Experiment No. 4
Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

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.

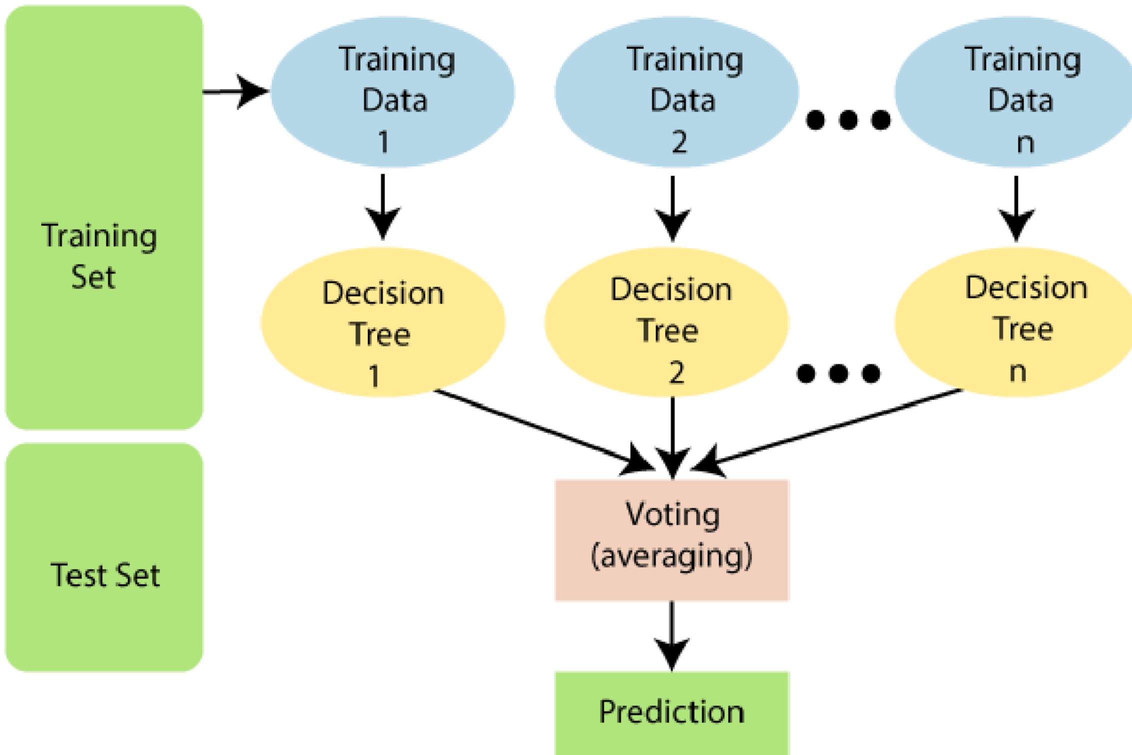
Theory:

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.

Code:

Conclusion:

1. State the observations about the data set from the correlation heat map.

- Positive Correlations: Income is moderately positively correlated with age, education level, capital gains, and hours worked per week.
- Negative Correlations: Income is negatively correlated with being in a relationship or being married.
- Weak Correlations: Many other features, such as workclass, race, and native country, have weak correlations with income.

2. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.

- Accuracy: Model accuracy is about 85.35%.This means the model correctly predicted the class label.
- Confusion Matrix: Detailed breakdown of true positive(4625),true negative(934),false positive(351) and false negative(603)
- Precision: Precision measures the accuracy of the positive predictions and the precision score obtained by our model is 72.68%
- Recall: The recall rate is about 61%.This means that the model can only correctly identify about 61% of all real cases.
- F1 Score: The F1 score of 66% is between precision and recall,providing a balanced view of model performance.

