Project

Find Default (Prediction of Credit Card fraud)

Abstract-

This Project is focused on credit card fraud detection in real world scenarios. Nowadays credit card frauds are drastically increasing in number as compared to earlier times. Criminals are using fake identity and various technologies to trap the users and get the money out of them. Therefore, it is very essential to find a solution to these types of frauds. In this proposed project we designed a model to detect the fraud activity in credit card transactions. This system can provide most of the important features required to detect illegal and illicit transactions. As technology changes constantly, it is becoming difficult to track the behavior and pattern of criminal transactions. To come up with the solution one can make use of technologies with the increase of machine learning, artificial intelligence and other relevant fields of information technology; it becomes feasible to automate this process and to save some of the intensive amounts of labor that is put into detecting credit card fraud. Initially, we will collect the credit card usage data-set by users and classify it as trained and testing dataset using a random forest algorithm and decision trees. Using this feasible algorithm, we can analyze the larger data-set and user provided current data-set. Then augment the accuracy of the result data. Proceeded with the application of processing of some of the attributes provided which can find affected fraud detection in viewing the graphical model of data visualization. The performance of the techniques is gauged based on accuracy, sensitivity, and specificity, precision. The result is indicated concerning the best accuracy for Random Forest are respectively. Keywords— Random forest algorithm, Criminal transactions, Credit card

Introduction

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. Eventually, it is also important for companies NOT to detect transactions which are genuine as fraudulent; otherwise, companies would keep blocking the credit card, and which may lead to customer dissatisfaction. So here are two important expects of this analysis:

- What would happen when the company will not able to detect the fraudulent transaction and would not confirm from a customer about this recent transaction whether it was made by him/her.
- In contract, what would happen when the company will detect a genuine transaction as fraudulent and keep calling customer for confirmation or might block the card.

The datasets contain transactions that have 492 frauds out of 284,807 transactions. So the dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. When we try to build the prediction model with this kind of unbalanced dataset, then the model will be more inclined towards to detect new unseen transaction as genuine as our dataset contains about 99% genuine data.

As our dataset is highly imbalanced, so we shouldn't use accuracy score as a metric because it will be usually high and misleading, instead use we should focus on f1-score, precision/recall score or confusion matrix.

Load Data-

```
# Import Libraries

Import warnings

warnings.filterwarnings ('ignore')

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# import cufflinks as cf

import plotly

import datetime

import math

import matplotlib

import sklearn
```

```
from IPython.display import HTML

import pickle

import os

import plotly.express as px

import plotly.graph_objects as go

import plotly.figure_factory as ff

from plotly.subplots import make_ subplots
```

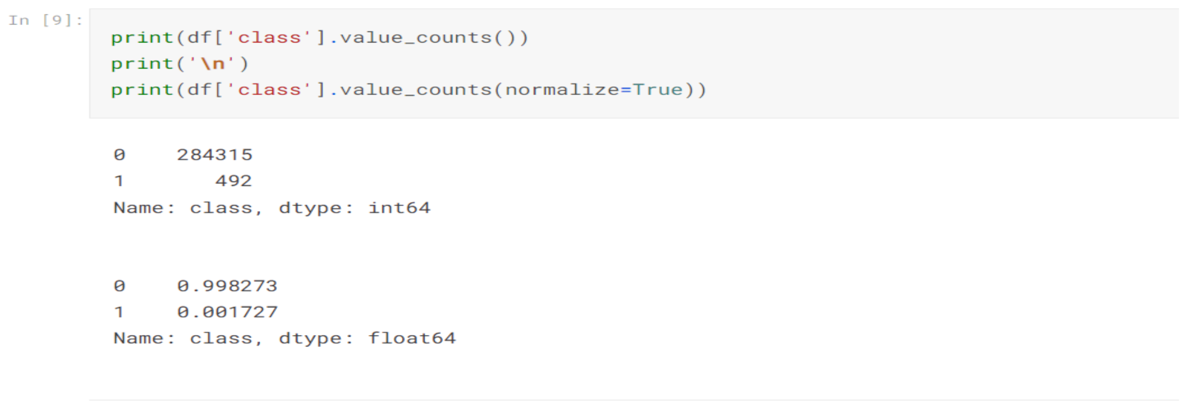
Output:

```
Numpy version : Numpy 1.24.3
Pandas version : Pandas 1.5.3
Matplotlib version: Matplotlib 3.7.1
Seaborn version: Seaborn 0.12.2
SkLearn version: SkLearn 1.2.2
Plotly version: plotly 5.9.0
```

Exploratory Data Analysis

Once the data is read into python, we need to explore/clean/filter it before processing it for machine learning it involves adding/deleting few columns or rows, joining some other data, and handling qualitative variables like dates.

