MD Wazid Ansari (DTU)

CREDIT CARD FRAUD DETECTION

I. PROBLEM STATEMENT

We all are bounded by technologies. If we want to talk to someone then we have social media applications like gmail,facebook,etc and if we want to pay our bills then for that we can use google pay, paytm, paypal etc while using these technologies our life became very easy, but everything has their pros and cons.
The technologies that we are using can be accessed by other peoples as well because people are very intelligent nowadays. While we are paying and depositing our money in the bank, someone can monitor our transactions.To secure our transactions from these kinds of people is our main concern for this study.

In this paper, our main focus of study is on credit cards. About 3 billion people use credit cards because credit cards make life easier. But there are certain people who can actually use these cards for fraud.Technologies can be hacked or misused by people and this is a very critical topic to discuss.

To prevent these frauds, we will use machine learning algorithms that will allow the users to use the credit card without any problem.If any ambiguous outlier is detected by our algorithm, it will block the user from doing fraud.
In this project we will try to find the best machine learning model for credit card fraud detection.

II. OBJECTIVES

The main objective of this paper is to analyze which machine learning algorithm performs well for fraud detection and then find the best Machine Learning algorithm for credit card fraud detection.

III. METHODOLOGY

In this paper we will use machine algorithms like logistic regression, knn classifier, svm classifier, decision tree classifier, random forest classifier and then we will check accuracy score to find the best model for fraud detection.

Right now many websites are available for collecting the Data like - kaggle, Socrata, Datasets on Github, Google Public Datasets,Data.gov etc.

Even learning websites like geeksforgeeks,hackerearth, greatlearning, etc provide data for learning and practice. Here, we will use a dataset from one of these learning websites known as hackerearth.
Recently hackerearth launched a competition on Data Science where they provided data for Credit card fraud detection and we are going to use that data in this paper.

Steps involved in this paper for prediction:

1. Downloading data
2. Fixing null values
3. Splitting into training and test set
4. Feature Reduction
5. Encoding categorical feature
6. Feature Scaling
7. Modeling our data

1. **Downloading data**

   First step is to import our data into the IDE. We are using python for the analysis because python has a lot of useful libraries which allow us to code the problem without explicitly coding everything.

   Libraries that we are going to use are:

   ```
   import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
   import seaborn as sns

   from sklearn.compose import ColumnTransformer
   from sklearn.preprocessing import OneHotEncoder
   from sklearn.preprocessing import StandardScaler
   ```

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv('train.csv')      # Importing Data
```

Feature exploration

We have 19 columns and 43,508 rows.
In our dataset, a good amount of columns are present. Let's take a look into these columns and check if we can assume something about these columns.

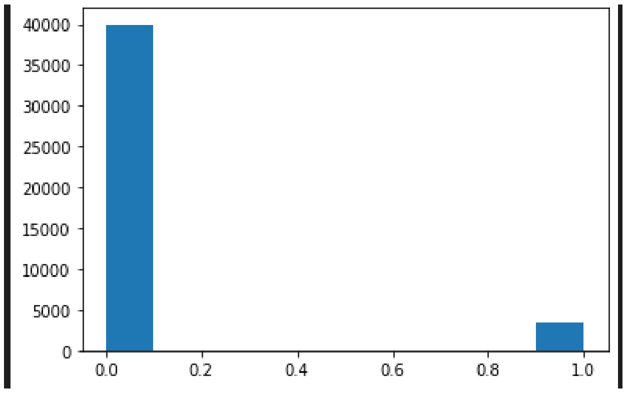
```
df.info()                          # Looking for columns
```

These are the columns that we have to deal with.

0	customer_id	5	owns_house
1	name	6	no_of_children
2	age	7	net_yearly_income
3	gender	8	no_of_days_employed
4	owns_car	9	occupation_type
10	total_family_members	15	credit_score
11	migrant_worker	16	prev_defaults
12	yearly_debt_payments	17	default_in_last_6months
13	credit_limit	18	credit_card_default
14	credit_limit_used(%)		

We have 19 columns out of which credit_card_default is our target feature containing 0 and 1 value.
Class 0 represents valid transactions while class 1 represents fraud transactions.

Fraud Vs Valid Transactions visualization



Above figure shows fraud transactions are very less as compared to valid transactions.Therefore, the outliers are the real culprit here for doing fraud, So we cannot just remove these outliers.

2. **Fixing null values.**

   Null values can reduce the performance of our model so in order to increase the performance of our model, we have to handle these null values.

   Our dataset contains null values but in very less quantity.

