Implement Boosting Algo

ADA BOOST

Program with Output:

```
#Data Pre-processing Step
# importing libraries
import numpy as np
import pandas as pd
```

```
[1] #Data Pre-processing Step
    # importing libraries
    import numpy as np
    import pandas as pd
```

```
#importing datasets
data = pd.read_csv("income (1).csv")
data
```

Nilesh Kawar Roll no : 59 Batch A

```
#importing datasets
data = pd.read_csv("income (1).csv")
data
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week	income_level
0	39	77516	13	2174	0	40	0
1	50	83311	13	0	0	13	0
2	38	215646	9	0	0	40	0
3	53	234721	7	0	0	40	0
4	28	338409	13	0	0	40	0
...	...	...	...	...	...	...	...
48837	39	215419	13	0	0	36	0
48838	64	321403	9	0	0	40	0
48839	38	374983	13	0	0	50	0
48840	44	83891	13	5455	0	40	0
48841	35	182148	13	0	0	60	1

48842 rows × 7 columns

```
#Separating predictors and response
x=data.iloc[:,0:6].values
y =data.iloc[:,6].values
```

```
#Separating predictors and response
x=data.iloc[:,0:6].values
y =data.iloc[:,6].values
```

```
#Splitting the dataset into training and test samples
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,random_state = 1)
```

```
#Splitting the dataset into training and test samples
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,random_state = 1)
```

```
#Initializing Adaboost classifier and fitting the training data
from sklearn.ensemble import AdaBoostClassifier
adaboost = AdaBoostClassifier(n_estimators = 100, base_estimator = None,
 learning_rate = 1)
```

```
#train Adaboost classifier
adaboost.fit(x_train, y_train)
```

```
#Initializing Adaboost classifier and fitting the training data
from sklearn.ensemble import AdaBoostClassifier
adaboost = AdaBoostClassifier(n_estimators = 100, base_estimator = None, learning_rate = 1)

#train Adaboost classifier
adaboost.fit(x_train, y_train)
```

```
AdaBoostClassifier(learning_rate=1, n_estimators=100)
```

```
#predict the response for test dataset
y_pred = adaboost.predict(x_test)
y_pred
```

```
[6] #predict the response for test dataset
    y_pred = adaboost.predict(x_test)
    y_pred

    array([0, 0, 0, ..., 1, 0, 0])
```

```
#printing accuracy score for adaBoost Algorithm
from sklearn.metrics import accuracy_score
print('Accuracy score for AdaBoost Algorithm: ',accuracy_score(y_test, y_pred),'\n')
```

```
[7] #printing accuracy score for adaBoost Algorithm
    from sklearn.metrics import accuracy_score
    print('Accuracy score for AdaBoost Algorithm: ',accuracy_score(y_test, y_pred),'\n')

    Accuracy score for AdaBoost Algorithm:  0.8392875422254069
```

```
# Load libraries
from sklearn.ensemble import AdaBoostClassifier

# Import Support Vector Classifier
from sklearn.svm import SVC
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
svc=SVC(probability=True, kernel='linear')

# Create adaboost classifer object
abc =AdaBoostClassifier(n_estimators=50, base_estimator=svc,learning_rate=1)
```

```
# Train Adaboost Classifer
model=adaboost.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)


# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
# Load libraries
from sklearn.ensemble import AdaBoostClassifier

# Import Support Vector Classifier
from sklearn.svm import SVC
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
svc=SVC(probability=True, kernel='linear')

# Create adaboost classifer object
abc =AdaBoostClassifier(n_estimators=50, base_estimator=svc,learning_rate=1)

# Train Adaboost Classifer
model=adaboost.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)


# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8392875422254069
```

```
from sklearn.linear_model import LogisticRegression

mylogregmodel = LogisticRegression()

#Create adaboost classifier object
adboost = AdaBoostClassifier(n_estimators = 50, base_estimator = mylogregmodel,
learni ng_rate = 1)

# Train Adaboost Classifer
model = adaboost.fit(x_train, y_train)

#Predict the response for test dataset
```

```
y_pred = model.predict(x_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

```
from sklearn.linear_model import LogisticRegression

mylogregmodel = LogisticRegression()

#Create adaboost classifier object
adboost = AdaBoostClassifier(n_estimators = 50, base_estimator = mylogregmodel, learning_rate = 1)

# Train Adaboost Classifer
model = adaboost.fit(x_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.8392875422254069
```

