Credit Card fraud detection using Machine Learning

Credit Card fraud:

Credit card fraud can be defined as a crime where a person uses another person's credit card for personal reasons while the cardholder and issuing authorities do not know that the card is being used. Due to the rise and speed of E-Commerce, there has been a massive use of credit cards in online purchases which has led to a high number of credit card-related frauds. When doing digitalization, the need to identify credit card fraud is required. Fraud detection involves monitoring and analyzing the performance of various users to measure the detection or avoidance of undesirable behavior. To detect successful credit card fraud, we need to understand the various technologies, algorithms and types involved in detecting credit card fraud. The algorithm can distinguish fraudulent transactions or not. Find out the fraud, they need to transfer the data and the knowledge of the fraudulent activity. They analyze the database and classify everything that is done. Fraud detection includes monitoring users' activities to balance, detect or avoid undesirable behavior, which includes fraud, intrusion and error. Machine learning algorithms are used to analyze all authorized transactions and suspicious reporting. These reports are investigated by experts who contact cardholders to verify that the transaction was genuine or counterfeit.

This implementation provides a response to an automated system used to train and update the machine learning algorithm to ultimately improve the performance of detecting fraud over time as follows.

Import the necessary packages and dataset:

Such as numpy , pandas, matplotlib and read dataset “creditcard.csv”

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv('creditcard.csv')#available on kaggle
print(data.shape)
data.head()
```

Output: Overview of dataset

```
(284807, 31)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267

5 rows × 31 columns

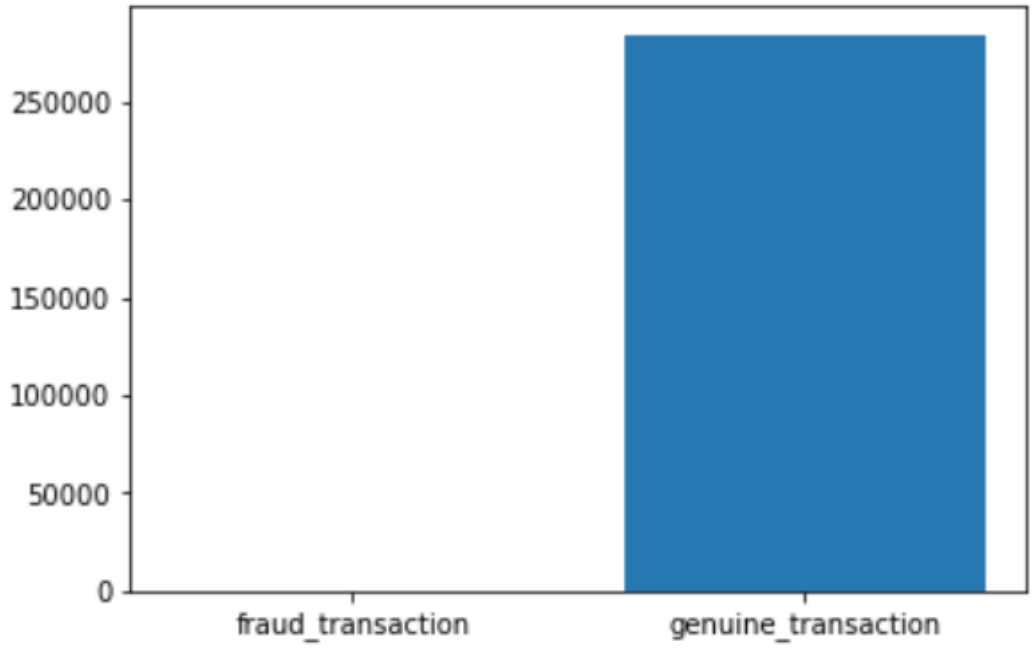
Plot and count Fraud Cases and Genuine Cases:

Visualize Fraud Cases and Genuine Cases

```
fraud_transaction=len(data[data["Class"]==1])
genuine_transaction=len(data[data["Class"]==0])
print("Number of Fraud Transactions",fraud_transaction)
print("Number of Genuine Transactions",genuine_transaction)
case = ['fraud_transaction', 'genuine_transaction']
count = [fraud_transaction,genuine_transaction]
plt.bar(case,count)
plt.show()
```

Output:

```
Number of Fraud Transactions 492
Number of Genuine Transactions 284315
```



From the above count we can conclude that the dataset is highly imbalanced as very few fraud cases in comparison with genuine cases.