Now that we have the data, I wanted to run a few initial comparisons between the three columns - Time, Amount, and Class.

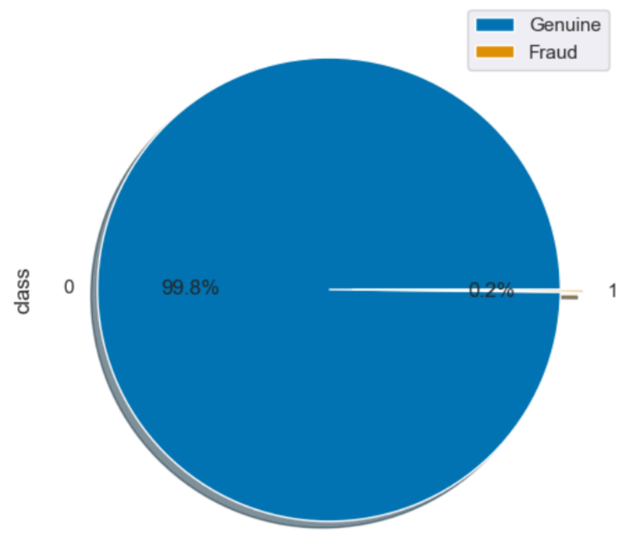


```
In [9]:
print(df['class'].value_counts())
print('\n')
print(df['class'].value_counts(normalize=True))
```

```
0    284315
1       492
Name: class, dtype: int64


0    0.998273
1    0.001727
Name: class, dtype: float64
```

Fraudulent and Non-Fraudulent Distribution



Highlights

This dataset has 492 frauds out of 284,315 transactions. The dataset is **highly unbalanced**, the positive class (frauds) account for 0.172% of all transactions. Most of the transactions are non-fraud. If we use this dataframe as the base for our predictive models and analysis, our algorithms will probably overfit since it will "assume" that most transactions are not a fraud. But we don't want our model to assume, we want our model to detect patterns that give signs of fraud!

