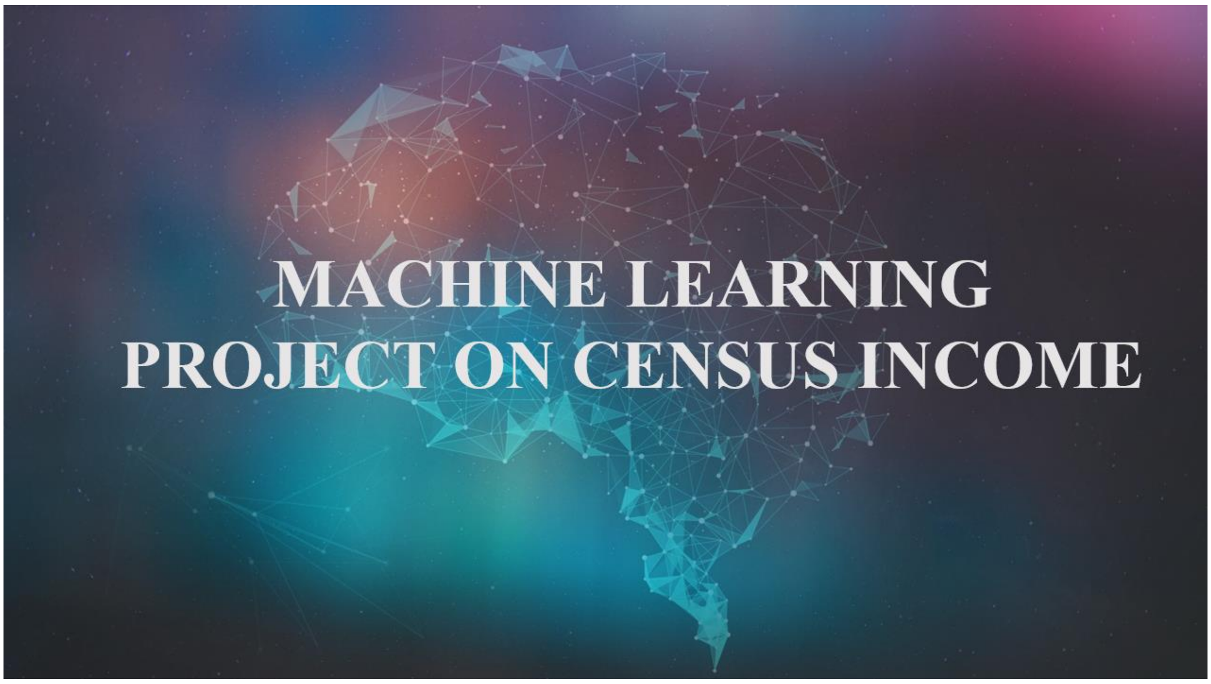
CENSUS INCOME PROJECT



In this blog, we will analyse the data that was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker.

Our task is to predict whether a person makes over $50K a year.

We can get dataset by using the link below:

https://raw.githubusercontent.com/dsrscientist/dataset1/master/census_income.csv.

Let’s start our project by knowing what is machine learning.

Machine Learning:

Machine learning is a capacity of the machines to analyse a set of data and build generic algorithms. There is no need to write codes, just feeding of data is enough it builds its logic based on it.

There are majorly two types of machine learning

- Supervised learning
- Un supervised learning

Supervised Learning

From the above context itself, it is so much evidence that there must be some supervisor as a teacher to carry on the process. Usually in this methodology, the training or the teaching is provided.

For Example, if it is to train to identify various kinds of fruits it must be like the shape of the fruit is round and a cavity is found at the top centre and the colour must be red. Which signifies Apple? If the shape is curved and long enough with the colour of green or yellow then it must be Banana. Now after this training it is given with another set of examples to identify.

Un Supervised Learning

This second type of learning in which there is no supervision or guidance required is called unsupervised learning. Here it can act without any guidance required. For example, if a picture of dogs and cats is given together to analyse, it has no information on Cats or Dogs. But still, it can categorize based on the similarities between them by analysing the patterns, Size, shape, figures, and differences.

Now let’s see the steps that we follow while building our model

- Problem Definition.
- Data Analysis.
- Building Machine Learning Models.
- Model Deployment
- Concluding Remarks.

Problem Definition:

Our requirement is to build best model that can predict whether a person makes over $50k based on the information present in dataset like age, work class, education, occupation, work hours per week and many more aspects.

