NMAM Institute of Technology

Department of Artificial Intelligence and Machine Learning

Machine Learning Lab Programs (2022-23)

Faculty Coordinator: Mrs. Disha D N **Semester:5th**

Part-A

1. Demonstrate how you handle imbalanced data using Random under sampling, Random over sampling and SMOTE analysis on given dataset

Importing Dataset

```
import pandas as pd
df=pd.read_csv("creditcard.csv")
df.head()
```

```
   Time        V1        V2        V3        V4        V5        V6
V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388
0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -
0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499
0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203
0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921
0.592941

         V8        V9  ...       V21       V22       V23       V24
V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928
0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846
0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -
0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575
0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -
0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

Checking Null Values

```
df.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

Counting Class Values

```
df['Class'].value_counts()
```

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

Trying with Imbalaned Data Without Resampling

Splitting Dataset

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

Splitting into Train & Test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Building the Model

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

Fitting

```
dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

Prediction

```
pred=dtc.predict(x_test)
```

Finding Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

```
0.9992099996488887
```

Classification Report

```
from sklearn.metrics import classification_report
targets=['0','1']
print(classification_report(y_test,pred,target_names=targets))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56884
           1       0.69      0.78      0.73        78

    accuracy                           1.00     56962
   macro avg       0.84      0.89      0.87     56962
weighted avg       1.00      1.00      1.00     56962
```

Here, we observed much difference in F1-Score for the variables. Hence we should resample.

Now We Try Using Resampling

Undersampling ->

Splitting Dataset into Two Dataframes

```
Legit=df[df['Class']==0]
Fraud=df[df['Class']==1]
```

Checking Number of Rows & Columns

```
Legit.shape
```

```
(284315, 31)
```

```
Fraud.shape
```

```
(492, 31)
```

Sampling Equal to The Number of Rows in Minority Class

```
Legit_Sample=Legit.sample(n=492)
```

```
Legit_Sample.shape
```

```
(492, 31)
```

Concatenating to Form New Dataframe

```
new_df=pd.concat([Legit_Sample,Fraud],axis=0)
```

Counting Class Values

```
new_df['Class'].value_counts()
```

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

Splitting Dataset

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

Splitting into Train & Test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Building the Model

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

Fitting

```
dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

Prediction

```
pred=dtc.predict(x_test)
```

Finding Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

```
0.8984771573604061
```

Classification Report

```
from sklearn.metrics import classification_report
targets=['0','1']
print(classification_report(y_test,pred,target_names=targets))
```

```
              precision    recall  f1-score   support

           0       0.91      0.89      0.90        98
           1       0.89      0.91      0.90        99

    accuracy                           0.90       197
   macro avg       0.90      0.90      0.90       197
weighted avg       0.90      0.90      0.90       197
```

Here, We See That The F1-Score is Same(Almost).

Here, We See That The F1-Score is Same(Almost).

Oversampling ->

Splitting Dataset into Two Dataframes

```
Legit=df[df['Class']==0]
Fraud=df[df['Class']==1]
```

Checking Number of Rows & Columns

```
Legit.shape
```

```
(284315, 31)
```

```
Fraud.shape
```

```
(492, 31)
```

Sampling Equal to The Number of Rows in Minority Class

```
Fraud_Sample=Fraud.sample(n=284315,replace=True)
```

```
Fraud_Sample.shape
```

```
(284315, 31)
```

Concatenating to Form New Dataframe

```
new_df=pd.concat([Legit,Fraud_Sample],axis=0)
```

Counting Class Values

```
new_df['Class'].value_counts()
```

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

Splitting Dataset

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

Splitting into Train & Test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Building the Model

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

Fitting

```
dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

Prediction

```
pred=dtc.predict(x_test)
```

Finding Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

```
0.9997625872711605
```

Classification Report

```
from sklearn.metrics import classification_report
targets=['0','1']
print(classification_report(y_test,pred,target_names=targets))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56932
           1       1.00      1.00      1.00     56794

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

Here, We See That The F1-Score is Same(Almost).

SMOTE Oversampling ->

Installing Imblearn

```
# !pip install imblearn
```

Splitting Dataset

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

Oversampling using Imblearn SMOTE

```
from imblearn.over_sampling import SMOTE
oversample=SMOTE()
x,y=oversample.fit_resample(x,y)
```

Splitting into Train & Test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Building the Model

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

Fitting

```
dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

Prediction

```
pred=dtc.predict(x_test)
```

Finding Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

```
0.9985315583068076
```

Classification Report

```
from sklearn.metrics import classification_report
targets=['0','1']
print(classification_report(y_test,pred,target_names=targets))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56971
           1       1.00      1.00      1.00     56755

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

Here, We See That The F1-Score is Same(Almost).

2. Demonstrate feature scaling operation using standardization and Normalization techniques on any given dataset and predict the results with and without scaling operation

Standardization

```
from sklearn import datasets
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
import seaborn as sns
```

Load Dataset

```
iris=datasets.load_iris()
x=pd.DataFrame(iris['data'],columns=iris['feature_names'])
y=pd.DataFrame(iris.target)
x.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2
```