CENSUS INCOME PROJECT

CONTENT:

- PROBLEM STATEMENT
- DATA ANALYSIS
- EDA
- PRE-PROCESSING PIPELINE
- MODEL BUILDING
- CONCLUSION

AUTHOR: M.UMA MAHESWARI

Census Income Project

PROBLEM STATEMENT:

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). The prediction task is to determine whether a person makes over $50K a year.

Description of fnlwgt (final weight) The weights on the Current Population Survey (CPS) files are controlled to independent estimates of the civilian non-institutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau. We use 3 sets of controls. These are:

1.A single cell estimate of the population 16+ for each state.

2.Controls for Hispanic Origin by age and sex.

3.Controls by Race, age and sex.

We use all three sets of controls in our weighting program and "rake" through them 6 times so that by the end we come back to all the controls we used. The term estimate refers to population totals derived from CPS by creating "weighted tallies" of any specified socio-economic characteristics of the population. People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state.

UPLOADING ALL THE LIBRARIES:

- The necessary libraries are uploaded where pandas are used to import the dataset and create the dataframe

- Matplot and seaborn are used to for comparing and analysing each coloumn.
- Sklearn is used for building the model to predict its accuracy.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.stats import zscore
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")
```

IMPORTING THE DATASET:

We have to import the necessary CSV file to build the model

In [2]:

```
df = pd.read_csv("https://raw.githubusercontent.com/dsrscientist/dataset1/master/census_income.csv")
```

In [3]:

```
df.head()
```

Out[3]:

gt	Education	Education_num	Marital_status	Occupation	Relationship	Race	Sex	Capital_gain	Capital_loss	Hours_per_week	Native_country	Income
11	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
46	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
21	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
09	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
82	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K

In [4]:

```
df.shape
```

Out[4]: (32560, 15)

With the CSV file we study the feature and the target variable and able to know about categorical and continuous data.In our dataset Income column is our target variable to predict whether a person makes over 50k a year.

By using shape we can come to know about the no. of rows and columns present in our dataset.

EDA(EXPLORATORY DATA ANALYSIS)

While analyzing the data we will look into the null values, unique values, and value count for each column.

We use an encoding technique to fill the null values

If there is unnecessary information or very less data present we drop the column.

We replace ? with the relevant data from the data. And encode the categorical data

Income

Encoding the categorical data

In [22]:
```
le = LabelEncoder()
df["Income"] = le.fit_transform(df["Income"])
df.head()
```

In [23]:
```
# Ordinal Encoder