```
df.isnull().mean()*100      # Looking for percentage of missing values
```

customer_id	0.00%
name	0.00%
age	0.00%
gender	0.00%
owns_car	1.20%
owns_house	0.00%
no_of_children	1.70%
net_yearly_income	0.00%
no_of_days_employed	1.017%
occupation_type	0.00%
total_family_members	0.18%
migrant_worker	0.19%
yearly_debt_payments	0.20%
credit_limit	0.00%
credit_limit_used(%)	0.00%
credit_score	0.017%
prev_defaults	0.00%
default_in_last_6months	0.00%

```
credit_card_default                0.00%
```

Since our dataset doesn’t contain missing values more than 2 % so it is safe to remove these rows.

```
df = df.dropna()                        # Deleting missing rows
```

Splitting our data into dependent and independent variable

For prediction, we need to declare a target variable and input variable, for that we are going to split our data into X and Y.

```
X = df.iloc[:,-1]                       # Independent Variable
Y = df.iloc[:,18]                       # Dependent Variable
```

3. **Splitting our data into training and test set**

Now we will split our data for training and testing. Our 80% of data will train and on the remaining 20% of data we will try to predict the values and calculate accuracy score for the prediction.

```
X_train,X_test,y_train,y_test = train_test_split(
                                        X,Y,test_size=0.2,
                                        random_state = 42)
```

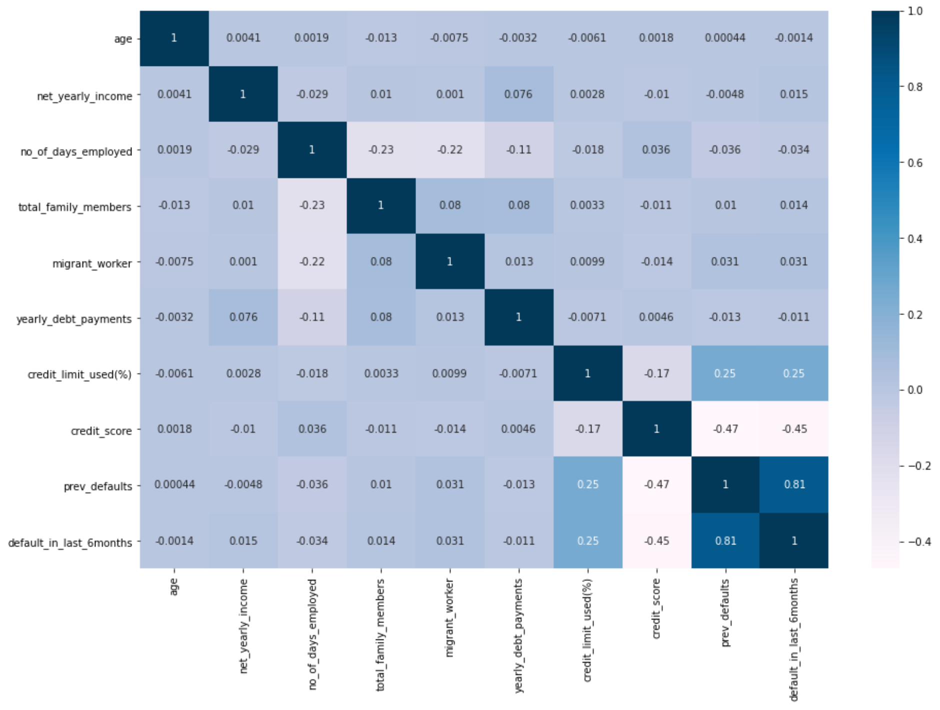
Now that we have so many features to deal with, we can ease our work by reducing some features.Even irrelevant features can actually reduce the performance of a machine learning model.For example : KNN algorithm works of nearest neighbour logic, if it find the nearest value from the irrelevant column then our model performance will decrease. So, it's a good practice to reduce the columns.

Let’s see the relation of every column with each other, for that we will use correlation heatmap.

Correlation basically tells how one feature is changing with another feature.Its value ranges from -1 to 1.If it is +1 then it means one feature is increasing with another feature and if the value is -1 then one feature is decreasing while the other one is increasing.

4. **Feature Reduction**