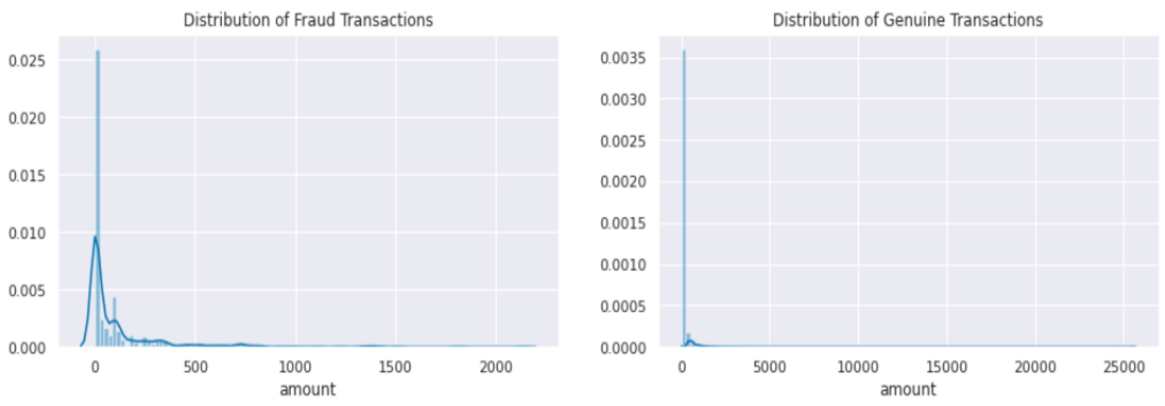
Highlights

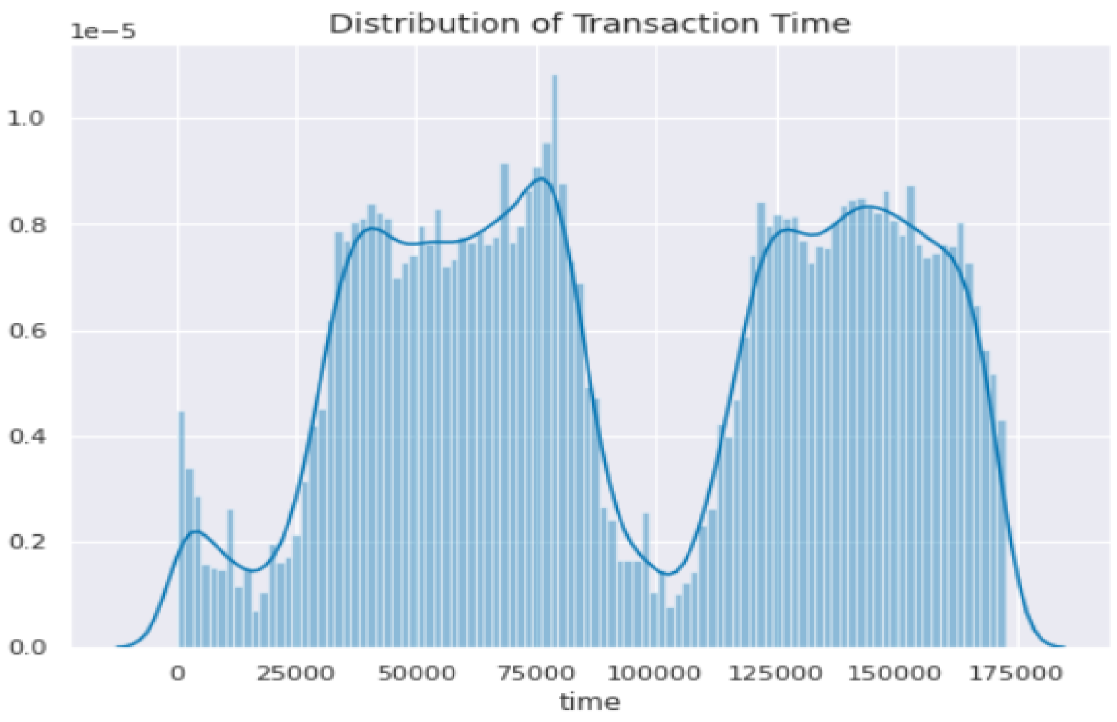
- Dataset contains details of 284807 transactions with 31 features.
- There is no missing data in our dataset, every column contain exactly 284807 rows.
- All data types are float64, except 1: Class
- All data types are float64, except 1: Class
- 28 columns have Sequential Names and values that don't make any logical sense - > V1, V2 ....V28
- 3 columns: TIME, AMOUNT and CLASS which can be analysed for various INSIGHTS!
- Memory Usage: 67.4 MB, not so Harsh !!

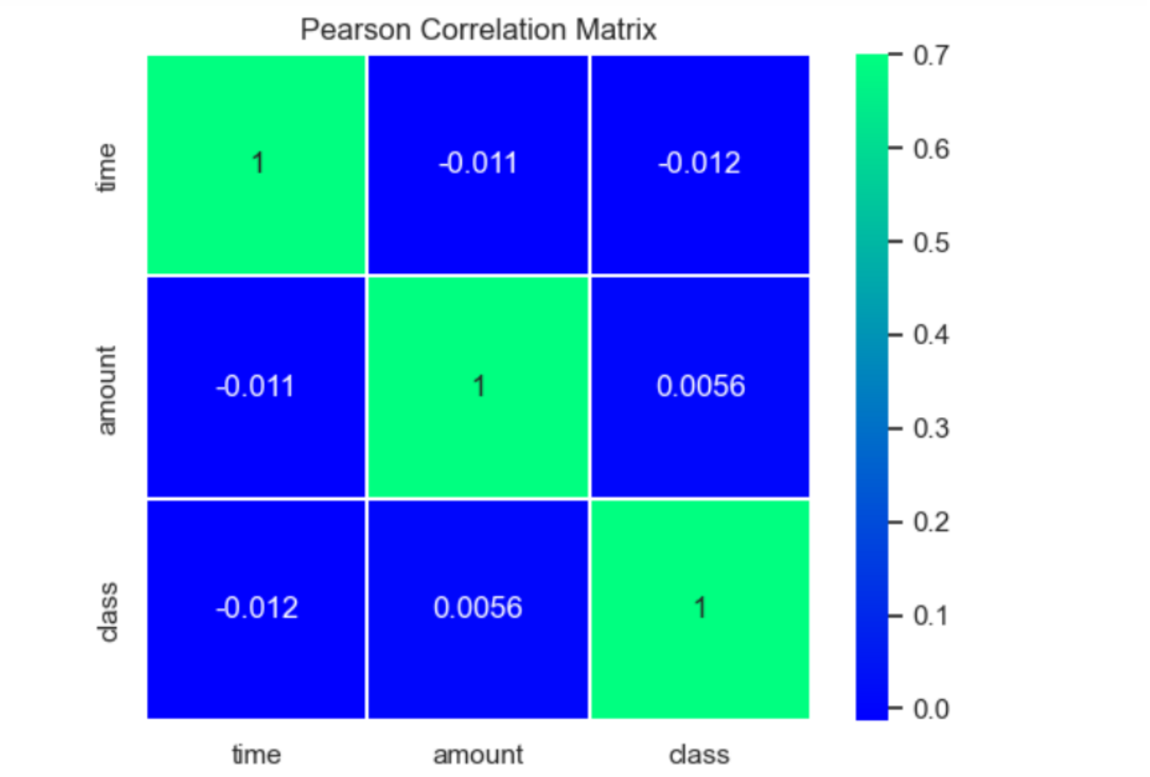
```
fig, axs = plt.subplots(ncols=2,figsize=(16,4))
sns.distplot(df[df['class'] == 1]['amount'], bins=100, ax=axs[0])
axs[0].set_title("Distribution of Fraud Transactions")

sns.distplot(df[df['class'] == 0]['amount'], bins=100, ax=axs[1])
axs[1].set_title("Distribution of Genuine Transactions")

plt.show()
```





