Experiment No. 4
Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:16-08-2023
Date of Submission:27-09-2023

Aim: Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: Able to perform various feature engineering tasks, apply Random Forest Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

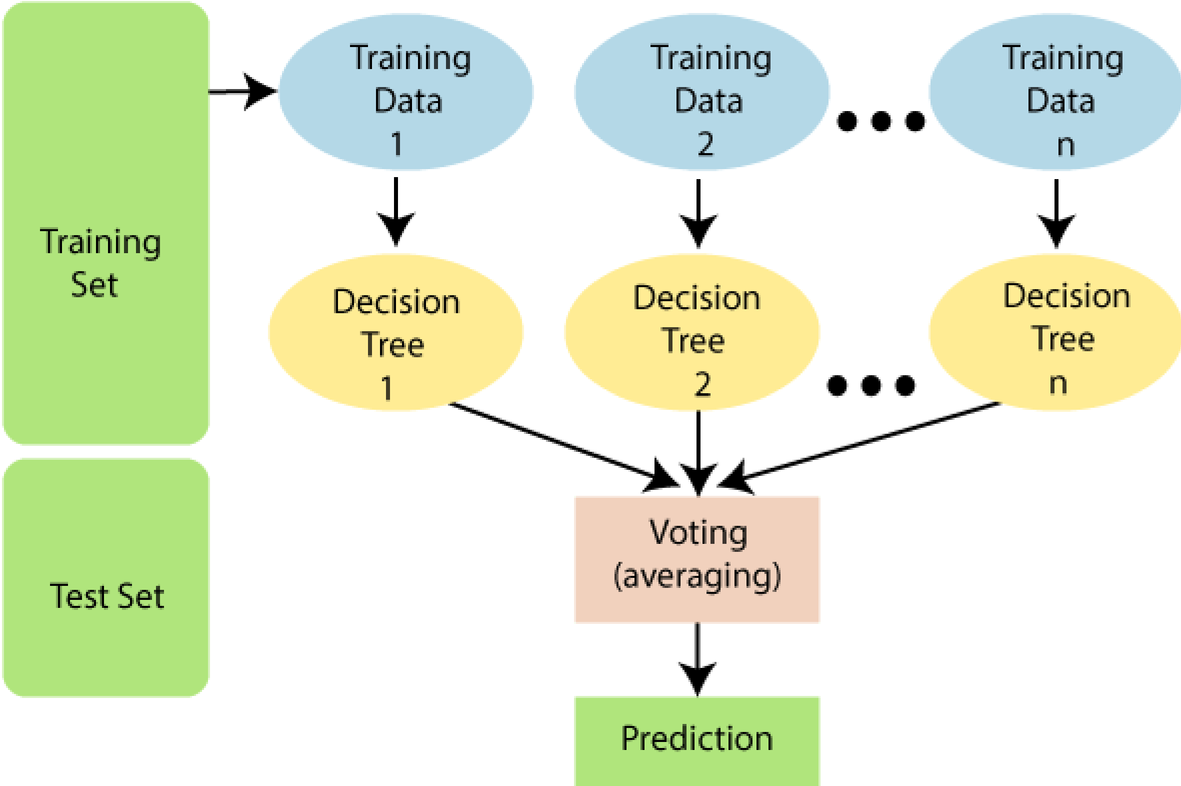
Theory:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



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.

exp-4-ml

October 9, 2023

[ ]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(style='white', context='notebook', palette='deep')
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score,␣
  ↪StratifiedKFold, learning_curve, train_test_split, KFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
# To ignore warning messages
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
df = pd.read_csv('./adult.csv')
```

[ ]:
```
df.head(5)
```

[ ]:
```
   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

          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

   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
```

[ ]:
```
print ("Rows : " ,df.shape[0])
print ("Columns : " ,df.shape[1])
print ("\nFeatures : \n" ,df.columns.tolist())
print ("\nMissing values : ", df.isnull().sum().values.sum())
print ("\nUnique values : \n",df.nunique())
```

