| Experiment No. 5 |
| --- |
| Apply Boosting Algorithm on Adult Census Income Dataset  and analyze the performance of the model |
| Date of Performance: 20/8/2024 |
| Date of Submission: 3/9/2024 |

**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

1. Use training set D

i

to derive a model M

i

1. Computer error(M ), the error rate of M

i i

1. Error(M )=∑w \*err(X )

i j j

1. If Error(M )>0.5 then

i

1. Go back to step 3 and try again
2. endif
3. for each tuple in D

i

that was correctly classified do

1. Multiply the weight of the tuple by error(Mi)/(1-error(M )

i

1. Normalize the weight of each tuple
2. 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 =log((1-error(M ))/error(M ))//weight of the classifiers vote

i i i

1. C=M (X) // get class prediction for X from M

i i

1. Add w

i

1. end for

to weight for class C

1. 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:



|  | age | workclass | fnlwgt | education | education.num | marital.status | \ |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 90 | ? | 77053 | HS-grad | 9 | Widowed |  |
| 1 | 82 | Private | 132870 | HS-grad | 9 | Widowed |  |
| 2 | 66 | ? | 186061 | Some-college | 10 | Widowed |  |
| 3 | 54 | Private | 140359 | 7th-8th | 4 | Divorced |  |
| 4 | 41 | Private | 264663 | Some-college | 10 | Separated |  |
| ... | ... | ... | ... | ... | ... | ... |  |
| 32556 | 22 | Private | 310152 | Some-college | 10 | Never-married |  |
| 32557 | 27 | Private | 257302 | Assoc-acdm | 12 | Married-civ-spouse |  |
| 32558 | 40 | Private | 154374 | HS-grad | 9 | Married-civ-spouse |  |
| 32559 | 58 | Private | 151910 | HS-grad | 9 | Widowed |  |
| 32560 | 22 | Private | 201490 | HS-grad | 9 | Never-married |  |

|  | occupation | relationship | race | sex | capital.gain | \ |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | ? | Not-in-family | White | Female | 0 |  |
| 1 | Exec-managerial | Not-in-family | White | Female | 0 |  |
| 2 | ? | Unmarried | Black | Female | 0 |  |
| 3 | Machine-op-inspct | Unmarried | White | Female | 0 |  |
| 4 | Prof-specialty | Own-child | White | Female | 0 |  |
| ... | ... | ... | ... | ... | ... |  |
| 32556 | Protective-serv | Not-in-family | White | Male | 0 |  |
| 32557 | Tech-support | Wife | White | Female | 0 |  |
| 32558 | Machine-op-inspct | Husband | White | Male | 0 |  |
| 32559 | Adm-clerical | Unmarried | White | Female | 0 |  |
| 32560 | Adm-clerical | Own-child | White | Male | 0 |  |

capital.loss hours.per.week native.country income

| 0 | 4356 | 40 | United-States | <=50K |
| --- | --- | --- | --- | --- |
| 1 | 4356 | 18 | United-States | <=50K |
| 2 | 4356 | 40 | United-States | <=50K |
| 3 | 3900 | 40 | United-States | <=50K |
| 4 | 3900 | 40 | United-States | <=50K |
| ... | ... | ... | ... | ... |
| 32556 | 0 | 40 | United-States | <=50K |
| 32557 | 0 | 38 | United-States | <=50K |
| 32558 | 0 | 40 | United-States | >50K |
| 32559 | 0 | 40 | United-States | <=50K |
| 32560 | 0 | 20 | United-States | <=50K |

[32561 rows x 15 columns]



**age fnlwgt education.num capital.gain capital.loss hours.p**

**count** 32561.000000 3.256100e+04 32561.000000 32561.000000 32561.000000 32561

**mean** 38.581647 1.897784e+05 10.080679 1077.648844 87.303830 40

**std** 13.640433 1.055500e+05 2.572720 7385.292085 402.960219 12

**min** 17.000000 1.228500e+04 1.000000 0.000000 0.000000 1

**25%** 28.000000 1.178270e+05 9.000000 0.000000 0.000000 40

**50%** 37.000000 1.783560e+05 10.000000 0.000000 0.000000 40

**75%** 48.000000 2.370510e+05 12.000000 0.000000 0.000000 45



<class 'pandas.core.frame.DataFrame'> RangeIndex: 32561 entries, 0 to 32560 Data columns (total 15 columns):

