CREDIT CARD FRAUD DETECTION

ÖZGE GÜNEY
2100365

BAU
Bahçeşehir University

In recent years, because of increasing in online transactions, credit card frauds have increased. Therefore, banks had to deal with this problem using data mining techniques. In this data mining project, I am going to use python to create classification algorithms such as to detect credit card fraud by analyzing the old data.

Dataset which I used in this project is

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?select=creditcard.csv

About Dataset:

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days.
It contains only numerical input variables which are the result of a PCA transformation. 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
Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

The dataset have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

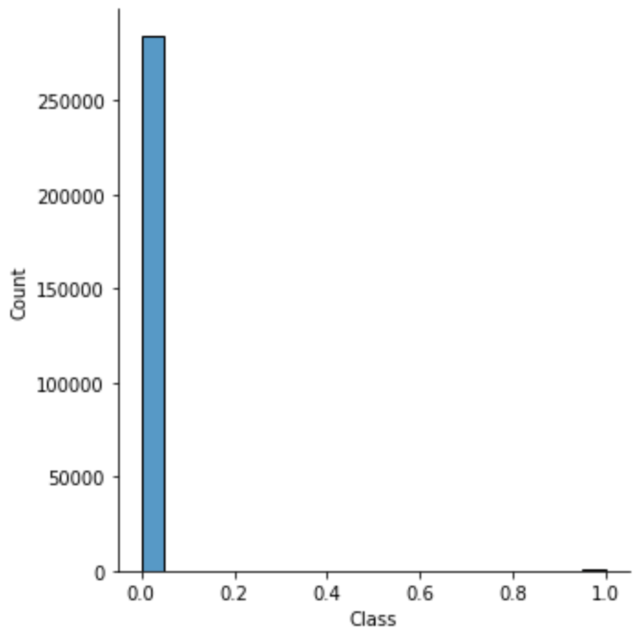


Figure 1: Numbers of Class 0 for normal transactions and Class1 for fraud transactions

If the data is unbalanced, accuracy is not important and minority class' precision and recall are important. If the data is balanced, I can use accuracy.

Dataset describe:

Here, our result is unbalanced, we can see it in Figure 1.

I used Area Under the Precision-Recall Curve (AUPRC) . As we see below in Figure 2, Random Forest is the best algorithm.

Unbalanced Dataset Describe

count	284807.000000
mean	0.001727
std	0.041527
min	0.000000
%25	0.000000
%50	0.000000
%75	0.000000
max	1.000000

Majority and minority class shape

(284315,31)	class2.shape
(492, 31)	class1.shape

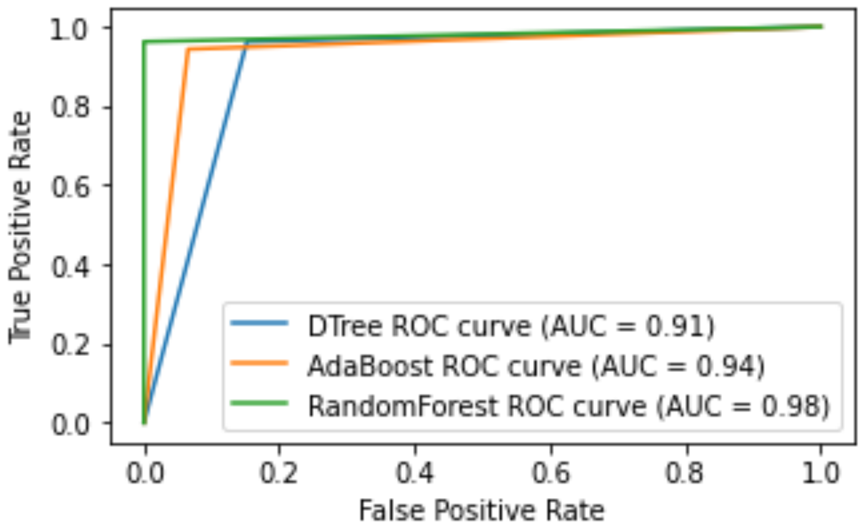


Figure 2: Accuracy of the different algorithms

I used in this project downsample: Now, they includes 492 rows.