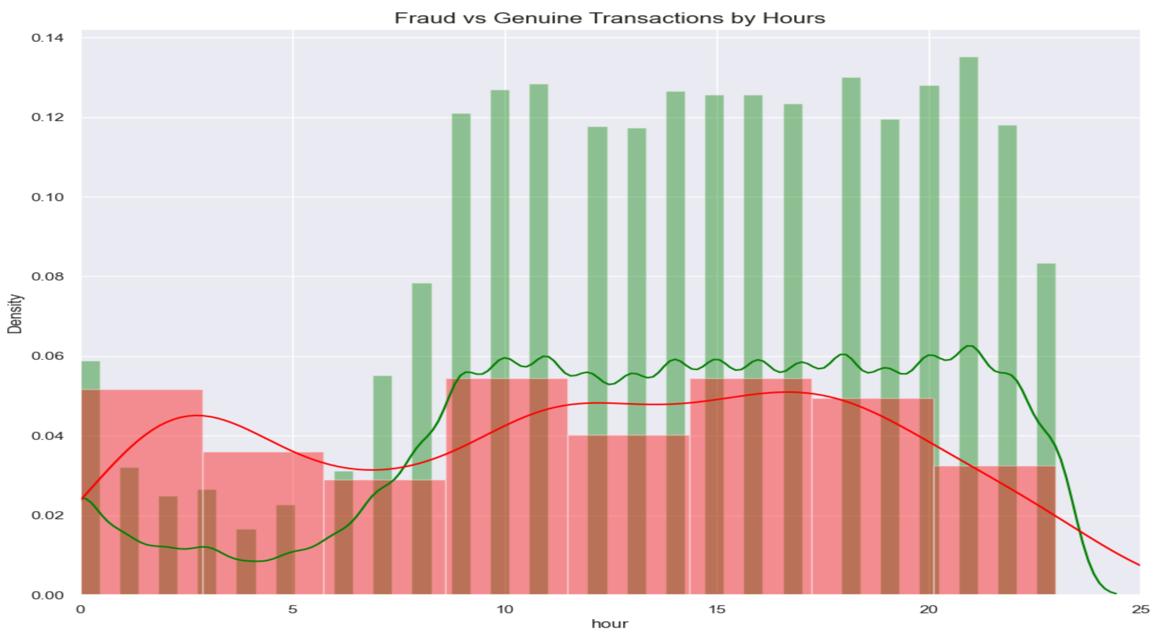


```
Highlights

It looks like that no features are highly correlated with any other features
```

Feature Engineering-

Create new features or transform the existing features for better performance of the ML Models



Model Selection

1 **Classification Models**

- Logistic Regression
- Decision Trees
- Random Forest
- Naive Bayes Classifier

2 **Class Imbalance Solutions**

- Under Sampling
- Over Sampling
- SMOTE
- ADASYN

3 **Metrics**

- Accuracy Score
- Confusion Matrix
- Precision Score
- Recall Score
- ROC_AUC
- F1 Score

In this project, we used a total of five classifications methods （**Logistic regression, KNN, Support vector machine (SVM), Decision tree (DT), Random forest)**. These classification algorithm methods are widely used for problems such as differential training dataset. Also, it commonly used in classification learning. That is the reason I compare them in the same training dataset. Also, it can be a cross-sectional comparison with other current studies in the final results.

Logistic regression algorithm

Use logistic regression to detect credit card fraud. Logistic regression is the classical and the best bicategorical algorithm which is preferred when dealing with classification problems, especially bicategorical ones. The choice of algorithm is based on the principle of simplicity before complexity. Logistic regression is also an excellent choice because it is a recognised statistical method used to predict the outcome of a binomial or polynomial. A multinomial logistic regression algorithm can regenerate the model. It will be a better classification algorithm when the target field or data is a set field with two or more possible values.

The advantage of logistic regression is that he is faster to process and is suitable for bicategorical problems. It is also more straightforward for any beginner to understand and directly see the weights of each feature. Then it is easier to update the model and incorporate new data for different problems (Aihua et al. 2007). Furthermore, it has a disadvantage. There is a limit to the data and the adaptability of the scene. Not as adaptable as the decision tree algorithm. But this is an issue that we can also determine in this project based on the actual situation whether the logistic regression has a better ability to adapt to an extensive data set of credit card transactions (Ng and Jordan 2002).

