Machine Learning Algorithms

Detecting Credit Card Fraud

Supervised by

Dr: Sara Sweidan

Eng: Sahar Mohamed

Detecting Credit Card Fraud

- **Team Leader**
  - Hazem Refai Mohamed Refai.
- **Team Member**
  - Khaled Ahmed Slama Eldifrawy.
  - Eslam Khaled Soliman Gad.
  - Ibrahim Mousa El-Sayed Habib.
  - Ahmed Samer Eid Abdul-Hamid.

Detecting Credit Card Fraud

❑ Agenda

- Project Idea.
- Project Goals.
- The Problem Before and After the Availability of ML to Make the Protection System.
- Dataset Description:-
- How to build ML System.
- Calculate Accuracy

Detecting Credit Card Fraud

❑ Project Idea…?

- **Detecting Credit Card Fraud** In light of the increasing number of customers using credit cards, as well as their information in the **bank system**, the bank must provide a good protection system.
- After a lot of research, they found that there is no alternative to entering **Machine learning** into the security system. This is a large number of customers and their information, but the model has gone through many stages in order to reach what we want and these stages are now clear with us.

Detecting Credit Card Fraud

❑ The Project Goal:-

- The goal of the project Here is two sides of the same coin, that is, to provide protection for the bank as well as the customer.

This is done by :-

1. Protecting the customer from any fraud or hacking to preserve his data
2. Protecting the bank from any losses in the event of any fraud
3. Provides an exchange of trust between customers and the bank
4. Increases the number of customers at the bank that uses the fraud against protection system

Detecting Credit Card Fraud

❑ The Problem Before and After the Availability of ML to Make the Protection System:-

- **Before a good protection system** was available by banks to their customers first, fraud was bigger and easier because there was not as much information available as now, and it was difficult for banks to save all this data without making mistakes, **and these mistakes turn led to disasters, especially for banks.**
- **After the availability of machine learning** the greater the number of users, the greater the fraud, and this requires the intervention of ML to solve this problem by learning from previous frauds, so that the same error will not occur again, and **The Greater the Safety Rate, The Lower the Fraud Rate**

Detecting Credit Card Fraud

❑ Dataset Description:-

- **The Dataset** contains transactions made by credit cards in September 2013 by European cardholders. We have 492 frauds out of 284,807 transactions.
- It contains only numerical input variables which are the result of a **PCA** transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features **V1, V2, … V28** are the principal components obtained with **PCA**, the only features which have not been transformed with **PCA** are 'Time' and 'Amount'. Feature ‘Time’ contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature ‘Amount’ is the transaction Amount.

Detecting Credit Card Fraud

How to build ML System:-

- Dataset and Application Of Some Operations on It:-

The Dataset consists of a large group of thousands of customers. Each customer has its own set of data called **Features**. The dataset consists of 30 columns. Each column carries a specific feature about each customer. Next to the set of features, there is **Class** for each customer that shows whether the customer was frauded or not. For fraud or not, and this is what we know easily, because if the customer was subjected to Fraud before, **The Model outputs 1 if fraud process** and **The Model outputs 0 if not frauded process**

Detecting Credit Card Fraud

How to build ML System “Cont.”

1. First import what library to read your dataset through import pandas and give it the path.

```
# Reading The Dataset By Using Pandas
credit_card_ds = pd.read_csv('creditcard.csv')
```

2. Then call the dataset info to describe the data it gives you if your data contains Null or no , the number of columns, the index of the rows, and the data ok for each column.

```
#dataset informations
credit_card_ds.info()
```

3. Check if there is a miss value or not, and then count the number of miss value in each column

```
#Before using dataset we need t
credit_card_ds.isnull().sum()
```

Detecting Credit Card Fraud

How to build ML System “Cont.”

4. If there is a Miss Value, we calculate the average for it and delete the entire row.

```
#We Watch that V1 Include 24 Missing value ,We need to get rid of missing value
credit_card_pre_null=credit_card_ds.fillna(value=credit_card_ds['V1'].mean())
credit_card_pre_null
```

5. And then find out how many zero and one, or in other words, how many legal and illegal cases in the data set.

```
#we need in dataset number of 0 and 1, y
credit_card_ds['Class'].value_counts()
```

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

Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

6. We separate the data for the X-train and the E-train until we insert it into the model

```
X = credit_card_ds.drop(columns='Class', axis=1
Y = credit_card_ds['Class']
pd.DataFrame(X)
pd.DataFrame(Y)
```

7. One of the most important operations that we used on the dataset is to import the StandardSclear function from the Preprocessing library in order to approximate the extent of the data from (-1:1)

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_new= sc.fit_transform(X)
pd.DataFrame(X_new)
```

Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

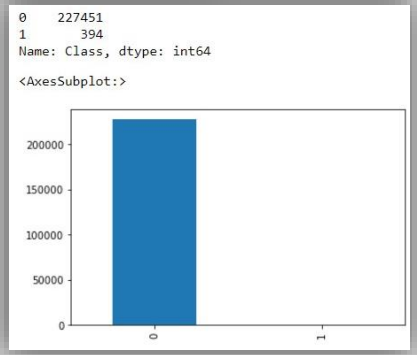
8. We separate the data for the **X-train** and the **Y-train** until we insert it into the model, we divide 0.20 by the test **X-test** and **Y-test** percentage and the rest of the ratio is train.

