In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
df = pd.read_csv('adult.csv')
df.head(10)
```

Out[2]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
6	29	?	227026	HS-grad	9	Never-married	?	Unmarried	Black	Male	0	0	40	United-States	<=50K
7	63	Self-emp-not-inc	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband	White	Male	3103	0	32	United-States	>50K
8	24	Private	369667	Some-college	10	Never-married	Other-service	Unmarried	White	Female	0	0	40	United-States	<=50K
9	55	Private	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	10	United-States	<=50K

In [3]:
```python
df.shape
```

Out[3]:
```
(48842, 15)
```

In [4]:
```python
df.dtypes
```

Out[4]:
```
age                int64
workclass         object
fnlwgt             int64
education         object
educational-num    int64
marital-status    object
occupation        object
relationship      object
race              object
gender            object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
income            object
dtype: object
```

In [5]:
```python
df.isnull().sum()
```

Out[5]:
```
age                0
workclass          0
fnlwgt             0
education          0
educational-num    0
marital-status     0
occupation         0
relationship       0
race               0
gender             0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

In [6]:
```python
df.nunique()
```

Out[6]:
```
age                   74
workclass              9
fnlwgt             28523
education             16
educational-num       16
marital-status         7
occupation            15
relationship           6
race                   5
gender                 2
capital-gain         123
capital-loss          99
hours-per-week        96
native-country        42
income                 2
dtype: int64
```

In [7]:
```python
df['workclass'].value_counts()
```

Out[7]:
```
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
?                    2799
State-gov            1981
Self-emp-inc         1695
Federal-gov          1432
Without-pay            21
Never-worked           10
Name: workclass, dtype: int64
```

In [8]:
```python
df.columns
```

Out[8]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'educational-num',
       'marital-status', 'occupation', 'relationship', 'race', 'gender',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income'],
      dtype='object')
```

In [9]:
```python
df.dtypes
```

Out[9]:
```
age                int64
workclass         object
fnlwgt             int64
education         object
educational-num    int64
marital-status    object
occupation        object
relationship      object
race              object
gender            object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
income            object
dtype: object
```

In [10]:
```python
df['fnlwgt'].nunique()
```

Out[10]:
```
28523
```

In [11]:
```python
df['workclass'] = df['workclass'].replace('?', 'Private')
```

In [12]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [13]:
```python
lb = LabelEncoder()
df['workclass'] = lb.fit_transform(df['workclass'])
```

In [14]:
```python
df.head(10)
```

Out[14]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	3	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	3	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	1	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	3	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	3	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K
5	34	3	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
6	29	3	227026	HS-grad	9	Never-married	?	Unmarried	Black	Male	0	0	40	United-States	<=50K
7	63	5	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband	White	Male	3103	0	32	United-States	>50K
8	24	3	369667	Some-college	10	Never-married	Other-service	Unmarried	White	Female	0	0	40	United-States	<=50K
9	55	3	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	10	United-States	<=50K

In [15]:
```python
df['workclass'].nunique()
```

Out[15]:
```
8
```

In [16]:
```python
df['workclass']
```

Out[16]:
```
0        3
1        3
2        1
3        3
4        3
        ..
48837    3
48838    3
48839    3
48840    3
48841    4
Name: workclass, Length: 48842, dtype: int32
```

In [17]:
```python
df['workclass'].value_counts()
```

Out[17]:
```
3    36705
5     3862
1     3136
6     1981
4     1695
0     1432
7       21
2       10
Name: workclass, dtype: int64
```

In [18]:
```python
df['occupation'] = df['occupation'].replace('?', 'Prof-specialty')
df['native-country'] = df['native-country'].replace('?', 'United-States')
```

In [19]:
```python
df.columns
```

Out[19]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'educational-num',
       'marital-status', 'occupation', 'relationship', 'race', 'gender',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income'],
      dtype='object')
```

