Experiment No. 6
Apply Boosting Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

Aim: Apply Boosting algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: Apply Boosting algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

Theory:

Suppose that as a patient, you have certain symptoms. Instead of consulting one doctor, you choose to consult several. Suppose you assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnosis they have made. The final diagnosis is then a combination of the weighted diagnosis. This is the essence behind boosting.

Algorithm: Adaboost- A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D , a set of d class labelled training tuples
- k, the number of rounds (one classifier is generated per round)
- a classification learning scheme

Output: A composite model

Method

1. Initialize the weight of each tuple in D is 1/d
2. For i=1 to k do // for each round
3. Sample D with replacement according to the tuple weights to obtain D_i
4. Use training set D_i to derive a model M_i
5. Computer error(M_i), the error rate of M_i
6. Error(M_i)=$\sum w_j * err(X_j)$
7. If Error(M_i)>0.5 then
8. Go back to step 3 and try again
9. endif
10. for each tuple in D_i that was correctly classified do
11. Multiply the weight of the tuple by error(Mi)/(1-error(M_i))
12. Normalize the weight of each tuple
13. end for

To use the ensemble to classify tuple X

1. Initialize the weight of each class to 0
2. for i=1 to k do // for each classifier
3. $w_i = \log((1-error(M_i))/error(M_i))$//weight of the classifiers vote
4. $C=M_i(X)$ // get class prediction for X from M_i
5. Add w_i to weight for class C
6. end for
7. Return the class with the largest weight.

Dataset:

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.
Attribute Information:
Listing of attributes:

>50K, <=50K.

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad &Tobago, Peru, Hong, Holand-Netherlands.

Code:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier  #
Import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Load the Adult Census Income Dataset
data = pd.read_csv('adult.csv')  # Make sure to provide the correct file path

# Split the data into features (X) and the target variable (y)
X = data.drop('income', axis=1)
y = data['income']

# Encode the categorical columns using LabelEncoder
categorical_columns = X.select_dtypes(include=['object']).columns
label_encoder = LabelEncoder()
X[categorical_columns] =
X[categorical_columns].apply(label_encoder.fit_transform)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize the AdaBoost classifier with a base estimator (Random Forest)
base_estimator = RandomForestClassifier( n_estimators=10,random_state=42)  #
You can adjust the number of estimators
```

```
adaboost_classifier = AdaBoostClassifier(base_estimator=base_estimator,
n_estimators=10, random_state=42)

# Fit the AdaBoost classifier to the training data
adaboost_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = adaboost_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='>50K')
recall = recall_score(y_test, y_pred, pos_label='>50K')
f1 = f1_score(y_test, y_pred, pos_label='>50K')
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
```

Output:

Accuracy: 0.85

Precision: 0.72

Recall: 0.60

F1 Score: 0.65

Confusion Matrix:

[[4613 363]

[613 924]]

Conclusion:

- **Accuracy**: An accuracy of 0.85 indicates that the AdaBoost model with a Random Forest base estimator correctly classifies 85% of the test instances. It's a decent overall performance metric.
- **Precision**: The precision of 0.72 means that when the model predicts '>50K', it is correct 72% of the time. This suggests that the model has a relatively good ability to avoid false positives.
- **Recall**: A recall of 0.60 indicates that the model correctly identifies 60% of the '>50K' instances. This metric is lower, indicating that the model may miss some instances of '>50K'.
- **F1 Score**: The F1 score of 0.65 represents a balance between precision and recall. It shows that the model has a reasonable trade-off between these two metrics.
- **Confusion Matrix**: The confusion matrix provides a detailed breakdown of the model's performance, showing the true positives, true negatives, false positives, and false negatives.

Both Boosting (AdaBoost) and Random Forest are ensemble learning techniques, but they work differently.

Boosting focuses on improving the classification of difficult instances by giving more weight to misclassified data points. It iteratively creates multiple weak learners and combines their predictions. Random Forest, on the other hand, builds multiple decision trees and combines their predictions through a majority vote (or averaging in regression tasks). It reduces overfitting and generally has good performance.In your results, the AdaBoost model achieved a decent accuracy but had a lower recall compared to precision, indicating that it may miss some '>50K' instances.

Random Forest models are known for their robustness and typically provide good generalization. The choice between AdaBoost and Random Forest may depend on the specific dataset and goals.