# Column Non-Null Count Dtype

* 1. age 32561 non-null int64
  2. workclass 32561 non-null object
  3. fnlwgt 32561 non-null int64
  4. education 32561 non-null object
  5. education.num 32561 non-null int64
  6. marital.status 32561 non-null object
  7. occupation 32561 non-null object
  8. relationship 32561 non-null object
  9. race 32561 non-null object
  10. sex 32561 non-null object
  11. capital.gain 32561 non-null int64
  12. capital.loss 32561 non-null int64
  13. hours.per.week 32561 non-null int64
  14. native.country 32561 non-null object
  15. income 32561 non-null object dtypes: int64(6), object(9)

memory usage: 3.7+ MB None



age 0

workclass 0

fnlwgt 0

education 0

education.num 0

marital.status 0

occupation 0

relationship 0

race 0

sex 0

capital.gain 0

capital.loss 0

hours.per.week 0

native.country 0

income 0

dtype: int64

| # Replace '?' with NaN in the dataset data.replace('?', pd.NA, inplace=True) |
| --- |
| # Drop rows with missing values data.dropna(inplace=True) |
| # Encode categorical variables label\_encoder = LabelEncoder()  categorical\_columns = data.select\_dtypes(include=['object']).columns for column in categorical\_columns:  data[column] = label\_encoder.fit\_transform(data[column]) |
| # Split the data into training and testing sets X = data.drop("income", axis=1)  y = data["income"]  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) |
| from sklearn.ensemble import AdaBoostClassifier  from sklearn.metrics import accuracy\_score, classification\_report |
| # Create the AdaBoost classifier  ada\_boost\_classifier = AdaBoostClassifier(n\_estimators=50, random\_state=42) |
| # Fit the classifier to the training data ada\_boost\_classifier.fit(X\_train, y\_train) |





The Accuracy for boosting algo is : 0.8538040775733466

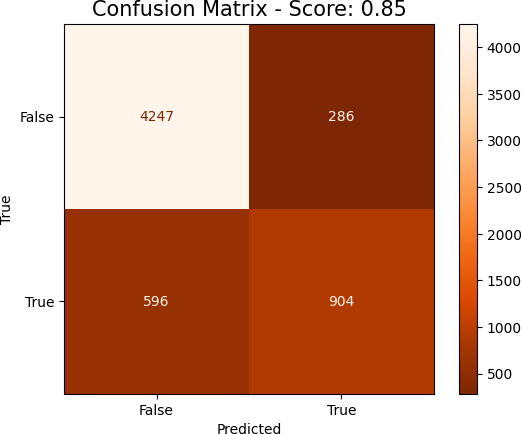


Confusion Matrix:

[[4247 286]

[ 596 904]]

| import matplotlib.pyplot as plt from sklearn import metrics  from sklearn.metrics import accuracy\_score, confusion\_matrix, ConfusionMatrixDisplay |
| --- |
| # Assuming you already have the y\_test and y\_pred values from your AdaBoost classifier confusion\_matrix = confusion\_matrix(y\_test, y\_pred) |
| # Calculate accuracy  accuracy = accuracy\_score(y\_test, y\_pred) |
| # Create a title for the plot with accuracy score  title = f'Confusion Matrix - Score: {round(accuracy, 2)}' |
| # Create the ConfusionMatrixDisplay  cm\_display = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix, display\_labels=[False, True]) |
| # Plot the confusion matrix with the specified title plt.figure(figsize=(8, 6)) cm\_display.plot(cmap='Oranges\_r', values\_format='d')  plt.title(title, size=15) plt.xlabel('Predicted') plt.ylabel('True')  plt.show() |



Accuracy Score: 0.8538040775733466



Classification Report:

precision recall f1-score support

| 0 | 0.88 | 0.94 | 0.91 | 4533 |
| --- | --- | --- | --- | --- |
| 1 | 0.76 | 0.60 | 0.67 | 1500 |

| accuracy |  |  | 0.85 | 6033 |
| --- | --- | --- | --- | --- |
| macro avg | 0.82 | 0.77 | 0.79 | 6033 |
| weighted avg | 0.85 | 0.85 | 0.85 | 6033 |

**Conclusion:**

Accuracy: 0.85, indicating that the model correctly predicts the income level.

Precision: For class 0, the precision is 88%, indicating that when the model predicts class 0, it is correct 88% of the time. For class 1, the precision is 76%, suggesting that the model's ability to correctly predict class 1 is not as high as for class 0.

Recall: For class 0, the recall is 94%, indicating that the model effectively captures 94% of all instances of class 0. For class 1, the recall is 60%, suggesting that the model's ability to identify all instances of class 1 is not as high as for class 0.

F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balance between these two metrics and is particularly useful when dealing with imbalanced datasets. For class 0, the F1 score is 0.91, indicating a good balance between precision and recall. For class 1, the F1 score is 0.67, which is lower and suggests room for improvement.

AdaBoost is a boosting algorithm that iteratively corrects errors by assigning greater weight to misclassified instances, making it more prone to overfitting if the base learner is complex. In contrast, Random Forest employs bagging to build multiple independent decision trees and combines their predictions by averaging or majority voting, reducing variance and overfitting. AdaBoost tends to balance both bias and variance, while Random Forest primarily reduces variance.