oe = OrdinalEncoder()
df['Workclass'] = oe.fit_transform(df['Workclass'].values.reshape(-1,1))
df['Education'] = oe.fit_transform(df['Education'].values.reshape(-1,1))
df['Marital_status'] = oe.fit_transform(df['Marital_status'].values.reshape(-1,1))
df['Occupation'] = oe.fit_transform(df['Occupation'].values.reshape(-1,1))
df['Relationship'] = oe.fit_transform(df['Relationship'].values.reshape(-1,1))
df['Native_country'] = oe.fit_transform(df['Native_country'].values.reshape(-1,1))
df['Race'] = oe.fit_transform(df['Race'].values.reshape(-1,1))
df['Sex'] = oe.fit_transform(df['Sex'].values.reshape(-1,1))
df.head()
```

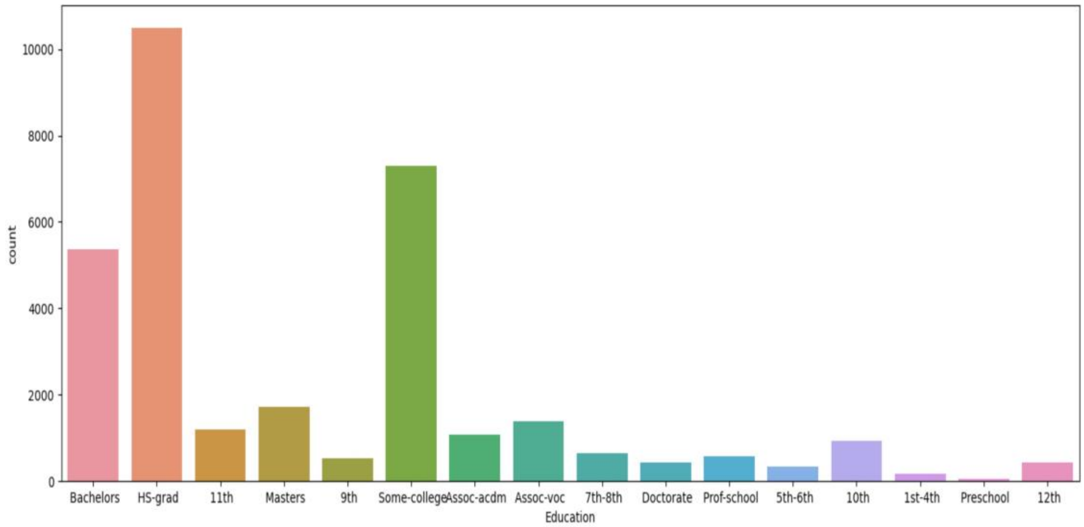
Out[23]:

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	Race	Sex	Capital_gain	Capital_loss	Hours_per_week
0	50	6.0	83311	9.0	13	2.0	3.0	0.0	4.0	1.0	0	0	13
1	38	4.0	215646	11.0	9	0.0	5.0	1.0	4.0	1.0	0	0	40
2	53	4.0	234721	1.0	7	2.0	5.0	0.0	2.0	1.0	0	0	40
3	28	4.0	338409	9.0	13	2.0	10.0	5.0	2.0	0.0	0	0	40