In [20]:
```python
df.drop('relationship',axis=1,inplace=True)
df.drop('race',axis=1,inplace=True)
```

In [21]:
```python
df.head(10)
```

Out[21]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	3	226802	11th	7	Never-married	Machine-op-inspct	Male	0	0	40	United-States	<=50K
1	38	3	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Male	0	0	50	United-States	<=50K
2	28	1	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Male	0	0	40	United-States	>50K
3	44	3	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Male	7688	0	40	United-States	>50K
4	18	3	103497	Some-college	10	Never-married	Prof-specialty	Female	0	0	30	United-States	<=50K
5	34	3	198693	10th	6	Never-married	Other-service	Male	0	0	30	United-States	<=50K
6	29	3	227026	HS-grad	9	Never-married	Prof-specialty	Male	0	0	40	United-States	<=50K
7	63	5	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Male	3103	0	32	United-States	>50K
8	24	3	369667	Some-college	10	Never-married	Other-service	Female	0	0	40	United-States	<=50K
9	55	3	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Male	0	0	10	United-States	<=50K

In [22]:
```python
df['education'] = lb.fit_transform(df['education'])
df['workclass'] = lb.fit_transform(df['workclass'])
```

In [23]:
```python
df.head()
```

Out[23]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	3	226802	1	7	Never-married	Machine-op-inspct	Male	0	0	40	United-States	<=50K
1	38	3	89814	11	9	Married-civ-spouse	Farming-fishing	Male	0	0	50	United-States	<=50K
2	28	1	336951	7	12	Married-civ-spouse	Protective-serv	Male	0	0	40	United-States	>50K
3	44	3	160323	15	10	Married-civ-spouse	Machine-op-inspct	Male	7688	0	40	United-States	>50K
4	18	3	103497	15	10	Never-married	Prof-specialty	Female	0	0	30	United-States	<=50K

In [24]:
```python
df['occupation'] = lb.fit_transform(df['occupation'])
df['marital-status'] = lb.fit_transform(df['marital-status'])
df['gender'] = lb.fit_transform(df['gender'])
df['native-country'] = lb.fit_transform(df['native-country'])
```

In [25]:
```python
df.dtypes
```

Out[25]:
```
age                int64
workclass          int64
fnlwgt             int64
education          int32
educational-num    int64
marital-status     int32
occupation         int32
gender             int32
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country     int32
income            object
dtype: object
```

Income Classification

In [26]:
```python
df.income = df.income.replace('<=50K', 0)
df.income = df.income.replace('>50K', 1)
```

In [27]:
```python
df['income'].nunique()
```

Out[27]:
```
2
```

Model Building

In [28]:
```python
x = df.drop('income',axis=1)
y = df['income']
print(type(x))
print(type(y))
print(x.shape)
print(y.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
(48842, 12)
(48842,)
```

In [29]:
```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

In [30]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

In [31]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(36631, 12)
(12211, 12)
(36631,)
(12211,)
```

In [32]:
```python
def gen_cls_metrics(ytest,ypred):
    print('Accuracy Score',accuracy_score(ytest,ypred))
    cm = confusion_matrix(ytest,ypred)
    print(cm)
    print(classification_report(ytest,ypred))

def train_test_score(model):
    print('Training Score',model.score(x_train,y_train))
    print('Testing Score',model.score(x_test,y_test))
```

KNN Model

In [33]:
```python
m1 = KNeighborsClassifier(n_neighbors=23)
m1.fit(x_train,y_train)
```

Out[33]:
```
KNeighborsClassifier(n_neighbors=23)
```

In [34]:
```python
train_test_score(m1)
```

```
Training Score 0.8061204990308755
Testing Score 0.7987879780525755
```

