Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

Experiment 7

(AdaBoost)

Aim: Evaluate the performance of boosting algorithm (AdaBoost) with different base learners and hyperparameter tuning.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:
Dataset 1: Synthetic dataset
Dataset 2: CreditcardFraud.csv: The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data are not provided. 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, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

1. Implement Decision Tree classifier and Logistic Regression on Dataset 1 using K fold cross validation and compare the results with AdaBoost classifier with base learner as Decision tree and Logistic Regression.
2. Check if there is class imbalance problem in Dataset 2. Compare the results of decision tree classifier and AdaBoost classifier on Dataset 2 and write your analysis.
3. Implement AdaBoost with base learner as decision tree on dataset 2 using K fold cross validation. Perform Hyperparameter tuning using (a) different depth, (b) different learning rate and (c) grid search CV. Show your results using Boxplot.

```
#libraries
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
[ ]  #synthetic dataset
     x, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, random_state=6)
     X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
print(x)
```

```
[[-3.47224758  1.95378146  0.04875169 ...  2.07283886  0.08385173
   0.91461126]
 [-2.42264447  1.49687583 -0.80110683 ...  1.14726175 -2.86306705
  -0.27575018]
 [-4.01744369 -2.26537329  2.72577799 ...  1.34014025 -0.78634498
  -1.17749558]
 ...
 [ 2.39019744 -0.28042398 -0.01286339 ... -0.95516099 -0.76710459
   1.70412285]
 [ 0.60081099  1.84539674 -1.58163928 ... -1.55912569 -2.26992832
   0.42082267]
 [-1.27669747  1.6527396  -1.39187956 ...  0.72869505 -0.49441791
   2.75397458]]
```

```
[ ]  print(y)
```

```
[0 1 1 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1
 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1
 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0
 1 1 0 0 0 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 0
 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0
 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1
 1 1 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0
 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 0 0 1 0 1 1 1 0 1 0 1 1 0
 1 0 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1
 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0
 0 0 0 0 1 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 0 1 1 0 0
 0 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 0
 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0
 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 1 0 0
 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 1 0 0 0
 0 1 1 0 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0
 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0
 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1 1 0
 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 1
 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0
 0 0 1 0 1 1 1 1 0 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 0 1 1 0
 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0
 0 0 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1
```

Adaboost on decision tree

```
[ ]  from sklearn.datasets import make_classification
     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import AdaBoostClassifier
     from sklearn.metrics import accuracy_score


     dt = DecisionTreeClassifier(random_state=42)
     dt.fit(X_train, y_train)
     dt_pred = dt.predict(X_test)
     dt_acc = accuracy_score(y_test, dt_pred)
     print(f"Decision Tree Accuracy: {dt_acc}")


     ada = AdaBoostClassifier(base_estimator=dt, random_state=42)
     ada.fit(X_train, y_train)
     ada_pred = ada.predict(X_test)
     ada_acc = accuracy_score(y_test, ada_pred)
     print(f"AdaBoost Accuracy: {ada_acc}")
```

```
Decision Tree Accuracy: 0.79
AdaBoost Accuracy: 0.76
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_base.py:166:
  warnings.warn(
```

```
[ ]  from sklearn.datasets import make_regression

     from sklearn.tree import DecisionTreeRegressor
     from sklearn.ensemble import AdaBoostRegressor
     from sklearn.metrics import mean_squared_error


     dt = DecisionTreeRegressor(random_state=42)
     dt.fit(X_train, y_train)
     dt_pred = dt.predict(X_test)
     dt_mse = mean_squared_error(y_test, dt_pred)
     print(f"Decision Tree MSE: {dt_mse}")


     ada = AdaBoostRegressor(base_estimator=dt, random_state=42)
     ada.fit(X_train, y_train)
     ada_pred = ada.predict(X_test)
     ada_mse = mean_squared_error(y_test, ada_pred)
     print(f"AdaBoost MSE: {ada_mse}")
```

```
Decision Tree MSE: 0.21
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_base.py:166
  warnings.warn(
AdaBoost MSE: 0.05
```

DATASET 2

```
import numpy as np
import pandas as pd
from numpy import mean
from numpy import std
from numpy import log,dot,e,shape
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.ensemble import AdaBoostClassifierfrom sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
[ ] df = pd.read_csv('/content/Bank_Personal_Loan_Modelling.csv')
```

```
[ ] df
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

```
#selecting target
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values
```

```
print(X)
```

```
[[1.000e+00 2.500e+01 1.000e+00 ... 1.000e+00 0.000e+00 0.000e+00]
 [2.000e+00 4.500e+01 1.900e+01 ... 1.000e+00 0.000e+00 0.000e+00]
 [3.000e+00 3.900e+01 1.500e+01 ... 0.000e+00 0.000e+00 0.000e+00]
 ...
 [4.998e+03 6.300e+01 3.900e+01 ... 0.000e+00 0.000e+00 0.000e+00]
 [4.999e+03 6.500e+01 4.000e+01 ... 0.000e+00 0.000e+00 1.000e+00]
 [5.000e+03 2.800e+01 4.000e+00 ... 0.000e+00 0.000e+00 1.000e+00]]
```

```
print(Y)
```

```
[0 0 0 ... 0 0 1]
```

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

```
tree_clf = DecisionTreeClassifier(max_depth=2)
ada_clf = AdaBoostClassifier(base_estimator=tree_clf, n_estimators=200, learning_rate=0.5, random_state=42)
ada_clf.fit(X_train, y_train)
y_pred = ada_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was rena
  warnings.warn(
Accuracy: 0.65
```

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
base_estimator = LinearRegression()
adaboost = AdaBoostRegressor(base_estimator=base_estimator, n_estimators=100, learning_rate=0.1)
adaboost.fit(X_train, y_train)
y_pred = adaboost.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean squared error:", mse)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimato
  warnings.warn(
Mean squared error: 0.18700528290026397
```