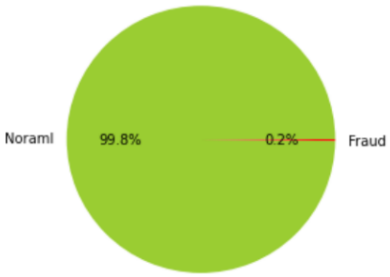
Fraudulent Transaction Detection

Author :
Last name: Terente
First name: Elizaveta

I. Introduction

The main task of this project is to demonstrate the capability to identify fraudulent transactions. I will make use of the binary classifier i trained on the transactions dataset to detect fraud. Our dataset contains 227845 rows and 31 columns. Time, 27 features without particular name, Amount and last column : class. Class values can be 0 (usual transaction) and 1 (frauded transaction). Dataset is imbalanced : we have 227451 samples with normal transactions and only 394 with frauded transactions.



II. Implementation / ML Process

Naive Bayes

1) *Data preprocessing*
   - The *StandardScaler* from *scikit-learn* is used to perform feature scaling .
   - Removing outliers with Z score (threshold value defined as 3). For better performance i separate data into 2 subsets : 1st - all samples with Class 0 and 2nd - all samples with Class 1 and perform outliners remove on this 2 separate sets. Only then concatenate them back together. Otherwise instances of class 1 can be removed as outliners. As a result 32393 outliners of class 0 and 40 outliners of class 1 removed. This increases performance much!
   - To investigate which features are better to choose i used *RandomForestClassifier* from *sklearn.ensemble* and analyzed which features have most correlation with target classes running build-in function for *feature_importances_* (*see figure 1*). By running it multiple times it got clear that the most impact has feature 13. By trying different combinations of features (feature 13 with other features) it was determined that best performance gives ONLY feature 13. Adding other features lowers performance (slightly and not). I save top feature to a model as a parameter and then in *leader_board_predict_fn* i clean data set leaving only top feature). I am performing this operation on the set with already removed outliners!

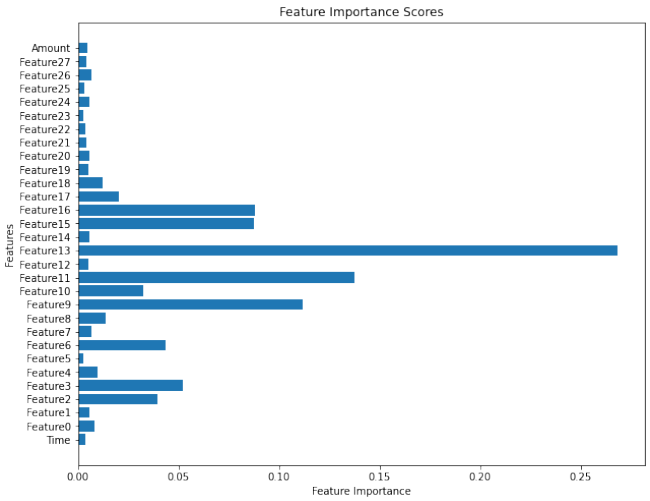


Fig. 1. Naive Bayes Feature Importance Score

   - I splitted our pre-processed train data into train set (70%) and test set(30%) to test classifier with *train_test_split* from *sklearn.model_selection*

2) *Classifier architecture*
   GaussianNB classifier from *sklearn.naive_bayes* is used. The architecture of the GaussianNB classifier is based on the Naive Bayes theorem, which is a probabilistic algorithm that uses Bayes' rule to estimate the probability of a sample belonging to a particular class. The "Gaussian" in GaussianNB refers to the assumption that the probability distributions of the features within each class are Gaussian or normal distributions.
   Classifier estimates the mean and standard deviation of feature(i have only one) for each class in the training data. During prediction, it calculates the probability of the sample belonging to each class based on the Gaussian (normal) distribution assumption. The class with the highest probability will be assigned to the sample.
3) *Hyperparameters*
   Default hyper parameters are being used + I am saving top feature in my model.

III. Results

The **Naive Bayes** classifier performs good on the dataset, with an Accuracy of 99% (but this value is high partly because of dataset imbalance), Precision of 1.0 which means that model avoids false positives very well(For non-fraudulent transactions all the values are at 100%), Recall of 85% (capturing of positive samples) , F1(balance between precision and recall) of 92% and AUC-ROC of 96%, which indicates the classifier's ability to distinguish between positive and negative instances.

TABLE I
ROC AUC score

Classifier	Train Set	Test Set
Random Forest	99,35%	95,27%

IV. Discussion

The GaussianNB **Naive Baye**s Classifier demonstrates great performance with a high accuracy of approximately 99.97% and a precision score of 1.0, indicating a low rate of false positives. It effectively captures positive instances with a recall score of approximately 85.47% and achieves a balanced F1-score of approximately 0.92. The AUC-ROC value of 0.96 indicates its strong ability to distinguish between positive and negative instances. Overall, the classifier exhibits robust performance, accurately classifying the majority of instances and effectively capturing positive cases in this particular classification task. The training of a model may require some time(up to 5 minutes), primarily due to the random forest's top feature search process. However, once the model is trained, it can be utilized swiftly and efficiently for making predictions or performing inference tasks. I tested classifier with big amount of different features and combinations of the features(detemined by random forest) to find the best combination (feature 13). I tried to apply weight to classes,because its highly imbalanced, but it didn't make a difference. I tried everything i wanted.

V. Conclusion

Final result for **Naive Bayes** on unseen data is Train Dataset Score: **95% on Train Dataset** and **97% on Test Dataset** i consider as successful. Classifier performs better on the test set than on the training set, it suggests that the classifier generalizes well to unseen data.This scenario indicates that the model has not overfit the training data and can effectively capture the underlying patterns in new, unseen instances. The improved performance on the test set demonstrates that the classifier has successfully learned the relationships between the features and the target variable, allowing it to make accurate predictions on previously unseen data.