The main methods of logistic regression method:

Objective: It is to look for some risk factor, then in this project, They want to find a particular transaction factor or reasons that are suspected of being fraudulent

Prediction: Predicting the probability of fraud under other independent variables, based on different algorithmic models.

Judgment: It is somewhat similar to prediction. It is also based on different models to see how likely it is that a transaction is a risk factor in a situation where fraud falls into a specific category.

Regression General Steps

- Finding the h-function (i.e., the prediction function)

Constructing the predictive function h(x), the logistic function, or also known as the sigmoid function, we generally the first step is to build the predictive process, where the training data for the vector, as well as the best parameters. The basic form of the function shown in figure 1

$$g(z) = \frac{1}{1+e^{-z}}$$

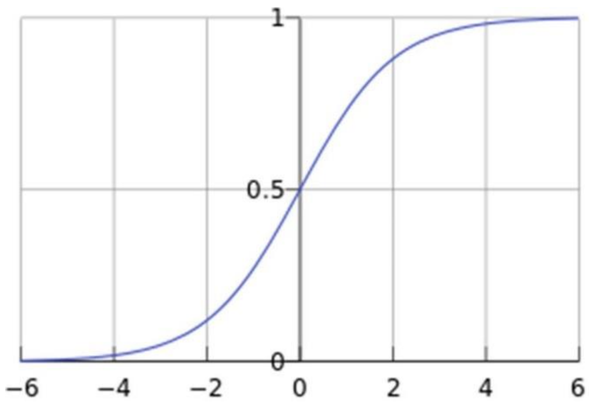


Figure 1: Logical function expressions

- Constructing the J-function (loss function)

The second step is that we need to construct the loss function-j. In general, there will be m samples, each with n characteristics. The Cost and J functions are as follows, and they are derived based on maximum likelihood estimation (Sahin and Duman 2011).

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x_i), y_i) = -\frac{1}{m}\left[\sum_{i=1}^{m}(y_i log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i)))\right]$$

Figure out how to make the J-function minimal and find the regression parameter (θ)

The final step is that we, using gradient descent, solve for the minimum value of θ. The process of updating θ can then be summarised as follows

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right) x_i^j$$

The process function of updating θ

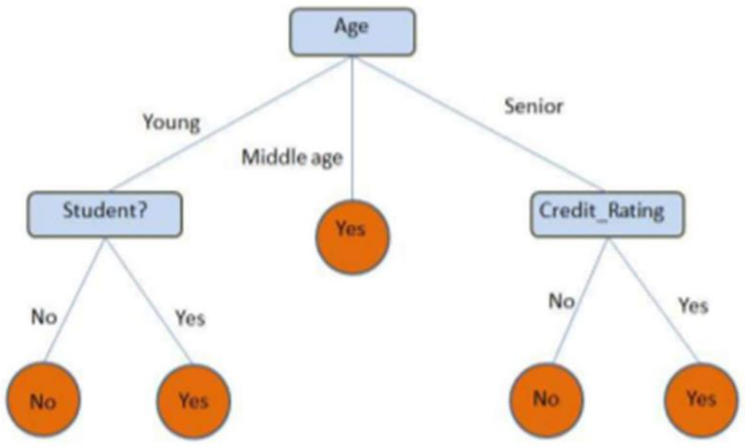
Decision Tree (DT)

The use of decision tree is usually based on the known probability of various scenarios, and the decision tree is formed to find the possibility that the expected net present value is greater than or equal to zero to evaluate the risk of the training project (Shen et al. 2007). Also, it judges the feasibility of the decision analysis method. Then we know that because this decision branch is drawn as a graph much like the trunk of a tree, we name it a decision tree.

Decision trees are a primary classification and regression method, and learning typically involves threesteps: feature selection, decision tree generation, and decision tree pruning.

In machine learning, a decision tree is a predictive model that represents a mapping between object properties and object values. Classification tree (decision tree) is a very commonly used classification method. Similar to the dataset classification problem mentioned in this paper, the decision tree is a technique that is often used to analyze data and can also be used to make predictions. That is why we chose it for the training of the fraud detection system (Şahin and Duman 2011).

That is a simple decision tree classification model: the red boxes are features.



A simple decision tree

People may be wondering why we chose a decision tree? There are two universal reasons: Decision trees usually mimic human horizontal thinking, so it is easy to understand the data we provide and make some excellent interpretations. Decision trees allow you to see the logic of howthe data is interpreted, unlike SVM, NN. And other similar black-box algorithms where you do not

see any internal information(Gaikwad et al. 2014).For example, as the figure above, we can see howthe logic makes decisions. Plain and simple.