3. Compare the results obtained by applying random forest and decision tree algorithm on the Adult Census Income Dataset

Random Forest algorithm outperforms the Decision Tree on the Adult Census Income Dataset in terms of accuracy, precision, and F1 score. Random Forest provides a better balance between these metrics, making it a stronger choice for this dataset.

ml-exp4

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn.tree import DecisionTreeClassifier
df= pd.read_csv('adult.csv')
```

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

[ ]:
```
   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
```

[ ]:
```
df.shape
```

[ ]: (32561, 15)

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

[ ]:
```
                age     fnlwgt education.num capital.gain capital.loss \
count 32561.000000            32561.000000           32561.000000
3.256100e+04                  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              16.000000 99999.000000 4356.000000
      1.484705e+06
      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
```

[ ]:
```
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['income'].value_counts()
```

[ ]:
```
<=50K 24720
>50K   7841
Name: income, dtype: int64
```

[ ]:
```
df['sex'].value_counts()
```

[ ]:
```
Male   21790
Female 10771
Name: sex, dtype: int64
```

[ ]:
```
df['native.country'].value_counts()
```

[ ]:
```
United-States          29170
Mexico                   643
?                        583
Philippines              198
Germany                  137
Canada                   121
Puerto-Rico              114
El-Salvador              106
India                    100
Cuba                      95
England                   90
Jamaica                   81
South                     80
China                     75
Italy                     73
Dominican-Republic        70
Vietnam                   67
Guatemala                 64
Japan                     62
Poland                    60
Columbia                  59
Taiwan                    51
Haiti                     44
Iran                      43
Portugal                  37
Nicaragua                 34
Peru                      31
```

```
 Greece                          29
 France                          29
 Ecuador                         28
 Ireland                         24
 Hong                            20
 Cambodia                        19
 Trinadad&Tobago                 19
 Laos                            18
 Thailand                        18
 Yugoslavia                      16
 Outlying-US(Guam-USVI-etc)      14
 Hungary                         13
 Honduras                        13
 Scotland                        12
 Holand-Netherlands               1
Name: native.country, dtype: int64
```

[ ]:
```
df['workclass'].value_counts()
```

[ ]:
```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

[ ]:
```
df['occupation'].value_counts()
```

[ ]:
```
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
```

[ ]:
```
df.replace('?', np.NaN,inplace = True)
df.head()
```

[ ]:
```
 age workclass fnlwgt  education education.num marital.status \
0   90 NaN 77053  HS-grad    9     Widowed
1   82 Private 132870  HS-grad    9     Widowed
2   66 NaN 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              NaN Not-in-family White Female      0
1              Exec-managerial Not-in-family White Female     0
2              NaN      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
```

[ ]:
```
#fills missing values (NaNs) with the most recent non-missing value
df.fillna(method = 'ffill', inplace = True)
```

[ ]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder() df['workclass'] =
le.fit_transform(df['workclass']) df['education'] =
le.fit_transform(df['education'])
df['marital.status'] =
le.fit_transform(df['marital.status'])
df['occupation'] =
le.fit_transform(df['occupation'])
df['relationship'] =
le.fit_transform(df['relationship']) df['race'] =
le.fit_transform(df['race']) df['sex'] =
le.fit_transform(df['sex']) df['native.country'] =
le.fit_transform(df['native.country']) df['income']
= le.fit_transform(df['income']) df.head()
```

[ ]:
```
 age workclass fnlwgt education education.num marital.status \
0   90  8     77053 11    9     6
1   82  3 132870   11    9     6
```

```
2   66  3 186061   15    10    6
3   54  3 140359   5     4     0  4   41    3 264663   15    10    5

occupation relationship race sex capital.gain capital.loss \
0          14      1     4     0     0    4356
1          3 1     4     0     0     4356
2          3 4     2     0     0     4356
3          6 4     4     0     0     3900
4          9 3     4     0     0     3900

