A python program to implement ADA Boosting

Aim:

To implement a python program for Ada Boosting.

Algorithm:

Step 1: Import necessary Libraries

import numpy as np

import pandas as pd

import Decision Tree classifier from sklearn.tree

Step 2: Load and prepare Data

Load your dataset using pd.read_csv() (eg. df = pd.read_csv('data.csv')).

Separate features (x) and target (y).

Step 3: Initialize parameters:

Set the number of weak classifiers. n_estimators. Initialize an array weights ~~from~~ for instance weights.

Step 4: Train weak classifiers.

loop for n estimators iterations. Predict the target values using the trained week classifier.

step 6: make predictions

step 8: Evaluate the model
Compute the accuracy of the adaBoost model on the testing set using accuracy_score() .

Step 7: output Results
Print or plot the final accuracy and possibly other evaluation metrics.

Program:

```
import numpy as np
import pandas as pd
from sklearn.tree import decision Tree classifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

x = df[['x1','x2']].values

y = df['label'].values

n_estimators =5
n_Samples = X_train.Shape[0].
Weights = np.ones(n_samples)/n_samples
classifiers =[]
 alphas =[]
```

5

```
if err == 0;
     err = 1e-10
alpha = 0.5 * np.log (1 - err /err)
weights = weights * np.exp(-alpha * y_train *
                                          y_pred)

weights = /= np.sum(weights)
classifiers.append(clf)
alphas.append (alpha)
def predict (x):
final_score = np.zeros(x.shape[0])
for clf, alpha in zip(classifiers alphas):
    pred = clf.predict (x)
     final score += alpha * preds
return np.sign (final score)
y_pred_test = predict(x_test)
Print(" final accuracy on test set: ", acc)
Print (" classifier weights (alphas): ", alphas)
```

Result:

Thus the python program to implement adaboosting has been executed successfully and the results have been verified and analyzed.

Ex no:6

A python program to implement Gradient boosting

Aim:

To implement a python program using the gradient boosting model.

Algorithm:

Step 1: Import necessary libraries

import numpy as np

import pandas as pd

import decision tree Regressor from sklearn. tree

Step 2: Prepare the data

Load your dataset into a dataframe using Pd. read_csv ("your_dataset.csv"). Split the dataset into features (x) and target (y).

Step 3: Initialize parameters.

Set the number of boosting rounds set the weak learners.

Step 4: Initialize the base model.

Compoute the initial prediction as the mean of the target values. Initialize the Predictions to the base model's prediction.

Step 5: Iterate over boosting Rounds.

For each boosting round. Fit a decision tree to the pseudo_residuals: Append the fitted tree and learning rate to their respective lists.

Step 6: make predictions on test data

Initialize the test predictions with the base model's predictions.

Step 7: Evaluatel the model

Compute the mean squared error on the training data. Compute the mean_squared error on the test data.

Program :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
np.random.seed(42)
df=pd.Data frame()
df['x']=X.reshape(100)
df['res1']=df['y']-df['pred1']
plt.Satter(df['x'],df['y'])
plt.plot(df['x'],df['pred1'],color='red')
```

```
from sklearn.tree import DecisionTreeRegressor
from 1. fit(df['x'].values.reshape(100,1),
                    df['rest'].values)

plot_tree(tree 1)
plot.show()
np.random_seed(42)
x=np.random.rand(100,1)-0.5
y=3* x[:,0]** 2 +0.05* np.random.randon(100)
gradient_boost(x,y,5,lr=1)
```

Result:

Thus the python program to implement gradient boosting for the standard uniform distribution has been succesfully implemented and the results have been verified and analized successfully.