Balanced Dataset describe

count	984.000000
mean	0.500000
std	0.500254
min	0.000000
%25	0.000000
%50	0.500000
%75	1.000000
max	1.000000

Majority and minority class

(492,31)	class2.shape
(492, 31)	class1.shape

Accuracy of Different Algorithms

Decision Tree accuracy	0.91
AdaBoost accuracy	0.94
RandomForest accuracy	0.98

Correlations:

We can see the correlation between features in Figure 3.

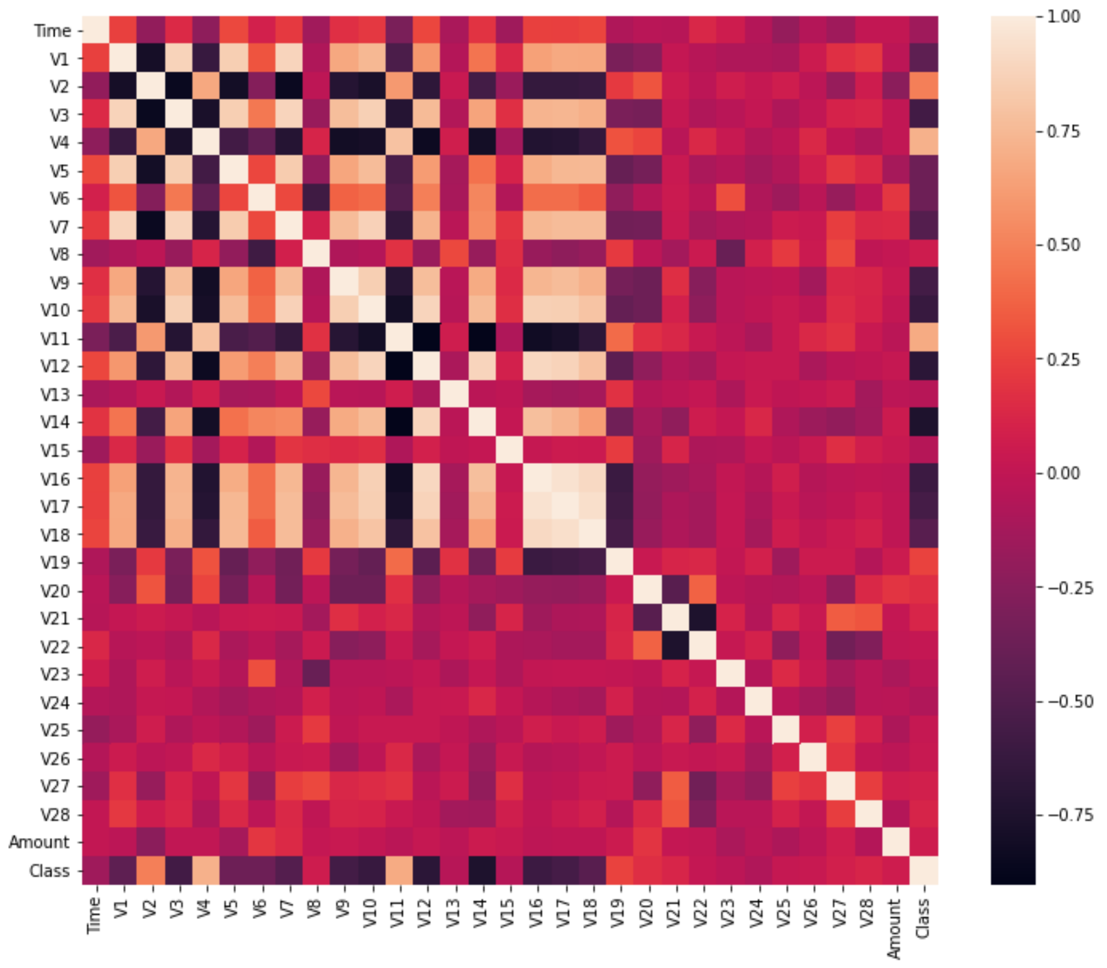


Figure 3:The Correlation of Balanced credit card

get top correlations with “Class”