   hours.per.week native.country income
0            40   38    0
1            18   38    0
2            40   38    0
3            40   38    0
4            40   38    0
```

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

[ ]:
```
               age    workclass      fnlwgt   education education.num \
count 32561.000000 32561.000000 3.256100e+04 32561.00000032561.000000
mean    38.581647    3.102761 1.897784e+05   10.298210    10.080679
std     13.640433    1.136995 1.055500e+05    3.870264     2.572720
min     17.000000    0.000000 1.228500e+04    0.000000     1.000000
25%     28.000000    3.000000 1.178270e+05    9.000000     9.000000
50%     37.000000    3.000000 1.783560e+05   11.000000    10.000000
75%     48.000000    3.000000 2.370510e+05   12.000000    12.000000
max      90.000000    8.000000    1.484705e+06    15.000000    16.000000
      marital.status occupation relationship race sex \

count   32561.000000 32561.000000 32561.000000 32561.000000
        32561.000000
mean       2.611836     5.967108     1.446362     3.665858     0.669205
std        1.506222     4.025021     1.606771     0.848806     0.470506
min        0.000000     0.000000     0.000000     0.000000     0.000000
25%        2.000000     2.000000     0.000000     4.000000     0.000000
50%        2.000000     6.000000     1.000000     4.000000     1.000000
75%        4.000000     9.000000     3.000000     4.000000     1.000000
max        6.000000    14.000000     5.000000     4.000000     1.000000
    capital.gain capital.loss hours.per.week native.country \
count 32561.000000 32561.00000032561.000000 32561.000000
mean  1077.648844    87.303830    40.437456     36.388901
std   7385.292085   402.960219    12.347429      6.110988
min      0.000000     0.000000    1.000000       0.000000
25%      0.000000     0.000000    40.000000     38.000000
50%      0.000000     0.000000    40.000000     38.000000
75%      0.000000     0.000000    45.000000     38.000000
```

```
max   99999.000000  4356.000000    99.000000     40.000000

             income
count
32561.000000
mean     0.240810
std  0.427581  min
0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

[ ]:
```
#Splitting the data set into features and outcome
X = df.drop(['income'], axis=1)
Y = df['income']
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
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
```

[ ]:
```
df.duplicated().sum()
```

[ ]: 24

[ ]:
```
df=df.dropna()
```

[ ]:
```
#Splitting the data into test data and training data

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 = 42)
```

[ ]:
```
X_train.head()
```

[ ]:
```
       age  workclass  fnlwgt  education  education.num  marital.status  \
5514    26          3  256263         11              9               4
19777   24          3  170277         11              9               4
10781   36          3   75826          9             13               0
32240   22          6   24395         15             10               2
9876    31          1  356689          9             13               2

       occupation  relationship  race  sex  capital.gain  capital.loss  \
5514            2             1     4    1             0             0
19777           7             1     4    0             0             0
10781           0             4     4    0             0             0
32240           0             5     4    0             0             0
9876            9             0     4    1             0             0

       hours.per.week  native.country
5514               25              38

19777              35              38

10781              40              38

32240              20              38

9876               40              38
```

[ ]:
```
Y_train = Y_train.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
```