```
X_train, X_test, Y_train, Y_test = train_test_split(X_new , Y , test_size=0.20, stratify=Y, random_state=0)
```

9. In this step, we will see how many and how many are zero,
the model will be trained on them and we show them, we see that the number of one is much less than zero, and therefore we must see a solution

```
print(Y_train.value_counts())
Y_train.value_counts().sort_index().plot.bar()
```

```
0    227451
1       394
Name: Class, dtype: int64
```



Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

10. And in order to equal the number 1 and 0, this method is done in two stages or a stage using an **under sampling**, and this reduces the large percentage to twice the small percentage, and then we do a **over sampling** to increase the small percentage to be equal to the small percentage.

```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(sampling_strategy=0.5, random_state=0)

X_train_rus, Y_train_rus = rus.fit_resample(X_train,Y_train)

print(Y_train_rus.value_counts())

from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(sampling_strategy=1.0, random_state=0)

X_train_balanced, y_train_balanced = ros.fit_resample(X_train_rus, Y_train_rus)

#print(y_train_rus1.value_counts())
print(y_train_balanced.value_counts())
y_train_balanced.value_counts().sort_index().plot.bar()
```

```
0    788
1    394
Name: Class, dtype: int64
0    788
1    788
Name: Class, dtype: int64
```

Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

11. We call the algorithm from its library and then dividing the data into Train and a Test, the models do the training by taking **X-train** and the **Y-train.** Calculating the training percentage and its quality in learning if the model trains well and predicts bad results in the test.

- **Logistic-Regression.**
- **Naïve-Bayes**
- **Support Vector Classification (SVC)**
- **K-NearestNeighbour(Knn)**
- **Random-Forest**

Detecting Credit Card Fraud

How to build ML System “Cont.”

- **Logistic-Regression.**

```
from sklearn.linear_model import LogisticRegression #import library NaiveBayes
model1 = LogisticRegression()
model1.fit(X_train_balanced, y_train_balanced )
X_train_prediction1 = model1.predict(X_train_balanced)
training_data_accuracy1 = accuracy_score(X_train_prediction1, y_train_balanced)
print('Accuracy on Training data_LogisticRegression : ', training_data_accuracy1)
X_test_prediction1 = model1.predict(X_test)
test_data_accuracy1 = accuracy_score(X_test_prediction1, Y_test)
print('Accuracy score on Test Data LogisticRegression: ', test_data_accuracy1)
cm1 = confusion_matrix(Y_test, X_test_prediction1)
print('LogisticRegression:')
print(Y_test.shape)
pd.DataFrame(cm1)
```

It is used for predicting the categorical dependent variable using a given set of independent variable. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Regression problems, whereas Logistic regression is used for solving the classification problems. It use the sigmoid function is a mathematical function used to map the predicted values to probabilities. in logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. This is what serves our model, **So we used it in our Model**

Detecting Credit Card Fraud

How to build ML System “Cont.”

- **Naïve-Bayes**

The probability of an event is estimated from the observed data by dividing the number of trials in which the event occurred by the total number of trials. The probability of all the possible outcomes of a trial must always sum to 1.

```
from sklearn.naive_bayes import GaussianNB # import library naive_bayes
model2 = GaussianNB()
model2.fit(X_train_balanced, y_train_balanced )
X_train_prediction2 = model2.predict(X_train_balanced)
training_data_accuracy2 = accuracy_score(X_train_prediction2, y_train_balanced)
print('Accuracy on Training data_NaiveBayes : ', training_data_accuracy2)
X_test_prediction2 = model2.predict(X_test)
test_data_accuracy2 = accuracy_score(X_test_prediction2, Y_test)
print('Accuracy score on Test Data NaiveBayes : ', test_data_accuracy2)
cm2 = confusion_matrix(Y_test, X_test_prediction2)
print('NaiveBayes')
pd.DataFrame(cm2)
```

The Naive Bayes algorithm describes a simple method to apply Bayes' theorem to classification problems. Although it is not the only machine learning method that utilizes Bayesian methods, it is the most common one, Since we are making a model based on the Knife Base, it is also compatible with the dataset and has an accuracy rate that is not bad **So we used it in our Model**

Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

- **Support Vector Classification (SVC)**