```
Rows :  32561
Columns :  15

Features :
 ['age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.status',
'occupation', 'relationship', 'race', 'sex', 'capital.gain', 'capital.loss',
'hours.per.week', 'native.country', 'income']

Missing values :  0

Unique values :
 age                 73
workclass            9
fnlwgt           21648
education           16
education.num       16
marital.status       7
occupation          15
relationship         6
race                 5
sex                  2
capital.gain       119
capital.loss        92
hours.per.week      94
native.country      42
income               2
dtype: int64
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
```

```
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[ ]:
```
df.describe()
```

[ ]:
```
                age        fnlwgt  education.num  capital.gain  capital.loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085    402.960219
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours.per.week
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

[ ]:
```
# checking "?" total values present in particular 'workclass' feature
df_check_missing_workclass = (df['workclass']=='?').sum()
df_check_missing_workclass
```

[ ]: 1836

[ ]:
```
# checking "?" total values present in particular 'occupation' feature
df_check_missing_occupation = (df['occupation']=='?').sum()
```

```
df_check_missing_occupation
```

[ ]: 1843

[ ]:
```
# checking "?" values, how many are there in the whole dataset
df_missing = (df=='?').sum()
df_missing
```

[ ]:
```
age                  0
workclass         1836
fnlwgt               0
education            0
education.num        0
marital.status       0
occupation        1843
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
native.country     583
income               0
dtype: int64
```

[ ]:
```
percent_missing = (df=='?').sum() * 100/len(df)
percent_missing
```

[ ]:
```
age               0.000000
workclass         5.638647
fnlwgt            0.000000
education         0.000000
education.num     0.000000
marital.status    0.000000
occupation        5.660146
relationship      0.000000
race              0.000000
sex               0.000000
capital.gain      0.000000
capital.loss      0.000000
hours.per.week    0.000000
native.country    1.790486
income            0.000000
dtype: float64
```

[ ]:
```
# find total number of rows which doesn't contain any missing value as '?'
df.apply(lambda x: x !='?',axis=1).sum()
```

[ ]:
```
age               32561
workclass         30725
fnlwgt            32561
education         32561
education.num     32561
marital.status    32561
occupation        30718
relationship      32561
race              32561
sex               32561
capital.gain      32561
capital.loss      32561
hours.per.week    32561
native.country    31978
income            32561
dtype: int64
```

[ ]:
```python
# dropping the rows having missing values in workclass
df = df[df['workclass'] !='?']
df.head(5)
```

[ ]:
```
   age workclass  fnlwgt     education  education.num marital.status  \
1   82   Private  132870       HS-grad              9        Widowed
3   54   Private  140359       7th-8th              4       Divorced
4   41   Private  264663  Some-college             10      Separated
5   34   Private  216864       HS-grad              9       Divorced
6   38   Private  150601          10th              6      Separated

          occupation   relationship   race     sex  capital.gain  \
1    Exec-managerial  Not-in-family  White  Female             0
3  Machine-op-inspct      Unmarried  White  Female             0
4     Prof-specialty      Own-child  White  Female             0
5      Other-service      Unmarried  White  Female             0
6       Adm-clerical      Unmarried  White    Male             0