Then, what is a decision tree now? A decision tree is kind like a tree which each node represents anelement (attribute), each link (branch) means a decision (rule), and each leaf represents a result (categorical or continuous value). The core of the entire decision tree is to create a tree-like this for the whole of the data. And the decision tree process individual results (or minimise errors in each leaf) on each plate.

k-nearest neighbour (KNN)

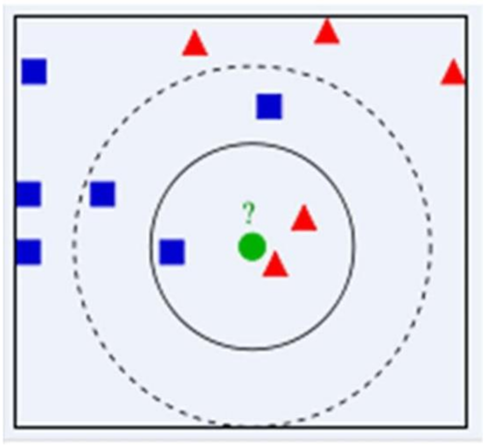
Initially proposed by Cover and Hart in 1968, Knn is a theoretically mature method that is one of the simplest of the data mining classification techniques. The term K nearest neighbours mean K nearest neighbours which says that its closest K neighbouring values can represent each sample. The nearest neighbour algorithm is a method of classifying every record in a data set.

The implementation principle of KNN nearest neighbor classification algorithm is: to determine the Category of unknown samples by taking all the examples of known types as a reference and at the same time calculate the distance between the new models and all the available pieces, from which the nearest K has known examples are selected, according to the rule of majority-majority-voting, the unknown samples(Bunsen et al. 2014) and the K nearest models belong to a category with more categories(Duman et al. 2013).

$$d((x_1,\ldots,x_n),(y_1,\ldots,y_n)) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p}$$

The K value of the KNN algorithm in 'scikit-learn' is adjusted by the n_neighbors parameter, and the default value is 5.

As shown in the figure below, *how do people determine which Category a green circle should belong to, whether it is a red triangle or a blue square?* If K=3, the green process will be judged to belong to the red triangle class because the proportion of red triangles is 2/3, and if K =5, the green circle will be considered to belong to the blue square class because the ratio of blue squares is 3/5(Gaikwad et al.2014).



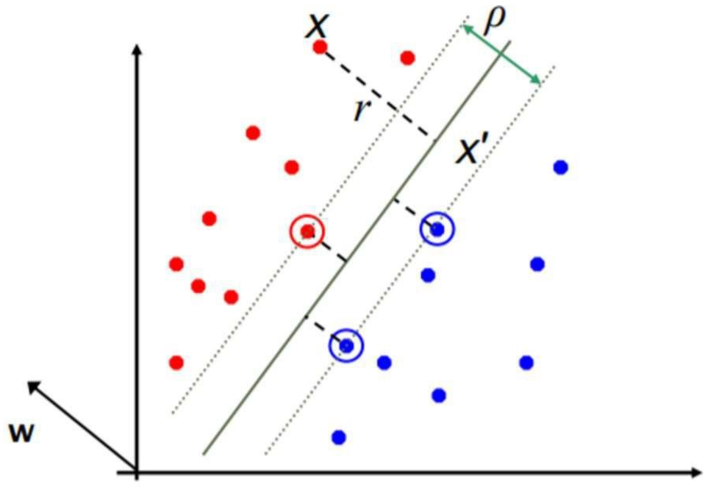
Support vector machine (SVM)

Support Vector Machine (often abbreviated as SVM) is a supervised learning method, most widely used in statistical classification and regression analysis. It is also the focus of this project. Support vector machines belong to a family of generalised linear classifiers which are characterised by their ability to both minimise empirical errors and maximise geometric edge regions. Hence support vectormachines are also known as maximum edge region classifiers.

The core principle of the support vector machine is: mapping the vectors into a higher dimensional space where a maximum spacing hyperplane is established. Two parallel hyperplanes are built on either side of the hyperplane that separates the data. Also, the separated hyperplanes maximise the distance between the two parallel hyperplanes (Singh et al. 2012). It is assumed that the greater the space or gap between the parallel hyperplanes, the smaller the total error of the classifier. In this project, SVM is the supervised learning algorithm used to solve the multi-class classification (Bhattacharyya et al. 2011).

Distance from example to the separator is $r = y\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$

Examples closest to the hyperplane are support vectors. Margin ρ of the separator is the width of separation between support vectors of classes.



Random forest

Random forests or **random decision forests** is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.[Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method,[2] which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Bremen and Adele Cutler, who registered"Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).[10] The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE is called Synthetic Minority Oversampling Technique which is an improvement of the random oversampling algorithm. The basic idea of the SMOTE algorithm is to analyse a small number of samples data and add new samples to the dataset based on the analysis of a small number of sample (Stolfo et al. 1997) s.