[ ]:
```
Y_train.head()
```

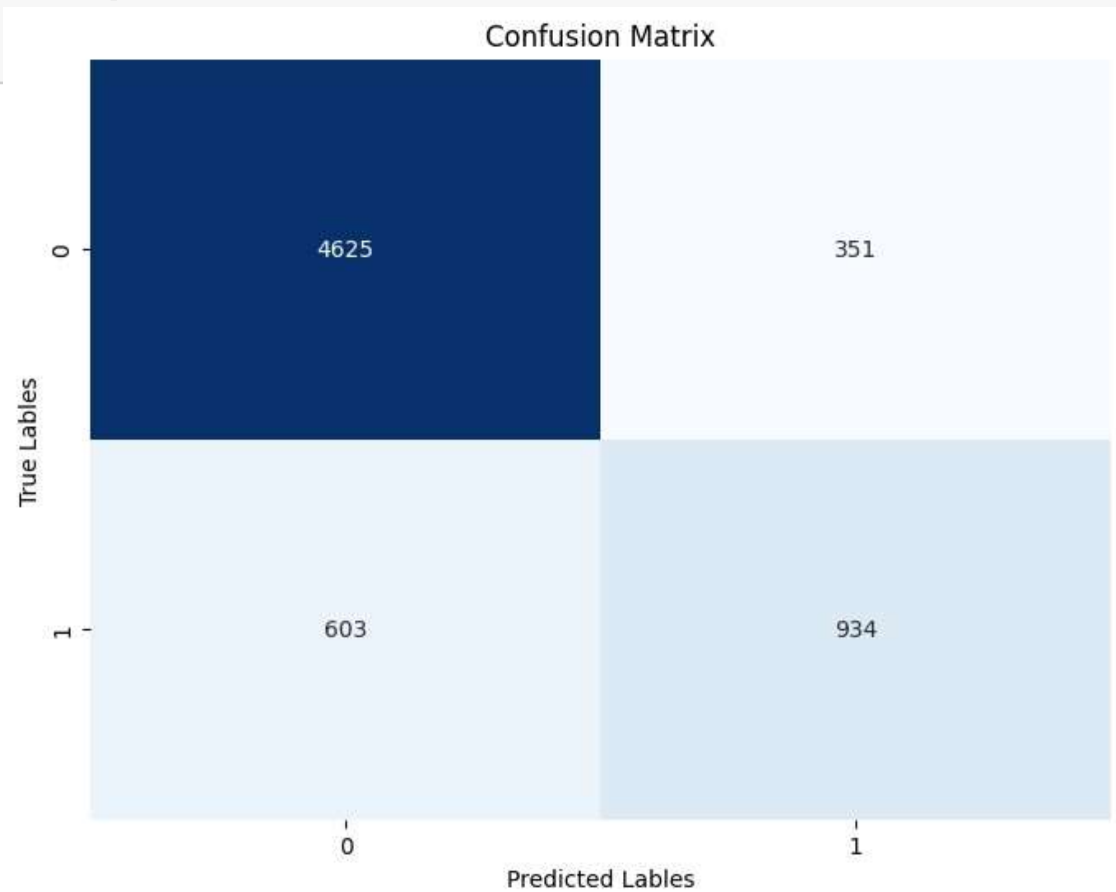
[ ]:
```
0    0
1    0
2    0
3    0
4    0
Name: income, dtype: int64
```

```
[ ]: Y_test = Y_test.replace((np.inf, -np.inf, np.nan),
0).reset_index(drop=True)
```

```
[ ]: from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state = 2022)
model.fit(X_train, Y_train)
Y_pred_dec_tree = model.predict(X_test)
```

```
[ ]: from sklearn.metrics._plot.confusion_matrix import confusion_matrix
conf_matrix = confusion_matrix(Y_test,Y_pred_dec_tree)
```

