select the most suitable model for the specific use case.

Here are the model tuning results for Logistic Regression, Random Forest, XGBoost, and k-Nearest Neighbours (knn) based on precision, recall, and F1-score for class 1 (fraudulent transactions):

Logistic Regression: Precision: 0.11 Recall: 0.90 F1-score: 0.19
Random Forest: Precision: 0.62 Recall: 0.87 F1-score: 0.73
XGBoost: Precision: 0.85 Recall: 0.83 F1-score: 0.84
k-Nearest Neighbours (knn): Precision: 0.52 Recall: 0.85 F1-score: 0.64

General Observations:

- Logistic Regression has a high recall for class 1, indicating it captures most of the fraudulent transactions but has low precision, resulting in many false positives.
- Random Forest achieves a better balance between precision and recall compared to Logistic Regression, with a higher F1-score for class 1.

- XGBoost shows further improvements in precision, recall, and F1-score for class 1, making it a promising choice.
- k-Nearest Neighbours (knn) has a high recall but a relatively low precision and F1-score for class 1.

  It appears that XGBoost and Random Forest perform well after tuning, with XGBoost having a slightly higher precision and F1-score for class 1. However, the choice of the best model should consider other factors such as computational resources and interpretability. Further evaluation and testing may be required to make the final model selection.