We need to use here ensemble technique(Random Forest Classifier) to classify cases.

Splitting dataset:

Splitting dataset as features(x) and target(y) .

```
x=data.drop(['Class'],axis=1)
```

```
x.shape
y=data['Class']
y.shape

x_data=x.values
y_data=y.values
```

Split further for training and testing purposes.

```
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test=tts(x_data,y_data,test_size=0.25,
random_state=0)
```

Apply model:

Model creation and training model on dataset

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier()

model.fit(x_train,y_train)
```

Output:

```
RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=
1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_sp
lit=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=
None, ccp_alpha=0.0, max_samples=None)
```

Prediction:

Predict the output of testing dataset as follows

```
y_pred=model.predict(x_test)
```

Evaluation and accuracy check:

Accuracy score checked using sklearn.metrics package

```
from sklearn.metrics import confusion_matrix,accuracy_score
print(accuracy_score(y_test,y_pred)*100)
print(confusion_matrix(y_test,y_pred))
```

Output:

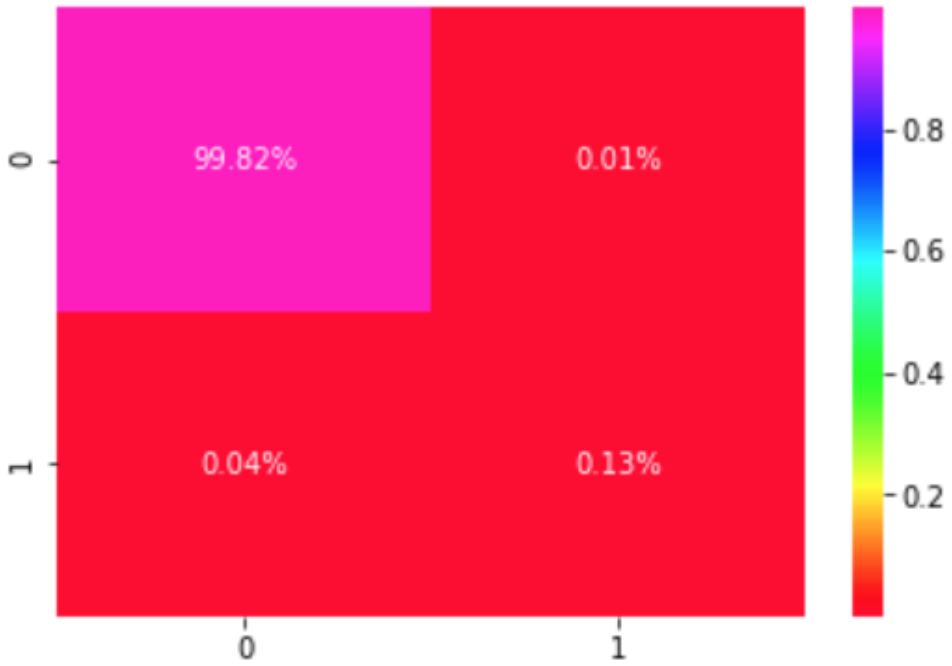
```
99.95365298727565
[[71075     7]
 [   26    94]]
```

Plot the Confusion Matrix:

For better visualization of confusion matrix use seaborn plotting package

```
cf_matrix=confusion_matrix(y_test,y_pred)
import seaborn as sns
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,fmt='.2%',
cmap='gist_rainbow')
```

Output:



Using Random Forest Classifier 94 is correctly found as Fraud Cases. Now let us calculate how many real fraud cases are there in y_test data for that convert y-test(array) to DataFrame as follows,

```
l_y_test=pd.DataFrame(y_test,columns=['Class'])
len(l_y_test[l_y_test["Class"]==1])
```

Output:

```
120
```

Therefore using Random Forest Classifier 94 is detected as fraud cases out of 120 cases 94/120*100=78.33 % accuracy which is quite good.