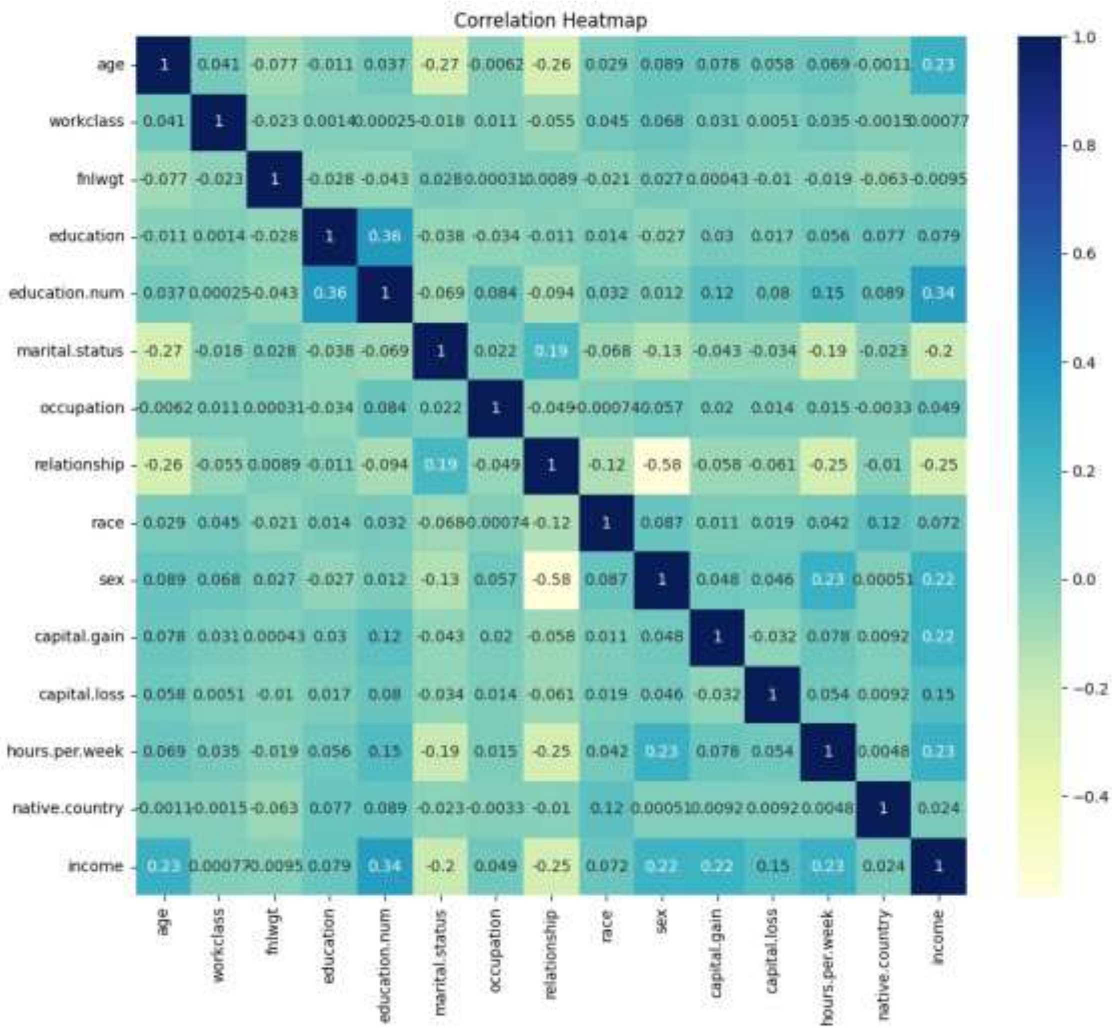
```
[ ]: plt.figure(figsize=(8,6)) sns.heatmap(conf_matrix,annot=True,
fmt="d",   cmap="Blues",   cbar=False)   plt.xlabel("Predicted
Lables")   plt.ylabel("True   Lables")   plt.title("Confusion
Matrix") plt.show()
```