4	37	4.0	284582	12.0	14	2.0	3.0	5.0	4.0	0.0	0	0	40

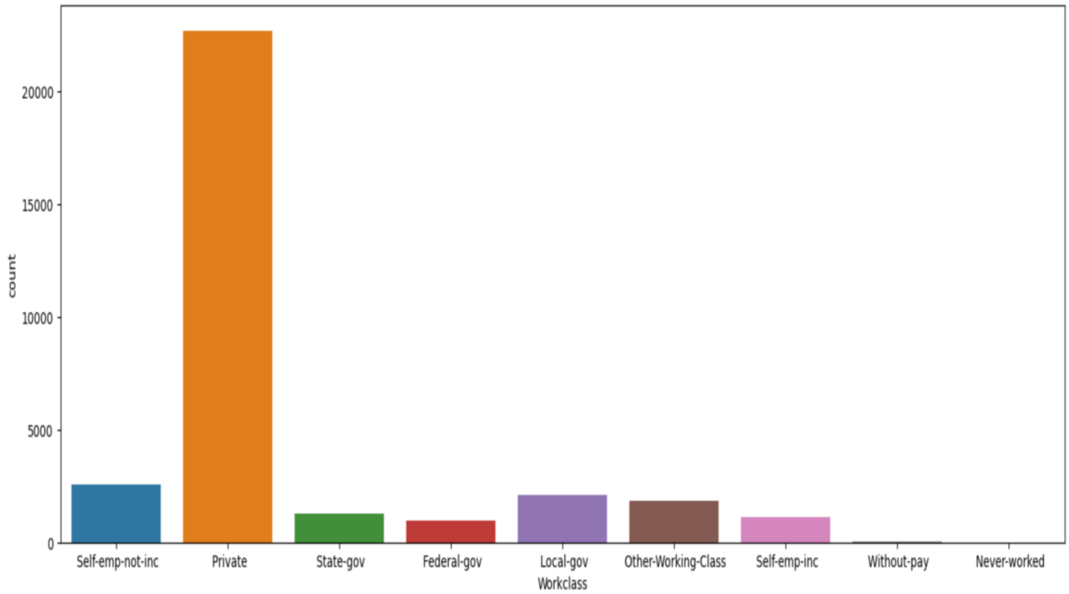
VISUALIZATION

Here we visually analyze each column and also compare the feature column with the target column. and using the multivariant analysis we check the relationship between every column. we can gather maximum information between the feature and target column and its correlation. From the heatmap, we can get information on whether multicollinearity exists or not. And also we can know whether the variable is positively correlated or negatively corelated.

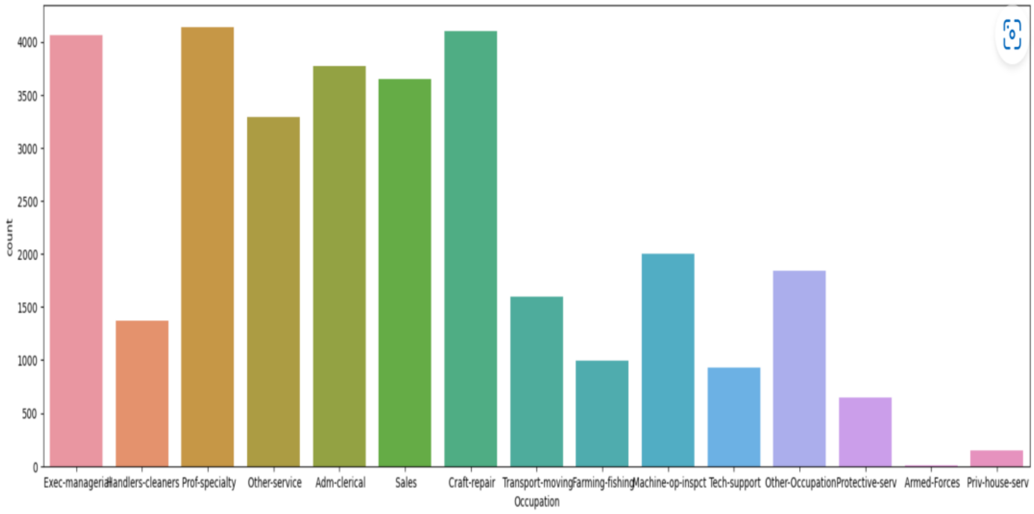
In [14]:
```
plt.figure(figsize=(18,6))
sns.countplot(df["Education"])
plt.show()
```



In [13]:
```
plt.figure(figsize=(18,6))
sns.countplot(df["Workclass"])
plt.show()
```

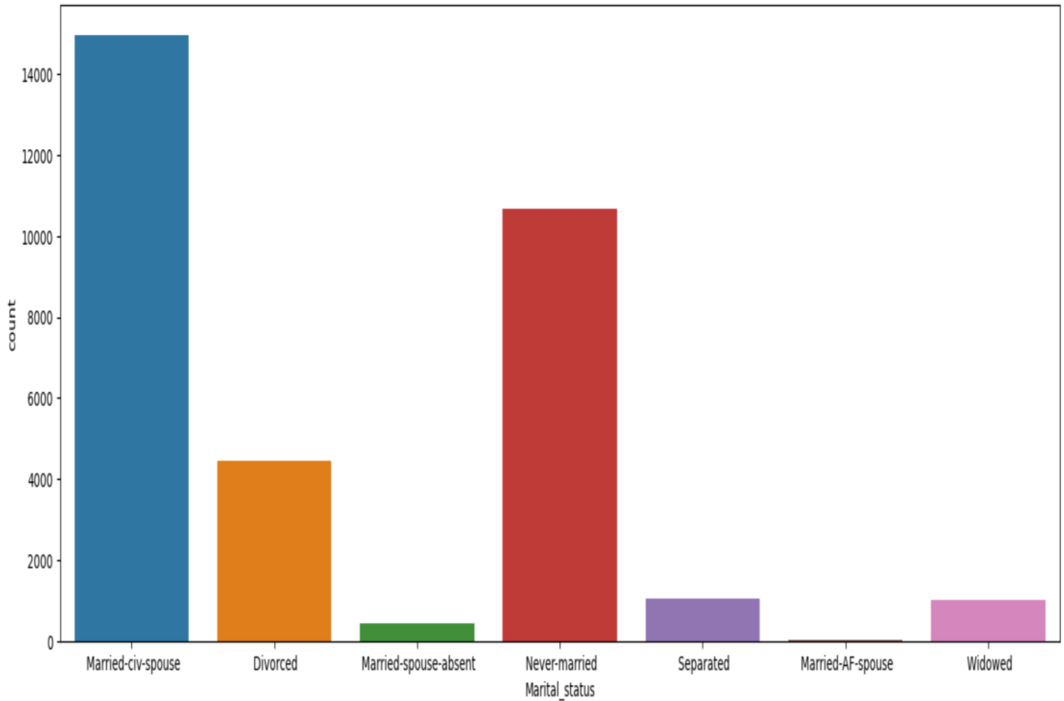


In [16]:
```
plt.figure(figsize=(22,6))
sns.countplot(df["Occupation"])
plt.show()
```

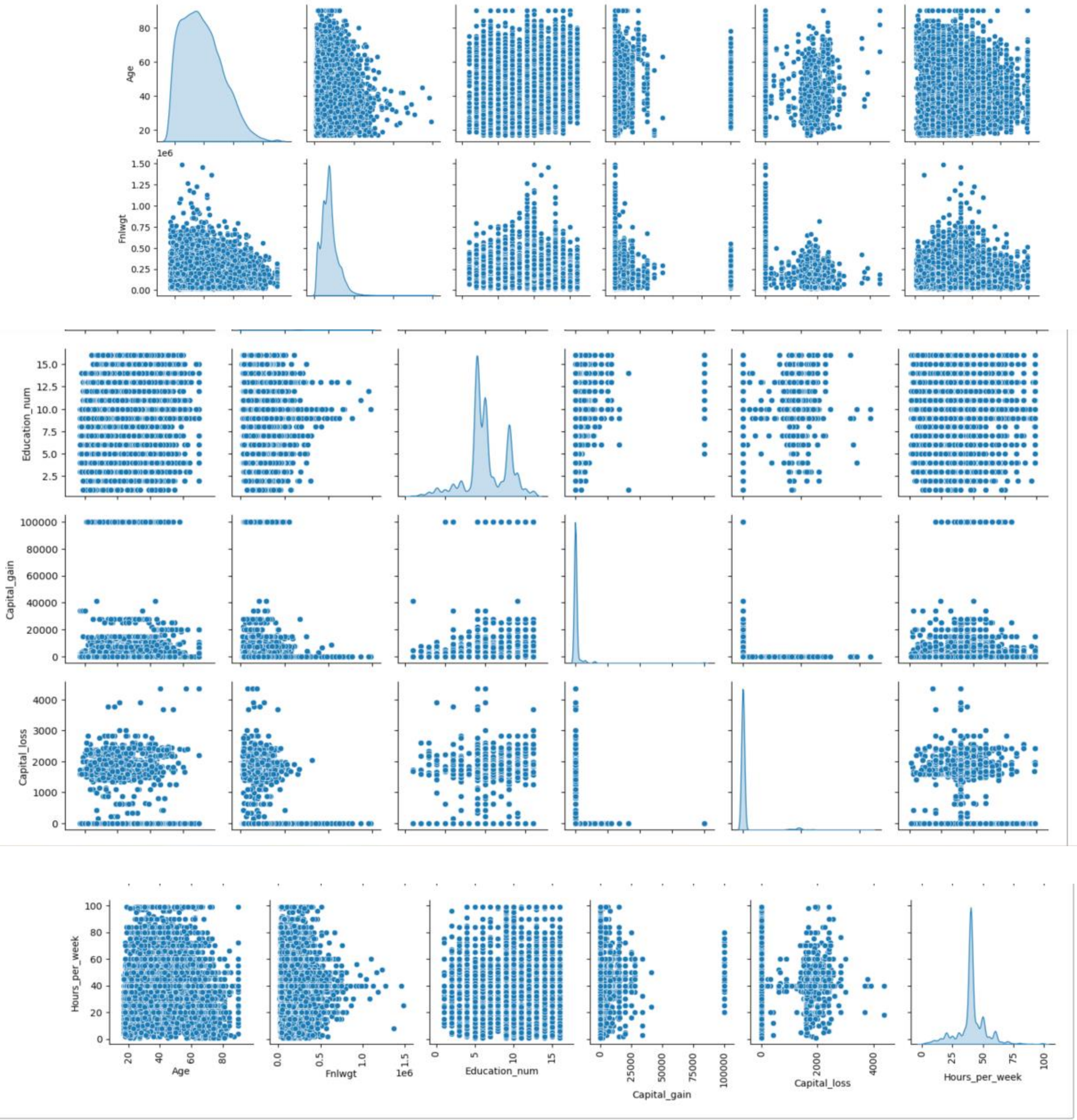


Education

In [15]:
```
plt.figure(figsize=(18,6))