In [35]:
```python
ypred_m1 = m1.predict(x_test)
```

In [36]:
```python
print('Metrics for KNN Classifier')
gen_cls_metrics(y_test,ypred_m1)
```

```
Metrics for KNN Classifier
Accuracy Score 0.7987879780525755
[[9120  144]
 [2313  634]]
              precision    recall  f1-score   support

           0       0.80      0.98      0.88      9264
           1       0.81      0.22      0.34      2947

    accuracy                           0.80     12211
   macro avg       0.81      0.60      0.61     12211
weighted avg       0.80      0.80      0.75     12211
```

LogisticRegression Model

In [42]:
```python
m2 = LogisticRegression(max_iter = 1000)
m2.fit(x_train,y_train)
```

Out[42]:
```
LogisticRegression(max_iter=1000)
```

In [43]:
```python
train_test_score(m2)
```

```
Training Score 0.7920067702219432
Testing Score 0.7895340266972402
```

In [44]:
```python
ypred_m2 = m2.predict(x_test)
```

In [45]:
```python
print('Metrics for Log_Reg Classifier')
gen_cls_metrics(y_test,ypred_m2)
```

```
Metrics for Log_Reg Classifier
Accuracy Score 0.7895340266972402
[[8800  464]
 [2106  841]]
              precision    recall  f1-score   support

           0       0.81      0.95      0.87      9264
           1       0.64      0.29      0.40      2947

    accuracy                           0.79     12211
   macro avg       0.73      0.62      0.63     12211
weighted avg       0.77      0.79      0.76     12211
```

SVM

In [ ]:
```python
m3 = SVC(kernel='linear',C=0.1)
m3.fit(x_train,y_train)
```

In [ ]:
```python
train_test_score(m3)
```

In [ ]:
```python
ypred_m3 = m3.predict(x_test)
```

In [ ]:
```python
print('Metrics for SVM Classifier')
gen_cls_metrics(y_test,ypred_m3)
```

Decision Tree Classifier

In [55]:
```python
print(x_train.shape)
print(y_train.shape)
```

```
(36631, 12)
(36631,)
```

In [56]:
```python
m4 = DecisionTreeClassifier(criterion='gini',max_depth=5)
m4.fit(x_train,y_train)
```

Out[56]:
```
DecisionTreeClassifier(max_depth=5)
```

In [57]:
```python
train_test_score(m4)
```

```
Training Score 0.8463323414594196
Testing Score 0.8407992793383016
```

In [58]:
```python
ypred_m4 = m4.predict(x_test)
```

In [59]:
```python
print('Metrics for Decistion Tree Classifier')
gen_cls_metrics(y_test,ypred_m4)
```

```
Metrics for Decistion Tree Classifier
Accuracy Score 0.8407992793383016
[[8782  482]
 [1462 1485]]
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      9264
           1       0.75      0.50      0.60      2947

    accuracy                           0.84     12211
   macro avg       0.81      0.73      0.75     12211
weighted avg       0.83      0.84      0.83     12211
```

RandomForest Classifier

In [60]:
```python
m5 = RandomForestClassifier(n_estimators=70,criterion='gini',max_depth=5)
m5.fit(x_train,y_train)
```

Out[60]:
```
RandomForestClassifier(max_depth=5, n_estimators=70)
```

In [61]:
```python
train_test_score(m5)
```

```
Training Score 0.8530752641205537
Testing Score 0.8507083776922447
```

In [62]:
```python
ypred_m5 = m5.predict(x_test)
```

In [63]:
```python
print('Metrics for random Forest Classifier')
gen_cls_metrics(y_test,ypred_m5)
```

```
Metrics for random Forest Classifier
Accuracy Score 0.8507083776922447
[[8988  276]
 [1547 1400]]
              precision    recall  f1-score   support

           0       0.85      0.97      0.91      9264
           1       0.84      0.48      0.61      2947

    accuracy                           0.85     12211
   macro avg       0.84      0.72      0.76     12211
weighted avg       0.85      0.85      0.83     12211
```

Conclusion
Therefore the model with best accuracy is Random Forest Classifier with an accuracy of 85%

In [56]:

In [ ]:

In [ ]: