# FINAL PROJECT

**Applied Data Science, Machine Learning & IoT**

**Credit Card Fraud Detection**

## Nitin Kashinath Gaikwad

Nitin Kashinath Gaikwad
ntngaikwad@gmail.com
14 Jan 2023

# Contents

1. Business Problem ..... 2
2. Software and tools ..... 2
3. We will follow the high-level approach given below ..... 2
4. Executive Summary ..... 5
5. Conclusion ..... 7
6. Annexure: Brief Exploratory Data Analysis (EDA) ..... 7

## 1. Business Problem

**Credit card fraud detection:**

**Objective: Build an ML model to detect fraud in credit card transactions.**

We need to classify whether a credit card transaction is genuine or whether someone has hacked and tried to perform the fraudulent transaction. Here the challenge is to ensure that real customers are not annoyed and protected.

This case follows in the binary classification, so we will try our multiple classification algorithms and finalize the one that performs best for this data type.

**Note: I have referred to and taken the dataset from https://www.kaggle.com/ site.**

The dataset contains transactions made by credit cards in September 2013 by European cardholders.

Per the Kaggle, this dataset presents transactions that occurred in two days, with 492 frauds out of 284,807 transactions. The dataset is highly unbalanced; the positive class (frauds) accounts for 0.172% of all transactions.

## 2. Software and tools

**We will use the ML classification algorithms in Python using ANACONDA Jupiter Notebook.**

## 3. We will follow the high-level approach given below

1. Understand the data set / Data Description
    a. Attribute Information
    b. Variable/Features data types and categories
2. Data Preparation & Handling Missing Values
3. Exploratory Data Analysis (EDA)
    a. Descriptive Statistics
    b. Data Visualization

c. Feature Selection

d. Data Pre-processing data

e. Transformation

- Scaling, Decomposition, Aggregation, Data balancing,

4. Model Selection (Modelling):

A. Logistics Regression

B. KNN

C. Naïve Bayes

a. GaussianNB

b. MultinomialNB

c. BernoulliNB

D. SVM

E. Gradient boosting

a. AdaBoostClassifier

b. GradientBoostingClassifier

c. XGBClassifier

## Steps followed for each ML Model

**i. Preparing the data to train a model**

1. Dataset splitting

a. Training set.

b. Test set

2. Model training

3. Model evaluation and testing

4. Module Turing and Improving predictions

5. Cross-validation

6. Prediction

**ii. Model Performance metrics**

5. Conclusion

**NOTE:** This document contains brief project reports, and the Jupyter notebook has details and code with results.
**Please refer:**

1. **"NKG_Final_Project_CreditCardFraudDetection_classification_v10.ipynb"** Jupyter notebook for the project given dataset (**creditcard.csv**) for details and code with results.

2. **"creditcard.csv"** is the dataset provided along with Final Project**.**

3. **"NKG_CCFD_classification_model_comparison.csv"** different classification models actual training & test accuracy, precision, recall, and FI Score results with comparison.

- **Classification tried-out Model list (Total of 9 Models tried out):**

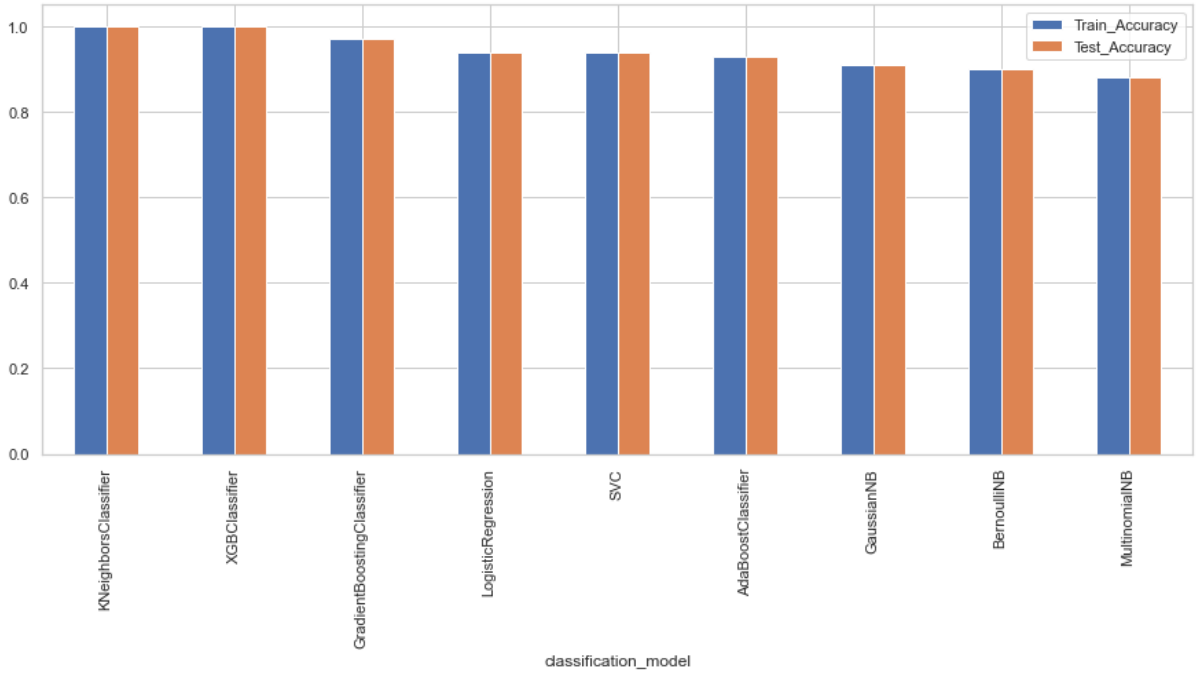
    1. **Logistics Regression**
    2. **KNN (K-Nearest Neighbor)**
    3. **Naïve Bayes**
        A. **GaussianNB**
        B. **MultinomialNB**
        C. **BernoulliNB**
    4. **SVM**
    5. **Gradient boosting**
        A. **AdaBoostClassifier**
        B. **GradientBoostingClassifier**
        C. **XGBClassifier**