sns.countplot(df["Marital_status"])
plt.show()
```



In [27]:
```python
numvalues = df.drop(['Workclass', 'Education', 'Marital_status', 'Occupation', 'Relationship', 'Race', 'Sex',
                     'Native_country', 'Income'], axis=1)
plt.style.use('default')
g = sns.pairplot(numvalues, diag_kind="kde")
for ax in g.axes.flat:
    ax.tick_params("x", labelrotation=90)
plt.show()
```



Heatmap ¶

In [28]:
```python
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, fmt='0.3f')
plt.show()
```

CONCLUSION FROM EDA:

- ✓ Most of the people from their 40 will make more than 50k per year.
- ✓ The Education num is 13 for the people who have more than 50k per annum
- ✓ People with a high profession will earn 50k per annum.
- ✓ People who work 40 to 60 hours per week earn above 50kper annum
- ✓ People who are earning >50K are mostly from the relationship status husband or wife.
- ✓ we can see that the lower education number is almost negligible for people with income >50K and it, therefore, emphasizes the importance of education too.
- ✓ For the Work Class column, the highest number of people working for the private sector and the other work classes or people who are unemployed is quite less to negligible.
- ✓ Nearly 10.4% of divorced column get >50k per annum.
- ✓ Only few members of unmarried crossed >50 per annum.
- ✓ People with capital gain more than 13k achieve >50k per annum.
- ✓ we can see the skewness details present in our numerical data columns which need to be treated.
- ✓ With the correlation details, we can determine that there is no multi colinearity issue between our columns.

- ✓ we see that columns such as relationship and marital status are the only one's negatively correlated rest all the other feature columns are positively correlated with our label column.
- ✓ we can see the outlier details present in our numerical data columns which should be treated

PRE-PROCESSING PIPELINE

Splitting Feature abd Target variable