```
sns.heatmap(X_train.corr(),cmap='PuBu',annot=True)
```



From the above matrix**,** the correlation value between:

- no_of_children & total_family_member > 0.8
- net_yearly_income & credit_limit > 0.8
- prev_defaults & default_in_last_6months > 0.8

(Here 0.8 is our threshold value, above which is strong positive correlation)

So it is safe to remove 3 columns from above 6 columns.

Our dataset contains 3 data type columns:

1. object
2. int64
3. float64

5. Handling categorical Data.

- Since a machine learning model cannot handle categorical features so we need to encode categorical features.
- Our dataset contains nominal categories features so we will use OneHotEncoder to encode our dataset.

```
Ohe = OneHotEncoder(sparse=False,handle_unknown=’ignore)
Ohe.fit_transform(X_train,y_train)
Ohe.fit(X_test,y_test)
```

6. Scaling categorical Data.

It is necessary to scale our data because some machine learning algorithms like Logistics regression,KNN,SVM uses distance calculation and If we don’t scale our data then the bigger value will dominate the lower value by huge difference and our model will not perform properly.

```
sc = StandardScaler()
sc.fit(X_train,y_train)
```

Now our dataset is ready for training.
We will now call algorithms

7. Modeling our Data

Since we are using python, it is easy to deploy the machine algorithm libraries without explicitly programming.
Now we are creating objects for each Algorithm class.

```
lr = LogisticRegression()
rf = RandomForestClassifier()
kn = KNeighborsClassifier()
svm = SVC()
dt = DecisionTreeClassifier()
```

Now that we have created the objects, we can fit these algorithms in our training set.
Fitting these objects using fit() function

```
lr.fit(X_ftrain,y_train)
rf.fit(X_ftrain,y_train)
kn.fit(X_ftrain,y_train)
svm.fit(X_ftrain,y_train)
dt.fit(X_ftrain,y_train)
```

Predicting test results

Now we are going to calculate the value of dependent variable:

```
y_pred1 = lr.predict(X_ftest)
y_pred2 = kn.predict(X_ftest)
y_pred3 = svm.predict(X_ftest)
y_pred4 = dt.predict(X_ftest)
y_pred5 = rf.predict(X_ftest)
```

III. Results

Calculating R2 Coefficient of determination :

Now, we are going to see how well our model is performing based on the R2 coefficient of determination.

It basically tells how our predicted values are varying from the mean of actual value.

R2 = 1 - SSR/ SSM

SSR = sum(actual - predicted)2

SSM = sum(actual - mean)2

ulating Accuracy score of each algorithm

```
print("Logistic Regression        : ",round(accuracy_score(y_pred1,y_test)*100,2))
print("KNN Classifier score       : ",round(accuracy_score(y_pred2,y_test)*100,2))
print("SVM Classifier score       : ",round(accuracy_score(y_pred3,y_test)*100,2))
print("Decision Classifier score : ",round(accuracy_score(y_pred4,y_test)*100,2))
print("Random Classifier score : ",round(accuracy_score(y_pred5,y_test)*100,2))
```

```
Logistic Regression        :  95.69
KNN Classifier score       :  91.01
SVM Classifier score       :  91.51
Decision Classifier score  :  97.13
Random Classifier score    :  97.91
```

IV-. Conclusions

In this paper we have talked about how we can reduce the misuse of credit cards and for that we used various machine learning algorithms.

From the results obtained from the accuracy score we can conclude that the random forest and decision tree algorithm performs really well as compared to other algorithms. Logistic regression is also a good algorithm for fraud detection.

Random Forest > Decision Tree > Logistic regression > SVM > KNN

Hence, Owners should use Random forest or Decision Tree algorithm for credit card fraud detection.