However, the class-imbalance problem that we need to solve next in this project refers to the uneven distribution of classes in the training set used in the training classifier (Pun 2011). For example, for a binary problem with 1000 training samples, ideally, the number of positive and negative models is similar; if there are 995 positive samples and only five negative samples, it means there is class- imbalance. There is also the case for the dataset in this project. We can see more details in section 3.3.

For now, there are three main approaches.

- Adjusting the value of θ

Adjust the value of θ according to the proportion of positive and negative samples in the training set. It is done based on the assumptions made about the training set, as described above. However, whether this assumption holds in the given task is open to discussion.

- Over sampling

The classes with a small number of samples inside the training set (few types) are oversampled, and new models are synthesized to mitigate class imbalance.

- Under sampling

Under-sampling of classes with a large number of samples inside the training set (most categories),discarding some examples to mitigate class imbalance (Dal Pozzolo et al. 2015).

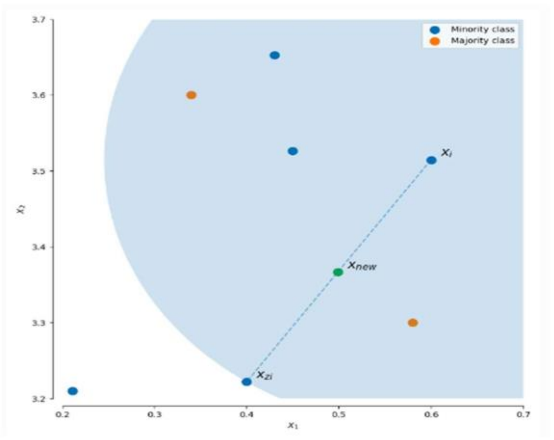
In this project, we use oversampling and under sampling to perform comparison operations. At the same time, we can also compare the results to analyse whether the two methods are more suitable forthis project's dataset, and what are the advantages and disadvantages of each technique.

The core idea of SMOTE (synthetic minority oversampling technique) in a nutshell is to interpolate between minority class samples to generate additional models. For example, for a minority sample xi use the k-nearest neighbour method (k values need to be specified in advance) to find the k nearest minority samples to xi (Sahin et al. 2013). The distance is defined as the Euclidean distance in the n-dimensional feature space between the models. One of the k nearest neighbours is then randomly selected to generate a new sample using the following formula (Han et al. 2005).

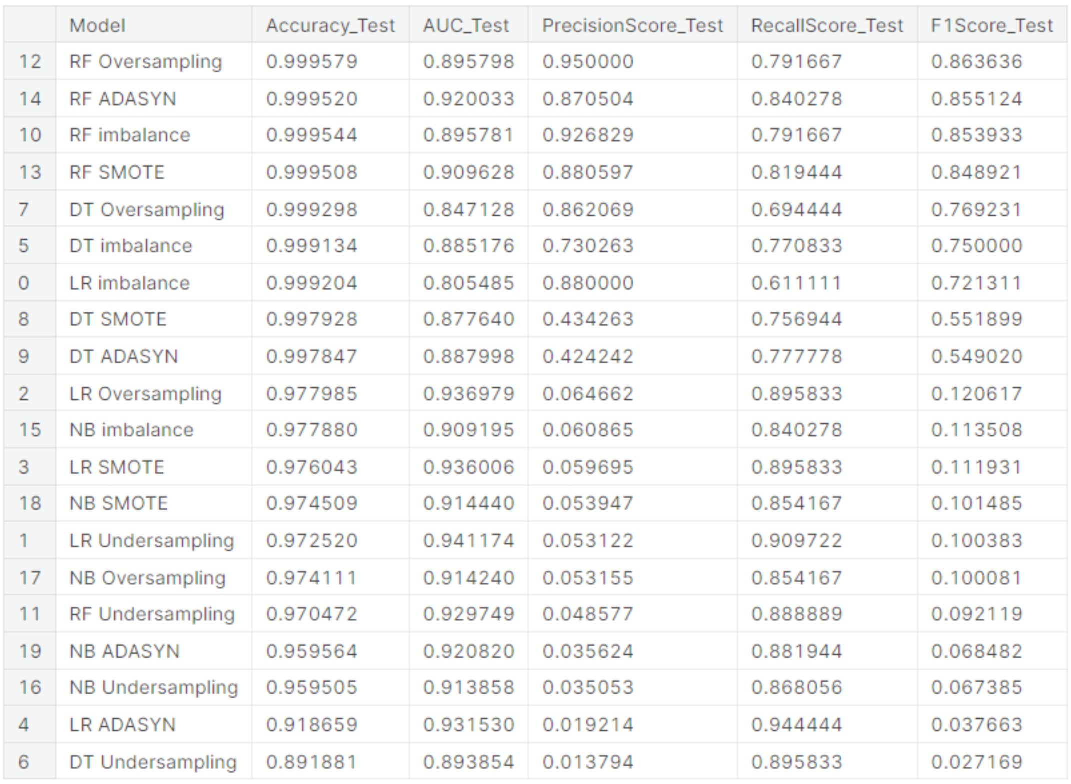
$$\mathbf{x}_{new} = \mathbf{x}_i + (\hat{\mathbf{x}}_i - \mathbf{x}_i) \times \delta$$