```
In [38]:  X = df.drop('Income', axis=1)
          Y = df['Income']
```

Balancing our Target variable

```
In [39]:  Y.value_counts()

Out[39]:  0    21552
          1     6230
          Name: Income, dtype: int64

In [40]:  oversample = SMOTE()
          X, Y = oversample.fit_resample(X, Y)

In [41]:  Y.value_counts()
Out[41]:  0    21552
          1    21552
          Name: Income, dtype: int64
```

Next we split our feature and target variable to train and test our data.

In this dataset,our target variable dataset is imbalanced so we use oversampling technique to balance our dataset.

Feature Scaling

```
In [65]:  scaler = StandardScaler()
          X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
          X.head()
```

We scale our data using a standard scaler.

MODEL BUILDING:

Model Building

In [43]:
```
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=i)
    lr=LogisticRegression()
    lr.fit(X_train, Y_train)
    pred = lr.predict(X_test)
    acc_score = (accuracy_score(Y_test, pred))*100

    if acc_score>maxAccu:
        maxAccu=acc_score
        maxRS=i

print("Best accuracy score is", maxAccu,"on Random State =", maxRS)
```

```
Best accuracy score is 78.00668151447661 on Random State = 245
```

First, we find the Best accuracy score at the random state and we got 78% at random state 245. We also look into the classification report, F1 score for each model and we select the best model based on their report. we use different models to predict the accuracy of the classification problem.

In [45]:
```
# Logistic Regression

model=LogisticRegression()
classify(model, X, Y)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.76      0.77      4312
           1       0.77      0.79      0.78      4309

    accuracy                           0.78      8621
   macro avg       0.78      0.78      0.78      8621
weighted avg       0.78      0.78      0.78      8621

Accuracy Score: 77.6824034334764
Cross Validation Score: 76.6587849612356

Accuracy Score - Cross Validation Score is 1.0236184722407984
```

With logistic regression we got accuracy of 77.68%

```
In [46]:  # Support Vector Classifier

          model=SVC(C=1.0, kernel='rbf', gamma='auto', random_state=41)
          classify(model, X, Y)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.79      0.84      4312
           1       0.81      0.91      0.86      4309

    accuracy                           0.85      8621
   macro avg       0.85      0.85      0.85      8621
weighted avg       0.85      0.85      0.85      8621

Accuracy Score: 84.75814870664657
Cross Validation Score: 84.04791865543885

Accuracy Score - Cross Validation Score is 0.7102300512077164
```

With support vector we got an accuracy of 84%

```
In [47]:  # Decision Tree Classifier

          model=DecisionTreeClassifier(random_state=21, max_depth=15)
          classify(model, X, Y)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.83      0.86      4312
           1       0.84      0.89      0.87      4309

    accuracy                           0.86      8621
   macro avg       0.86      0.86      0.86      8621
weighted avg       0.86      0.86      0.86      8621

Accuracy Score: 86.24289525577079
Cross Validation Score: 86.08256641972025

Accuracy Score - Cross Validation Score is 0.16032883605053883
```

With Decision Tree we got an accuracy of 86.02%

```
In [48]:  # Random Forest Classifier

          model=RandomForestClassifier(max_depth=15, random_state=41)
          classify(model, X, Y)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.84      0.88      4312
           1       0.85      0.93      0.89      4309

    accuracy                           0.89      8621
   macro avg       0.89      0.89      0.89      8621
weighted avg       0.89      0.89      0.89      8621

Accuracy Score: 88.55121215636237
Cross Validation Score: 88.00581863043651

Accuracy Score - Cross Validation Score is 0.54539352592586
```

With Random forest we got accuracy of 88.5%

```
In [50]:  # Extra Trees Classifier

          model=ExtraTreesClassifier()
          classify(model, X, Y)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.89      0.91      4312
           1       0.90      0.93      0.91      4309

    accuracy                           0.91      8621
   macro avg       0.91      0.91      0.91      8621
weighted avg       0.91      0.91      0.91      8621

Accuracy Score: 91.04512237559447
Cross Validation Score: 90.29098588645704

Accuracy Score - Cross Validation Score is 0.7541364891374371
```

With ExtraTrees classifier we got accuracy of 91%

```
In [51]:  # LGBM Classifier
          !pip install lightgbm
          import lightgbm as lgb
          model=lgb.LGBMClassifier()
          classify(model, X, Y)
```

```
Requirement already satisfied: lightgbm in c:\users\apkar\anaconda3\lib\site-packages (3.3.3)
Requirement already satisfied: wheel in c:\users\apkar\anaconda3\lib\site-packages (from lightgbm) (0.37.1)
Requirement already satisfied: scipy in c:\users\apkar\anaconda3\lib\site-packages (from lightgbm) (1.9.1)
Requirement already satisfied: numpy in c:\users\apkar\anaconda3\lib\site-packages (from lightgbm) (1.21.5)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\apkar\anaconda3\lib\site-packages (from lightgbm) (1.1.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\apkar\anaconda3\lib\site-packages (from scikit-learn!=0.22.0
->lightgbm) (2.2.0)
Requirement already satisfied: joblib>=1.0.0 in c:\users\apkar\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->light
gbm) (1.1.0)