4. **"Nitin Kashinath Gaikwad - Final Project- 14Jan2023.Zip"** contains all the above files, including the project summary document.

## 4. Executive Summary

Please note that I have used the GridSearchCV to find the best hyperparameters for the model for the dataset, but I have not created the final model for each classifier.

- **Tried-out Classification Models comparison: Training accuracy vs. Test Accuracy**




| | classification_model | Train_Accuracy | Test_Accuracy | Accuracy | weighted_Accuracy |
|---|---|---|---|---|---|
| **0** | KNeighborsClassifier | 1.00 | 1.00 | 1.00 | 1.00 |
| **1** | XGBClassifier | 1.00 | 1.00 | 1.00 | 1.00 |
| **2** | GradientBoostingClassifier | 0.97 | 0.97 | 0.97 | 0.97 |
| **3** | LogisticRegression | 0.94 | 0.94 | 0.94 | 0.94 |
| **4** | SVC | 0.94 | 0.94 | 0.94 | 0.94 |
| **5** | AdaBoostClassifier | 0.93 | 0.93 | 0.93 | 0.93 |
| **6** | GaussianNB | 0.91 | 0.91 | 0.91 | 0.91 |
| **7** | BernoulliNB | 0.90 | 0.90 | 0.90 | 0.90 |
| **8** | MultinomialNB | 0.88 | 0.88 | 0.88 | 0.88 |

- **Tried-out Classification Models comparison of the precision, recall, and FI Score**

| | classification_model | Train_f1_score_weighted_avg | Test_f1_score_weighted_avg |
|---|---|---|---|
| **0** | KNeighborsClassifier | 1.00 | 1.00 |
| **1** | XGBClassifier | 1.00 | 1.00 |
| **2** | GradientBoostingClassifier | 0.97 | 0.97 |
| **3** | LogisticRegression | 0.94 | 0.94 |
| **4** | SVC | 0.94 | 0.94 |
| **5** | AdaBoostClassifier | 0.93 | 0.93 |
| **6** | GaussianNB | 0.91 | 0.91 |
| **7** | BernoulliNB | 0.90 | 0.90 |
| **8** | MultinomialNB | 0.88 | 0.88 |

| | classification_model | Train_recall_weighted_avg | Test_recall_weighted_avg |
|---|---|---|---|
| **0** | KNeighborsClassifier | 1.00 | 1.00 |
| **1** | XGBClassifier | 1.00 | 1.00 |
| **2** | GradientBoostingClassifier | 0.97 | 0.97 |
| **3** | LogisticRegression | 0.94 | 0.94 |
| **4** | SVC | 0.94 | 0.94 |
| **5** | AdaBoostClassifier | 0.93 | 0.93 |
| **6** | GaussianNB | 0.91 | 0.91 |
| **7** | BernoulliNB | 0.90 | 0.90 |
| **8** | MultinomialNB | 0.88 | 0.88 |

| | classification_model | Train_precision_weighted_avg | Test_precision_weighted_avg |
|---|---|---|---|
| **0** | KNeighborsClassifier | 1.00 | 1.00 |
| **1** | XGBClassifier | 1.00 | 1.00 |
| **2** | GradientBoostingClassifier | 0.97 | 0.97 |
| **3** | LogisticRegression | 0.94 | 0.94 |
| **4** | SVC | 0.94 | 0.94 |
| **5** | AdaBoostClassifier | 0.93 | 0.93 |
| **6** | GaussianNB | 0.92 | 0.92 |
| **7** | BernoulliNB | 0.92 | 0.92 |
| **8** | MultinomialNB | 0.90 | 0.91 |

## 5. Conclusion

**After examining the confusion matrix, scores and RocCurve, we can choose XGBClassifier or KNeighborsClassifier. These two models have given good results for the provided data.**

## 6. Annexure: Brief Exploratory Data Analysis (EDA)

**For EDA details, Please refer to the NKG_Final_Project_CreditCardFraudDetection_classification_v10.ipynb file.**

**Brief Observations:**

- There were no missing values.
- We have 1081 duplicate rows, so we need to delete those.
- "Class" feature distribution clearly shows that this is the case of an imbalanced dataset.
- There are 473 frauds out of 283,726 transactions. It is 0.1667%
- We need to handle an imbalanced dataset, and there are many techniques like "Collect more data," "Use oversampling and undersampling," "Use class weights," or Use some algorithms. Considering options, I would prefer to go with SMOTE machine-learning technique.
- But before that, we need to understand more about the features and select the required features