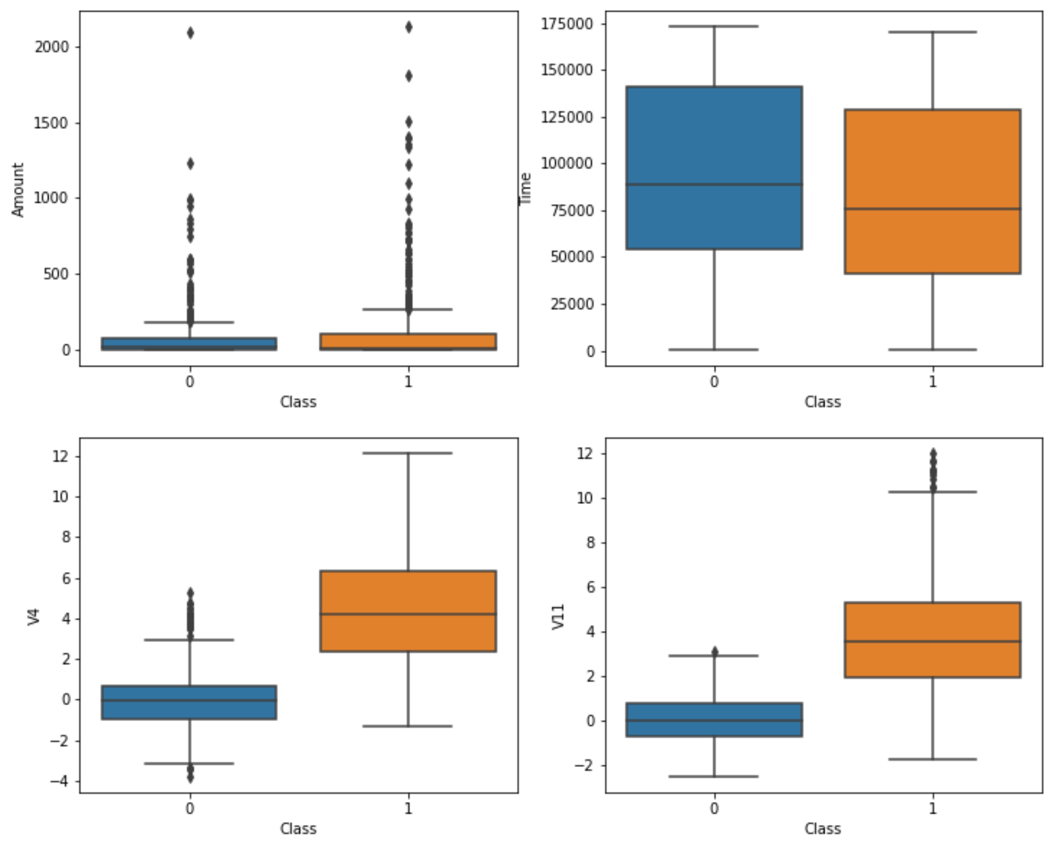
Class	1.000000
V4	0.712453
V11	0.681193
V2	0.491064
V19	0.253659
V20	0.165434
V21	0.120668
V28	0.110680
V27	0.082108
V8	0.061467

According to this result, these features (V4,V11,V2, V19, V20, V21, V28, V27, V8) effect highly Class output.

Anova F-value:
Below is the result of one-way anova test: As we see below, V4 and V11 features are very effective to the output.

AMOUNT : 8.477407838623522 0.0036771657911981483
V4: 1035.9695475616334 9.20042927677307e-156
V11: 859.8292895258559 2.8839456864094746e-136
Time : 11.578647456592767 0.0006940682737631966

As we see below, time and amount features are not effective to the output Class.