[ ]:
```python
import seaborn as sb
import matplotlib.pyplot as mp
plt.figure(figsize=(12,10))
print(df.corr())
dataplot = sb.heatmap(df.corr(), cmap="YlGnBu", annot=True)
plt.title("Correlation Heatmap")
mp.show()
```

```
                    age workclass   fnlwgt education education.num \
age            1.000000 0.041358 -0.076646 -0.010508    0.036527
workclass      0.041358 1.000000 -0.023077 0.001362     0.000252
fnlwgt       -0.076646 -0.023077 1.000000 -0.028145    -
                                                       0.043195
education     -0.010508 0.001362 -0.028145 1.000000     0.359153
education.num 0.036527 0.000252 -0.043195 0.359153     1.000000
marital.status -0.266288 -0.017704 0.028153 -          -
0.038407                                               0.069304
occupation    -0.006201  0.010604  0.000310 -0.034105    0.083740
relationship-0.263698 -0.054965 0.008931 -0.010876    -
                                                       0.094153
race           0.028718 0.045445 -0.021291 0.014131     0.031838
sex            0.088832  0.067850  0.026858 -0.027356    0.012280

capital.gain   0.077674   0.031075   0.000432   0.030046   0.122630
capital.loss   0.057775   0.005134   -0.010252   0.016746   0.079923
hours.per.week  0.068756  0.035189  -0.018768  0.055510  0.148123
native.country -0.001124 -0.001473 -0.063097 0.076738 0.089082
income          0.234037    0.000774    -0.009463    0.079317    0.335154
               marital.status occupation relationship race sex \

age                 -0.266288 -0.006201   -0.263698 0.028718 0.088832
workclass          -0.017704   0.010604   -0.054965 0.045445 0.067850
fnlwgt              0.028153   0.000310    0.008931 -0.021291 0.026858
education          -0.038407 -0.034105   -0.010876 0.014131 -0.027356
education.num      -0.069304   0.083740   -0.094153 0.031838 0.012280
marital.status      1.000000   0.022352   0.185451 -0.068013 -0.129314
occupation          0.022352   1.000000   -0.048752 -0.000745 0.056935
relationship        0.185451  -0.048752   1.000000 -0.116055 -0.582454
race               -0.068013 -0.000745   -0.116055 1.000000 0.087204
sex                -0.129314   0.056935   -0.582454 0.087204 1.000000
capital.gain       -0.043393   0.020328   -0.057919 0.011145 0.048480
capital.loss       -0.034187   0.013522   -0.061062 0.018899 0.045567
hours.per.week     -0.190519   0.014640   -0.248974 0.041910 0.229309
native.country     -0.022949 -0.003344   -0.010364 0.119375 0.000511
income              -0.199307  0.048913  -0.250918  0.071846  0.215980
               capital.gain       capital.loss       hours.per.week
               native.country \
```

```
age               0.077674    0.057775     0.068756     -0.001124
workclass         0.031075    0.005134     0.035189     -0.001473
fnlwgt            0.000432   -0.010252    -0.018768     -0.063097
education         0.030046    0.016746     0.055510      0.076738
education.num     0.122630    0.079923     0.148123      0.089082
marital.status   -0.043393   -0.034187    -0.190519     -0.022949
occupation        0.020328    0.013522     0.014640     -0.003344
relationship     -0.057919   -0.061062    -0.248974     -0.010364
race              0.011145    0.018899     0.041910      0.119375
sex               0.048480    0.045567     0.229309      0.000511
capital.gain      1.000000   -0.031615     0.078409      0.009212
capital.loss     -0.031615    1.000000     0.054256      0.009240
hours.per.week    0.078409    0.054256     1.000000      0.004796
native.country    0.009212    0.009240     0.004796      1.000000
income             0.223329    0.150526     0.229689      0.024103
                 income

age             0.234037

workclass       0.000774

fnlwgt         -0.009463

education       0.079317

education.num  0.335154
marital.status -0.199307
occupation 0.048913
relationship -0.250918
race   0.071846 sex
  0.215980 capital.gain
  0.223329 capital.loss
  0.150526
hours.per.week 0.229689
native.country 0.024103
income 1.000000
```




```
[ ]: from sklearn.metrics import
accuracy_score from sklearn.metrics
import f1_score from sklearn.metrics
import precision_score from
sklearn.metrics import recall_score
```

```
[ ]: print('Random Forest:') print('Accuracy
score:',accuracy_score(Y_test, Y_pred_dec_tree) * 100)
print('Precision :',precision_score(Y_test,Y_pred_dec_tree)
*100) print('Recall: ',recall_score(Y_test,Y_pred_dec_tree)*
100) print('F1 score: ',f1_score(Y_test,Y_pred_dec_tree)
*100)
```

```
Random Forest:
Accuracy score: 85.35237217871948
```

```
Precision : 72.68482490272373
Recall: 60.76772934287573
F1 score: 66.194188518781
```