Implement Boosting Algo

GRADIENT BOOST

Program with Output:

```
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingClassifier
```

```
[1] import pandas as pd
    from sklearn.model_selection import KFold
    from sklearn.model_selection import cross_val_score
    from sklearn.ensemble import GradientBoostingClassifier
```

```
#Importing various models to compare
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
```

```
from numpy import loadtxt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
#Importing various models to compare
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier

from numpy import loadtxt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
#loading data from csv
#mydata=loadtxt('/content/pima-indians-diabetes.csv', delimiter=",")
mydata = pd.read_csv('/content/pima-indians-diabetes.csv', delimiter=",")
print(mydata)
# spliting data into independent and dependent features
x = mydata.iloc[:,0:8].values
y = mydata.iloc[:,8].values
```

Nilesh Kawar Roll no : 59 Batch A

```
#loading  data from csv
#mydata=loadtxt('/content/pima-indians-diabetes.csv', delimiter=",")
mydata = pd.read_csv('/content/pima-indians-diabetes.csv', delimiter=",")
print(mydata)
# spliting data into independent and dependent features
x = mydata.iloc[:,0:8].values
y = mydata.iloc[:,8].values
```

```
       6  148  72  35    0  33.6  0.627  50  1
0      1   85  66  29    0  26.6  0.351  31  0
1      8  183  64   0    0  23.3  0.672  32  1
2      1   89  66  23   94  28.1  0.167  21  0
3      0  137  40  35  168  43.1  2.288  33  1
4      5  116  74   0    0  25.6  0.201  30  0
..    ..  ...  ..  ..  ...   ...    ...  ..  ..
762   10  101  76  48  180  32.9  0.171  63  0
763    2  122  70  27    0  36.8  0.340  27  0
764    5  121  72  23  112  26.2  0.245  30  0
765    1  126  60   0    0  30.1  0.349  47  1
766    1   93  70  31    0  30.4  0.315  23  0

[767 rows x 9 columns]
```

```
# split data into train and test sets
seed = 1
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state =seed)
```

```
[4] # split data into train and test sets
    seed = 1
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=seed)
```

```
#Running various models
models = []
models.append(('LogisticRegression', LogisticRegression()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('SVM', SVC()))
models.append(('XGB',XGBClassifier(eta=0.01,gamma=10))) #eta = 0.01,gamma = 10

import time

# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'

for name, model in models:
 start_time = time.time()
```

```
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
predictions = [round(value) for value in y_pred]


# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0),name)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
names = []
scoring = 'accuracy'

for name, model in models:
    start_time = time.time()
    model.fit(x_train, y_train)

    y_pred = model.predict(x_test)
    predictions = [round(value) for value in y_pred]


    # evaluate predictions
    accuracy = accuracy_score(y_test, predictions)
    print("Accuracy: %.2f%%" % (accuracy * 100.0),name)
    print("--- %s seconds ---" % (time.time() - start_time))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
Accuracy: 75.97% LogisticRegression
--- 0.0669870471954345 7 seconds ---
Accuracy: 74.68% KNN
--- 0.0145294666290283 2 seconds ---
Accuracy: 78.57% SVM
--- 0.0337100028991899 2 seconds ---
Accuracy: 74.68% XGB
--- 0.1885750293731689 5 seconds ---
```

Implement Boosting Algo

VOTING ENSEMBLE

Program with Output:

```
# importing libraries
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris #Loading iris dataset
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Nilesh Kawar Roll no : 59 Batch A

```
# importing libraries
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris   #Loading iris dataset
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
#loading iris dataset
iris = load_iris()

X = iris.data[:, :4]
Y = iris.target

# train test split
X_train, X_test, y_train, y_test = train_test_split(X, Y,
 test_size = 0.20,  random_state = 42)
```

```
# group / ensemble of models
estimator = []
estimator.append(('LR', LogisticRegression( solver ='lbfgs', multi_class
='multinomial ', max_iter = 200)))

estimator.append(('SVC', SVC(gamma ='auto', probability = True)))

estimator.append(('DTC', DecisionTreeClassifier()))
```

Nilesh Kawar Roll no : 59 Batch A

```
# Voting Classifier with hard voting
vot_hard = VotingClassifier(estimators = estimator, voting
='hard') vot_hard.fit(X_train, y_train)
y_pred = vot_hard.predict(X_test)

# using accuracy_score metric to predict accuracy score for Hard
Voting score = accuracy_score(y_test, y_pred)
print("Hard Voting Score % d" % score)
```

```
# Voting Classifier with soft voting
vot_soft = VotingClassifier(estimators = estimator, voting
='soft') vot_soft.fit(X_train, y_train)
y_pred = vot_soft.predict(X_test)

# using accuracy_score to predict accuracy score for Soft Voting
score = accuracy_score(y_test, y_pred)
print("Soft Voting Score % d" % score)
```

⚠