We can use SelectKBest to find features with the highest scores. By using SelectKBest I got this result.

features		values	p-values
0	V14	1259.408993	3.560270e-178
1	V4	1002.311300	3.580733e-152
2	V11	872.700886	9.402612e-138
3	V12	863.859359	9.848618e-137
4	V10	635.362783	1.622826e-108
5	V16	553.260524	2.177498e-97
6	V3	462.535291	2.250863e-84
7	V9	452.888546	6.084922e-83
8	V17	441.992780	2.589774e-81
9	V2	326.563603	3.064403e-63
10	V7	283.725974	4.045118e-56

As a final example, I want to show you a part of decision tree which is used here.

```
|--- feature_14 <= -2.19
|   |--- feature_14 <= -3.38
|   |   |--- feature_12 <= 0.19
|   |   |   |--- class: 1
|   |   |--- feature_12 >  0.19
|   |   |   |--- feature_16 <= 0.76
|   |   |   |   |--- class: 0
|   |   |   |--- feature_16 >  0.76
|   |   |   |   |--- class: 1
|   |--- feature_14 >  -3.38
|   |   |--- feature_17 <= 2.21
|   |   |   |--- feature_11 <= -0.06
|   |   |   |   |--- feature_29 <= 260.36
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_29 >  260.36
|   |   |   |   |   |--- class: 1
|   |   |   |--- feature_11 >  -0.06
|   |   |   |   |--- class: 1
|   |   |--- feature_17 >  2.21
|   |   |   |--- class: 0
```

As we see, again feature V14 is effective feature. Similarly, V12 were chosen by using SelectKBest and we see it here.

To sum up,

Fraud detection dataset is unbalanced dataset. With unbalanced dataset, we can see the result by using Precision-Recall Curve (AUPRC). In addition to this, we can use downsample method and now we can use accuracy to compare the algorithms. Both of them gave me that random forest is the best algorithm for this problem.

In data mining project, explanatory data analysis is very important. There are some methods. Here, I used SelectKBest algorithm. And I compared the other methods. Many of them gave similar result.