Where x^ is the elected k-nearest neighbour point, and δ ∈ [0,1] is a random number. An example ofa SMOTE-generated sample, using 3-nearest neighbours, is shown in the following figure which shows that the SMOTE-generated model generally lies on the line connected by xi and x^i.

The example of SMOTE formation sample



Performance measures of various classifiers



	Model	Accuracy_Test	AUC_Test	PrecisionScore_Test	RecallScore_Test	F1Score_Test
12	RF Oversampling	0.999579	0.895798	0.950000	0.791667	0.863636
14	RF ADASYN	0.999520	0.920033	0.870504	0.840278	0.855124
10	RF imbalance	0.999544	0.895781	0.926829	0.791667	0.853933
13	RF SMOTE	0.999508	0.909628	0.880597	0.819444	0.848921
7	DT Oversampling	0.999298	0.847128	0.862069	0.694444	0.769231
5	DT imbalance	0.999134	0.885176	0.730263	0.770833	0.750000
0	LR imbalance	0.999204	0.805485	0.880000	0.611111	0.721311
8	DT SMOTE	0.997928	0.877640	0.434263	0.756944	0.551899
9	DT ADASYN	0.997847	0.887998	0.424242	0.777778	0.549020
2	LR Oversampling	0.977985	0.936979	0.064662	0.895833	0.120617
15	NB imbalance	0.977880	0.909195	0.060865	0.840278	0.113508
3	LR SMOTE	0.976043	0.936006	0.059695	0.895833	0.111931
18	NB SMOTE	0.974509	0.914440	0.053947	0.854167	0.101485
1	LR Undersampling	0.972520	0.941174	0.053122	0.909722	0.100383
17	NB Oversampling	0.974111	0.914240	0.053155	0.854167	0.100081
11	RF Undersampling	0.970472	0.929749	0.048577	0.888889	0.092119
19	NB ADASYN	0.959564	0.920820	0.035624	0.881944	0.068482
16	NB Undersampling	0.959505	0.913858	0.035053	0.868056	0.067385
4	LR ADASYN	0.918659	0.931530	0.019214	0.944444	0.037663
6	DT Undersampling	0.891881	0.893854	0.013794	0.895833	0.027169

Hightlights

After training each of the models, these are the final results. All of the scores for Random Forest with Oversampling technique and the Random Forest with SMOTE technique models are very promising for our dataset! Each model has a high true positive rate and a low false-positive rate, which is exactly what we're looking for.

In the ROC graph above, the AUC scores for Random Forest with Oversampling technique is pretty high, which is what we'd like to see. As we move further right along the curve, we both capture more True Positives but also incur more False Positives. This means we capture more fraudulent transactions, but also flag even more normal transactions as fraudulent. **So Random Forest with Oversampling technique is our final model, as this gives highest F1 score of 86.47% on test datasets.**

Grid Search

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified.

A **model hyperparameter** is a characteristic of a model that is external to the model and whose value cannot be estimated from data. The value of the hyperparameter has to be set before the learning process begins. For example, c in Support Vector Machines, k in k-Nearest Neighbors, the number of hidden layers in Neural Networks.

In contrast, a **parameter** is an internal characteristic of the model and its value can be estimated from data. Example, beta coefficients of linear/logistic regression or support vectors in Support Vector Machines.

Result and Conclusion

We are pre-processing and feature engineering scales and selects features, and uses the smote algorithm (undersampling and downsampling) to deal with the unbalance of the data set. Then we build an anti-fraud prediction model based on the five algorithms: **Logistic regression, KNN, Support vector machine (SVM), Decision tree (DT), Random Forest**. The model can predict whether a user has made fraudulent purchases.

Then we used a confusion matrix to compare the results of the two sampling methods. The best solution is logistic regression (undersampling) which is more in line with our expectations. It also achieves an accuracy of 97.00%. Then although credit card spoofing detection, most of the current research is still using decision tree and logistic regression test. But in this project, I think two points where we added SVM and universal algorithm Random Forest, to make training comparison together. I also believe meaningful results emerged. Did not Random Forest perform poorly, and also we dealt with the sample imbalance problem to get significant marks.