```
from sklearn.svm import SVC #import library Support Vector machine
model3 = SVC(kernel='linear', random_state = 0, C=10)
model3.fit(X_train_balanced, y_train_balanced )
X_train_prediction3 = model3.predict(X_train_balanced)
training_data_accuracy3 = accuracy_score(X_train_prediction3, y_train_balanced)
print('Accuracy on Training data_SVC : ', training_data_accuracy3)
X_test_prediction3 = model3.predict(X_test)
test_data_accuracy3 = accuracy_score(X_test_prediction3, Y_test)
print('Accuracy score on Test Data SVC : ', test_data_accuracy3)
cm3 = confusion_matrix(Y_test, X_test_prediction3)
print('SVC')
pd.DataFrame(cm3)
```

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector

it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the nature. This best decision boundary is called a hyperplane. **Also, this technique serves the model and is compatible with the data set**

Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

- **K-NearestNeighbour(Knn)**

Classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of similar labeled examples. The letter k is a variable term implying that any number of nearest neighbors could be used.

```
#import library K Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
model4 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
model4.fit(X_train_balanced, y_train_balanced )
X_train_prediction4 = model4.predict(X_train_balanced)
training_data_accuracy4 = accuracy_score(X_train_prediction4, y_train_balanced)
print('Accuracy on Training data_Knn : ', training_data_accuracy4)
X_test_prediction4 = model4.predict(X_test)
test_data_accuracy4 = accuracy_score(X_test_prediction4, Y_te
print('Accuracy score on Test Data  Knn : ', test_data_accuracy4)
cm4 = confusion_matrix(Y_test, X_test_prediction4)
print('Knn')
pd.DataFrame(cm4)
```

Where relationships among the features and the target classes are numerous, complicated, or extremely difficult to understand, yet the items of similar class type tend to be fairly homogeneous. On the other hand, if the data is noisy and thus no clear distinction exists among the groups, the nearest neighbor algorithms may struggle to identify the class boundaries. **And when we did the Preprocessing StandardSclear, this made Knn one of our most prominent models.**

Detecting Credit Card Fraud

❑ How to build ML System “Cont.”

- **Random-Forest**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning

```
from sklearn.ensemble import RandomForestClassifier
model5 =  RandomForestClassifier(n_estimators = 5 , criterion = 'entropy', random_state = 10)
model5.fit(X_train_balanced, y_train_balanced )
X_train_prediction5 = model5.predict(X_train_balanced)
training_data_accuracy5 = accuracy_score(X_train_prediction5, y_train_balanced)
print('Accuracy on Training data_RandomForest : ', training_data_accuracy5)
X_test_prediction5 = model5.predict(X_test)
test_data_accuracy5 = accuracy_score(X_test_prediction5, Y_test)
print('Accuracy score on Test Data RandomForest : ', test_data_accuracy5)
cm5 = confusion_matrix(Y_test, X_test_prediction5)
print('RandomForest : ')
pd.DataFrame(cm5)
```

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning**. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. He is also good at dealing with our problem**

Detecting Credit Card Fraud

❑ Calculate Accuracy:-

- **We calculate the test in two ways:-**
  - ✓ Calculcate Confusion_Matrix.
  - ✓ Calculcate Accuracy score.

```
Accuracy on Training data_LogisticRegression :  0.9524111675126904
Accuracy on Training data_NaiveBayes :  0.9086294416243654
Accuracy on Training data_SVC :  0.9644670050761421
Accuracy on Training data_Knn :  0.9676395939086294
Accuracy on Training data_RandomForest :  0.9974619289340102

Accuracy score on Test Data LogisticRegression:  0.9696113198272532
Accuracy score on Test Data NaiveBayes :  0.9757557670025631
Accuracy score on Test Data SVC :  0.9693830975035989
Accuracy score on Test Data  Knn :  0.9655910958182649
Accuracy score on Test Data RandomForest :  0.9729819879919946
```

```
cm5 = confusion_matrix(Y_test, X_test_prediction5)
print('RandomForest : ')
pd.DataFrame(cm5)
```

RandomForest :

	0	1
0	55335	1529
1	10	88

```
cm4 = confusion_matrix(Y_test, X_test_prediction4)
print('Knn')
pd.DataFrame(cm4)
```

Knn

	0	1
0	54913	1951
1	9	89

```
cm1 = confusion_matrix(Y_test, X_test_prediction1)
print('LogisticRegression:')
print(Y_test.shape)
pd.DataFrame(cm1)
```

LogisticRegression:
(56962,)

	0	1
0	55145	1719
1	12	86

Detecting credit card fraud

THANK YOU