Credit Card Fraud Detection Using Machine Learning & Python

Final Project Report

Submitted by:

Adesh Padwal - app27@njit.edu
Daanish Quadri - dq33@njit.edu
Sachin Singh Satyanaraya - ss4725@njit.edu

Table of Contents

1. Introduction

Fraud in credit card transactions refers to the illegal and undesired use of a credit card account by someone other than the account owner. It is possible to halt this misuse with the necessary preventative measures, and it is also possible to study the behavior of such fraudulent operations to reduce future occurrences and safeguard against them. To put it another way, credit card fraud may be described as an instance where someone uses another person's credit card without the owner's or the card's issuing authority being aware of it. Monitoring user populations' behavior is a key component of fraud detection because it allows fraud, intrusion, and defaulting to be identified as well as other objectionable behavior. This is a pertinent issue that must be addressed by fields like machine learning and data science, where an automated solution is possible. From the standpoint of learning, this issue is particularly difficult since it is characterized by many characteristics, such class imbalance. There are much more legitimate transactions than fraudulent ones. Additionally, the statistical characteristics of the transaction patterns frequently vary over time.

It is important that credit card companies can recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Here, comes the need for a system that can track the pattern of all the transactions and if any pattern is abnormal then the transaction should be aborted. In this project, We are trying to determine which model is best to detect fraudulent activity.

We have many machine learning algorithms that can help us classify abnormal transactions. The only requirement is the past data and the suitable algorithm that can fit our data in a better form.

2. Importing the libraries

We use python as the programming language and the libraries we import for this project are as follows:

```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

# text customization
from termcolor import colored as cl

#Packages related to data visualization
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.impute import MissingIndicator, SimpleImputer
from sklearn.preprocessing import  PolynomialFeatures, KBinsDiscretizer, FunctionTransformer
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder,LabelBinarizer,
OrdinalEncoder
import statsmodels.formula.api as smf
import statsmodels.tsa as tsa
from sklearn.linear_model import LogisticRegression, LinearRegression, ElasticNet, Lasso, Ridge
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.tree import *
from sklearn.ensemble import BaggingClassifier, BaggingRegressor, RandomForestClassifier,
RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier,GradientBoostingRegressor,
AdaBoostClassifier, AdaBoostRegressor
from sklearn.svm import LinearSVC, LinearSVR, SVC, SVR
from xgboost import XGBClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

3. Loading the Dataset

We have taken the dataset for Credit Card Fraud Detection from the link below.

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

4. Splitting the Dataset

Before splitting the data into train & test. we defined dependent and independent variables.
The dependent variable is known as X and the independent variable is known as y.
Then, we have split the dataset into training and testing datasets.

5. Exploratory Data Analysis (EDA)

Dataset contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we could not be able to get the original features and more background information about the data. Features (V1-V28) are the principal components obtained with PCA, the only features which have not been transformed with PCA are '*Time*' and '*Amount*'.

Features:

1. Time: contains the seconds elapsed between each transaction and the first transaction in the dataset.
2. Amount: feature 'Amount' is the transaction Amount. this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable, and it takes value 1 in case of fraud and 0 otherwise.

6. Results:

models	Accuracy	F1 Score
Decision Tree	0.999288989494457	0.776255707762557
K-Nearest Neighbors	0.999506645771664	0.8365384615384616
Logistic Regression model	0.9991148644726914	0.6934673366834171
Support vector machine	0.9993615415868594	0.7777777777777779
Random Forest	0.9993615415868594	0.7843137254901961
XG Boost	0.9995211561901445	0.8421052631578947

7. Conclusion:

We have used XGBoost model, K-Nearest Neighbors model and Logistic Regression model, Decision Tree model, Support Vector model and Random Forest model. We have already received **99.95% accuracy** in our credit card fraud detection by using **XGBoost model,** which is highest accuracy we have received among all the six models. This number should not be surprising as our data was balanced towards one class. The good thing that we have noticed from the confusion matrix is that our model is not overfitted.

Based on class imbalance ratio, we decided measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC) because confusion matrix accuracy is not meaningful for unbalanced classification.

Finally, based on our accuracy score, XGBoost gave better accuracy than other two models. The only catch here is the data that we have received for model training is transformed version of PCA.