Code:

```
import pandas as pd
import seaborn as sns



credit_card = pd.read_csv("/Users/ozgeguney/.spyder-py3/DataMining/creditcard.csv")


print(credit_card.describe(include="all").loc[:,"Class"])

credit_card_class1 =  credit_card[credit_card.Class==1]
print(credit_card_class1.shape)

print(credit_card.Class.value_counts())

# sns.displot(credit_card.Class)
```

```
cc_majority = credit_card[credit_card.Class==0]
cc_minority = credit_card[credit_card.Class==1]



from sklearn.utils import resample
cc_majority_downsampled = resample(cc_majority,
                                   replace=False,
                                   n_samples=492)

cc_balanced = pd.concat([cc_minority, cc_majority_downsampled])
print(cc_balanced.Class.value_counts())
print(cc_balanced.describe(include="all").loc[:,"Class"])



X = cc_balanced.loc[:,'Time':'Amount']
y = cc_balanced.loc[:,'Class']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
X_train = X_train.fillna(X_train.mean())
X_test = X_test.fillna(X_test.mean())

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
clf1 = DecisionTreeClassifier()
clf2 = AdaBoostClassifier(n_estimators=200)
clf3 = RandomForestClassifier(n_estimators=100, bootstrap=True)

clf1.fit(X_train, y_train);
clf2.fit(X_train, y_train);
clf3.fit(X_train, y_train);

y_pred1 = clf1.predict(X_test)
y_pred2 = clf2.predict(X_test)
y_pred3 = clf3.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred1))
print(classification_report(y_test,y_pred2))
print(classification_report(y_test,y_pred3))

# from sklearn.metrics import recall_score
# recall_acc_DecisionTree = recall_score (y_test,y_pred1)
# print(recall_acc_DecisionTree)
# recall_acc_AdaBoost = recall_score (y_test,y_pred2)
# print(recall_acc_AdaBoost)
# recall_acc_RandomForest = recall_score (y_test,y_pred3)
# print(recall_acc_RandomForest)

from sklearn.metrics import roc_curve, auc
fpr1, tpr1, thresholds1 = roc_curve(y_test, y_pred1,drop_intermediate=False)
fpr2, tpr2, thresholds2 = roc_curve(y_test, y_pred2,drop_intermediate=False)
fpr3, tpr3, thresholds3 = roc_curve(y_test, y_pred3,drop_intermediate=False)


import matplotlib.pyplot as plt
```

```
auc1 = auc(fpr1, tpr1)
auc2 = auc(fpr2, tpr2)
auc3 = auc(fpr3, tpr3)

plt.plot(fpr1,tpr1,label='DTree ROC curve (AUC = %0.2f)' % auc1);
plt.plot(fpr2,tpr2,label='AdaBoost ROC curve (AUC = %0.2f)' % auc2);
plt.plot(fpr3,tpr3,label='RandomForest ROC curve (AUC = %0.2f)' % auc3);
plt.legend(loc="lower right")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')



plt.figure(figsize=(12,10));
sns.heatmap(cc_balanced.corr());



# get top correlations with Class
cors = cc_balanced.corr();
cors.loc[:, "Class"].sort_values(ascending = False ).head(10)


sns.displot(credit_card.Class);


sns.catplot(x='Class', y='Amount', data=cc_balanced, jitter=0.2);


sns.catplot(x='Class', y='Amount', kind="box", data=cc_balanced);

f, ax = plt.subplots(2,2)
f.set_size_inches(12,10)
sns.boxplot(y="Amount", x="Class", data= cc_balanced, ax = ax[0,0]);
sns.boxplot(y="Time", x="Class", data= cc_balanced, ax = ax[0,1]);
sns.boxplot(y="V4", x="Class", data= cc_balanced, ax = ax[1,0]);
sns.boxplot(y="V11", x="Class", data= cc_balanced, ax = ax[1,1]);


cc_balanced["Amount"].hist(by=cc_balanced["Class"])
cc_balanced["V4"].hist(by=cc_balanced["Class"])


import scipy.stats as stats
group1 = cc_balanced[cc_balanced["Class"]==1].Amount
group2 = cc_balanced[cc_balanced["Class"]==0].Amount
fvalue, pvalue = stats.f_oneway(group1, group2)
print(fvalue, pvalue)


group1 = cc_balanced[cc_balanced["Class"]==1].V4
group2 = cc_balanced[cc_balanced["Class"]==0].V4
fvalue, pvalue = stats.f_oneway(group1, group2)
print(fvalue, pvalue)

group1 = cc_balanced[cc_balanced["Class"]==1].Time
group2 = cc_balanced[cc_balanced["Class"]==0].Time
fvalue, pvalue = stats.f_oneway(group1, group2)
print(fvalue, pvalue)
```

```
from sklearn.feature_selection import  SelectKBest, f_classif
# from sklearn.feature_selection import mutual_info_classif
selector = SelectKBest(f_classif, k='all')
# selector = SelectKBest(score_func=mutual_info_classif, k='all')
selector.fit(X, y)

import numpy as np
sorted_idx = np.argsort(selector.scores_)[::-1]
sorted_vals = np.sort(selector.scores_)[::-1]

d = {"features":X.columns[sorted_idx], "values":sorted_vals, "p-
values":selector.pvalues_[sorted_idx]}
df = pd.DataFrame(d)
df

from sklearn import tree
text_representation = tree.export_text(clf1)
print(text_representation)



from sklearn.tree import plot_tree
feature_names = cc_balanced.columns


plot_tree(clf1,
        feature_names = feature_names,
        class_names = "Class",
        filled = True,
        rounded = True)

plt.savefig('tree_visualization.png')
```