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90      4312
           1       0.89      0.91      0.90      4309

    accuracy                           0.90      8621
   macro avg       0.90      0.90      0.90      8621
weighted avg       0.90      0.90      0.90      8621

Accuracy Score: 89.94316204616634
Cross Validation Score: 88.55102753191836
```

With LGBM claasifier we got 89.9% accuracy

From all the above models we chose Extra tree classifier to tune our Hyper parameter

Using the Gini or Entrophy and find the best accuracy score

Hyperparameter Tuning

In [52]:
```python
# Extra Trees Classifier

fmod_param = {'criterion' : ['gini', 'entropy'],
              'n_jobs' : [-2, -1, 1],
              'random_state' : [42, 739, 1000],
              'max_depth' : [0, 15, 30],
              'n_estimators' : [100, 200, 300]
             }
```

In [53]:
```python
GSCV = GridSearchCV(ExtraTreesClassifier(), fmod_param, cv=5)
```

In [54]:
```python
GSCV.fit(X_train,Y_train)
```

Out[54]:
```
GridSearchCV(cv=5, estimator=ExtraTreesClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [0, 15, 30],
                         'n_estimators': [100, 200, 300], 'n_jobs': [-2, -1, 1],
                         'random_state': [42, 739, 1000]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [55]:
```python
GSCV.best_params_
```

Out[55]:
```
{'criterion': 'gini',
 'max_depth': 30,
 'n_estimators': 300,
 'n_jobs': -2,
 'random_state': 42}
```

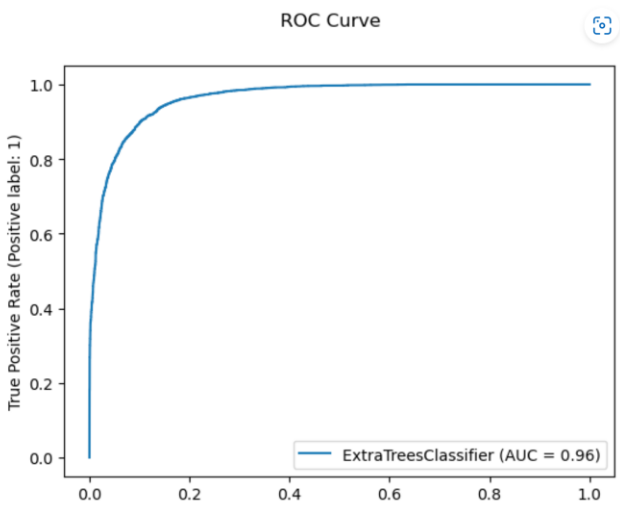
In [56]:
```python
Final = ExtraTreesClassifier(criterion='gini', max_depth=30, n_estimators=300, n_jobs=-2, random_state=1000)
Classifier = Final.fit(X_train, Y_train)
fmod_pred = Final.predict(X_test)
fmod_acc = (accuracy_score(Y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
```

```
Accuracy score for the Best Model is: 89.66221232368225
```

Our model predicts accuracy of 89.66%

With the hyperparameter tuning using the Extra tree classifier,we got an accuracy of 89.6%. which is pretty good.

```python
plt.show()
```



We also need find the AUC-ROC curve which gives 96%

saving the model

```
In [58]:  import pickle
          filename = "Census Income.pkl"
          pickle.dump(Final,open(filename,'wb'))

In [ ]:
```

Finally, we save the model using a pickle.

CONCLUSION:

From all the above models, we find Extra tree classifier predicts well and shows an accuracy of 89.6% and the AUC-ROC curve predicts 96%. And finally we save the model.