   capital.loss  hours.per.week native.country income
1          4356              18  United-States  <=50K
3          3900              40  United-States  <=50K
4          3900              40  United-States  <=50K
5          3770              45  United-States  <=50K
6          3770              40  United-States  <=50K
```

[ ]:
```python
# select all categorical variables
df_categorical = df.select_dtypes(include=['object'])
# checking whether any other column contains '?' value
df_categorical.apply(lambda x: x=='?',axis=1).sum()
```

[ ]:
```
workclass            0
education            0
marital.status       0
occupation           7
relationship         0
race                 0
sex                  0
native.country     556
income               0
dtype: int64
```

[ ]:
```
# dropping the "?"s from occupation and native.country
df = df[df['occupation'] !='?']
df = df[df['native.country'] !='?']
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   workclass       30162 non-null  object
 2   fnlwgt          30162 non-null  int64
 3   education       30162 non-null  object
 4   education.num   30162 non-null  int64
 5   marital.status  30162 non-null  object
 6   occupation      30162 non-null  object
 7   relationship    30162 non-null  object
 8   race            30162 non-null  object
 9   sex             30162 non-null  object
 10  capital.gain    30162 non-null  int64
 11  capital.loss    30162 non-null  int64
 12  hours.per.week  30162 non-null  int64
 13  native.country  30162 non-null  object
 14  income          30162 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[ ]:
```
from sklearn import preprocessing
# encode categorical variables using label Encoder
# select all categorical variables
df_categorical = df.select_dtypes(include=['object'])
df_categorical.head()
```

[ ]:
```
  workclass     education marital.status        occupation   relationship  \
1   Private       HS-grad        Widowed   Exec-managerial  Not-in-family
```

```
3  Private       7th-8th      Divorced  Machine-op-inspct      Unmarried
4  Private  Some-college     Separated     Prof-specialty      Own-child
5  Private       HS-grad      Divorced      Other-service      Unmarried
6  Private          10th     Separated       Adm-clerical      Unmarried

    race     sex native.country income
1  White  Female  United-States  <=50K
3  White  Female  United-States  <=50K
4  White  Female  United-States  <=50K
5  White  Female  United-States  <=50K
6  White    Male  United-States  <=50K
```

[ ]:
```
# apply label encoder to df_categorical
le = preprocessing.LabelEncoder()
df_categorical = df_categorical.apply(le.fit_transform)
df_categorical.head()
```

[ ]:
```
   workclass  education  marital.status  occupation  relationship  race  sex  \
1          2         11               6           3             1     4    0
3          2          5               0           6             4     4    0
4          2         15               5           9             3     4    0
5          2         11               0           7             4     4    0
6          2          0               5           0             4     4    1

   native.country  income
1              38       0
3              38       0
4              38       0
5              38       0
6              38       0
```

[ ]:
```
# Next, Concatenate df_categorical dataframe with original df (dataframe)
# first, Drop earlier duplicate columns which had categorical values
df = df.drop(df_categorical.columns,axis=1)
df = pd.concat([df,df_categorical],axis=1)
df.head(5)
```

[ ]:
```
   age  fnlwgt  education.num  capital.gain  capital.loss  hours.per.week  \
1   82  132870              9             0          4356              18
3   54  140359              4             0          3900              40
4   41  264663             10             0          3900              40
5   34  216864              9             0          3770              45
6   38  150601              6             0          3770              40

   workclass  education  marital.status  occupation  relationship  race  sex  \
1          2         11               6           3             1     4    0
3          2          5               0           6             4     4    0
```

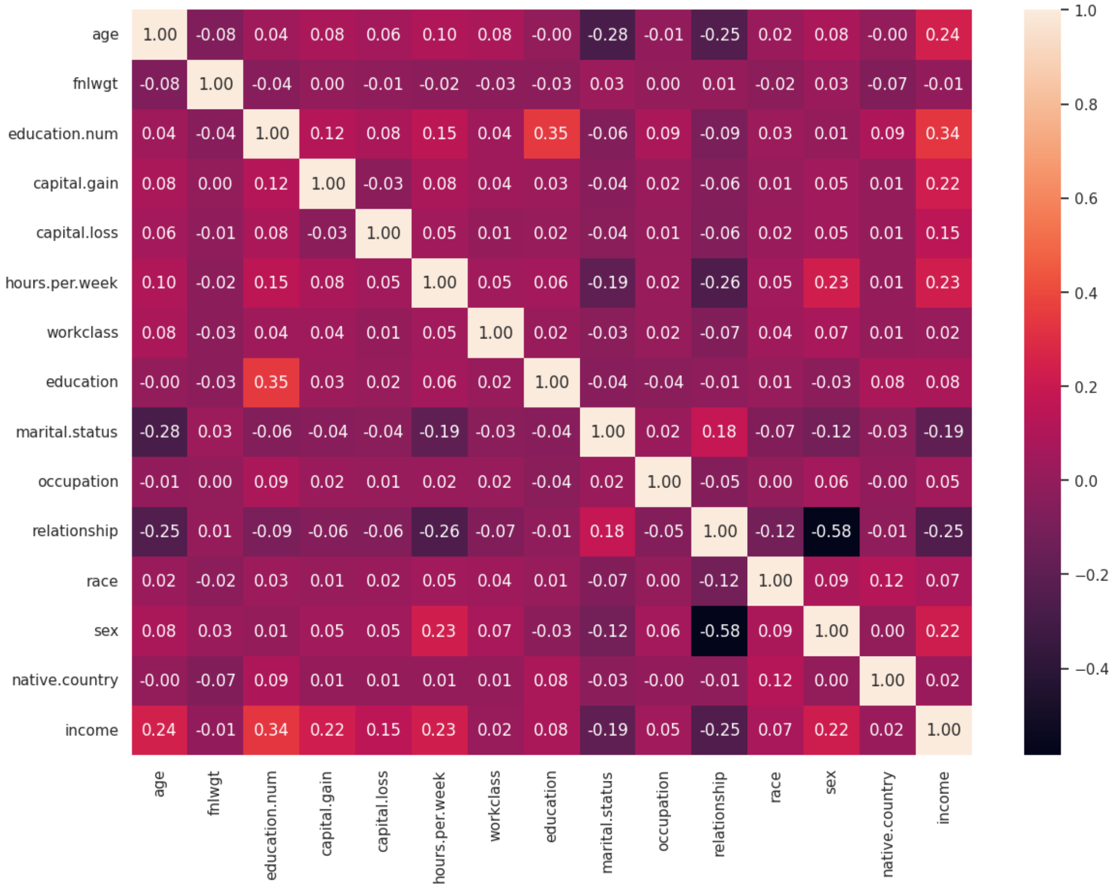
```
4           2          15                 5            9             3      4    0
5           2          11                 0            7             4      4    0
6           2           0                 5            0             4      4    1

   native.country  income
1              38       0
3              38       0
4              38       0
5              38       0
6              38       0
```

[ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   fnlwgt          30162 non-null  int64
 2   education.num   30162 non-null  int64
 3   capital.gain    30162 non-null  int64
 4   capital.loss    30162 non-null  int64
 5   hours.per.week  30162 non-null  int64
 6   workclass       30162 non-null  int64
 7   education       30162 non-null  int64
 8   marital.status  30162 non-null  int64
 9   occupation      30162 non-null  int64
 10  relationship    30162 non-null  int64
 11  race            30162 non-null  int64
 12  sex             30162 non-null  int64
 13  native.country  30162 non-null  int64
 14  income          30162 non-null  int64
dtypes: int64(15)
memory usage: 3.7 MB
```

[ ]:
```python
plt.figure(figsize=(14,10))
sns.heatmap(df.corr(),annot=True,fmt='.2f')
plt.show()
```



[ ]:
```python
# convert target variable income to categorical
df['income'] = df['income'].astype('category')
```

[ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   fnlwgt          30162 non-null  int64
 2   education.num   30162 non-null  int64
 3   capital.gain    30162 non-null  int64
 4   capital.loss    30162 non-null  int64
 5   hours.per.week  30162 non-null  int64
 6   workclass       30162 non-null  int64
 7   education       30162 non-null  int64
 8   marital.status  30162 non-null  int64
```

```
 9   occupation      30162 non-null  int64
 10  relationship    30162 non-null  int64
 11  race            30162 non-null  int64
 12  sex             30162 non-null  int64
 13  native.country  30162 non-null  int64
 14  income          30162 non-null  category
dtypes: category(1), int64(14)
memory usage: 3.5 MB
```

[ ]:
```python
# Importing train_test_split
from sklearn.model_selection import train_test_split
```

[ ]:
```python
# Putting independent variables/features to X
X = df.drop('income',axis=1)
# Putting response/dependent variable/feature to y
y = df['income']
```

[ ]:
```python
X.head(5)
```

[ ]:
```
   age  fnlwgt  education.num  capital.gain  capital.loss  hours.per.week  \
1   82  132870              9             0          4356              18
3   54  140359              4             0          3900              40
4   41  264663             10             0          3900              40
5   34  216864              9             0          3770              45
6   38  150601              6             0          3770              40

   workclass  education  marital.status  occupation  relationship  race  sex  \
1          2         11               6           3             1     4    0
3          2          5               0           6             4     4    0
4          2         15               5           9             3     4    0
5          2         11               0           7             4     4    0
6          2          0               5           0             4     4    1

   native.country
1              38
3              38
4              38
5              38
6              38
```

[ ]:
```python
y.head(5)
```

[ ]:
```
1    0
3    0
4    0
5    0
6    0
```

```
Name: income, dtype: category
Categories (2, int64): [0, 1]
```

[ ]:
```
# Splitting the data into train and test
X_train,X_test,y_train,y_test = train_test_split(X,y)
X_train.head()
```

[ ]:
```
       age  fnlwgt  education.num  capital.gain  capital.loss  hours.per.week  \
31795   48  207982             10             0             0              40
18861   63  113756              9             0             0              40
22081   37  405644              2             0             0              77
27727   43  240504             10             0             0              70
4200    43  397963              9           594             0              16

       workclass  education  marital.status  occupation  relationship  race  \
31795          2         15               4           7             4     2
18861          2         11               4           3             3     4
22081          2          3               3           4             2     4
27727          3         15               2           3             0     4
4200           2         11               0           5             1     4

       sex  native.country
31795    0              38
18861    0              38
22081    1              25
27727    1              38
4200     1              38
```

[ ]:
```
test_size = 0.20
seed = 7
num_folds = 10
scoring = 'accuracy'
# Params for Random Forest
num_trees = 100
max_features = 3
models = []
```

[ ]:
```
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, shuffle=True, random_state=seed)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,␣
↪scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

[ ]:
```
random_forest = RandomForestClassifier(n_estimators=250,max_features=5)
random_forest.fit(X_train, y_train)
predictions = random_forest.predict(X_test)
print("Accuracy: %s%%" % (100*accuracy_score(y_test, predictions)))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
Accuracy: 84.40525129293196%
[[5205  418]
 [ 758 1160]]
              precision    recall  f1-score   support

           0       0.87      0.93      0.90      5623
           1       0.74      0.60      0.66      1918

    accuracy                           0.84      7541
   macro avg       0.80      0.77      0.78      7541
weighted avg       0.84      0.84      0.84      7541
```

[ ]:
```

```

Conclusion:

A correlation heatmap is a graphical representation of a correlation matrix, where each cell in the heatmap represents the correlation between two variables. The correlation values are typically color-coded to help you quickly identify patterns..The correlation heat map obtained from the dataset specifies significant positive correlations between education level and income, suggesting that higher education is associated with higher earnings.

Accuracy obtained in the decision tree model is 84.405%. A confusion matrix is a tabular representation used in machine learning to evaluate the performance of a classification model, especially for binary classification problems.Precision Obtain is 0.87 for 0 and 0.74 for 1, recall obtained is 0.93 for 0 and 0.60 for 1 and f1 score is 0.99 for 0 and 0.66 for 1.