Now, let’s begin our model building by importing all the required libraries

```
import sklearn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

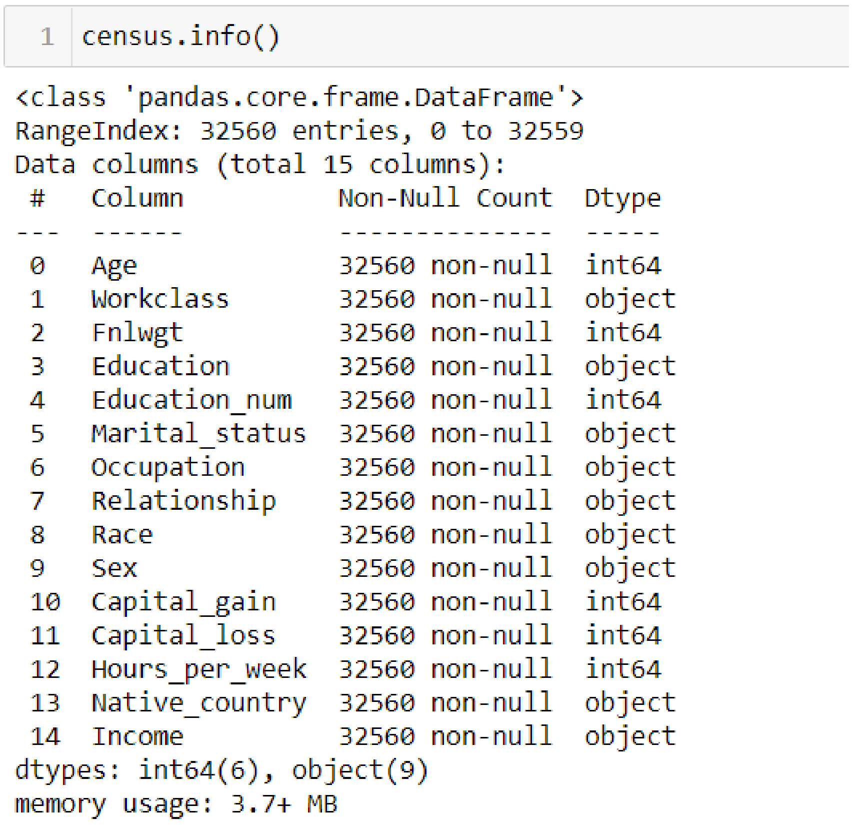
In the next step we need to extract the downloaded dataset by using pandas.

```
census = pd.read_csv('D:\DataTrained Projects\Evaluation Projects\Week 2/census_income.csv')
```

<u>Data Analysis</u>:

Now, we will start our analysis on data.

The size of our dataset is 32560 rows × 15 columns which means we have 32560 samples
We can see that size of all columns is same, so there might me no missing values. Most of the columns are in object datatype, so we need to convert all categorical columns to numerical columns.



```
census.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Age             32560 non-null  int64 
 1   Workclass       32560 non-null  object
 2   Fnlwgt          32560 non-null  int64 
 3   Education       32560 non-null  object
 4   Education_num   32560 non-null  int64 
 5   Marital_status  32560 non-null  object
 6   Occupation      32560 non-null  object
 7   Relationship    32560 non-null  object
 8   Race            32560 non-null  object
 9   Sex             32560 non-null  object
 10  Capital_gain    32560 non-null  int64 
 11  Capital_loss    32560 non-null  int64 
 12  Hours_per_week  32560 non-null  int64 
 13  Native_country  32560 non-null  object
 14  Income          32560 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Age – Gives information about age of a person

Workclass – What kind of work a person do?

Fnlwgt –

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:

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

Controls for Hispanic Origin by age and sex.

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.

Education – What is the educational qualification

Education_num – How many years a person spent on education.

Marital_status – Whether a person is married or single.

Occupation - Profession of a person.

Relationship – What is the position of that person in the family.

Race – What is the race of a person.

Sex – Whether a person is male or female.

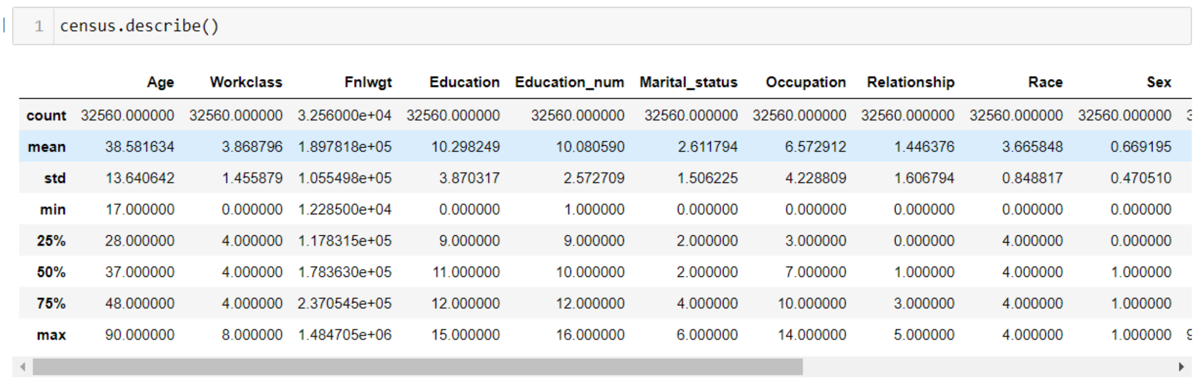
Capital_gain – Income from other investments other than salary.

Capital_loss – Loss incurred from investments made.

Hours_per_week – Number of hours a person works in a week.

Native_country – Native country of a person

Income – Income made by person in a year (whether <=50k or >=50k)



```
census.describe()
```

	Age	Workclass	Fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	Race	Sex
count	32560.000000	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581634	3.868796	1.897818e+05	10.298249	10.080590	2.611794	6.572912	1.446376	3.665848	0.669195
std	13.640642	1.455879	1.055498e+05	3.870317	2.572709	1.506225	4.228809	1.606794	0.848817	0.470510
min	17.000000	0.000000	1.228500e+04	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	28.000000	4.000000	1.178315e+05	9.000000	9.000000	2.000000	3.000000	0.000000	4.000000	0.000000
50%	37.000000	4.000000	1.783630e+05	11.000000	10.000000	2.000000	7.000000	1.000000	4.000000	1.000000
75%	48.000000	4.000000	2.370545e+05	12.000000	12.000000	4.000000	10.000000	3.000000	4.000000	1.000000
max	90.000000	8.000000	1.484705e+06	15.000000	16.000000	6.000000	14.000000	5.000000	4.000000	1.000000

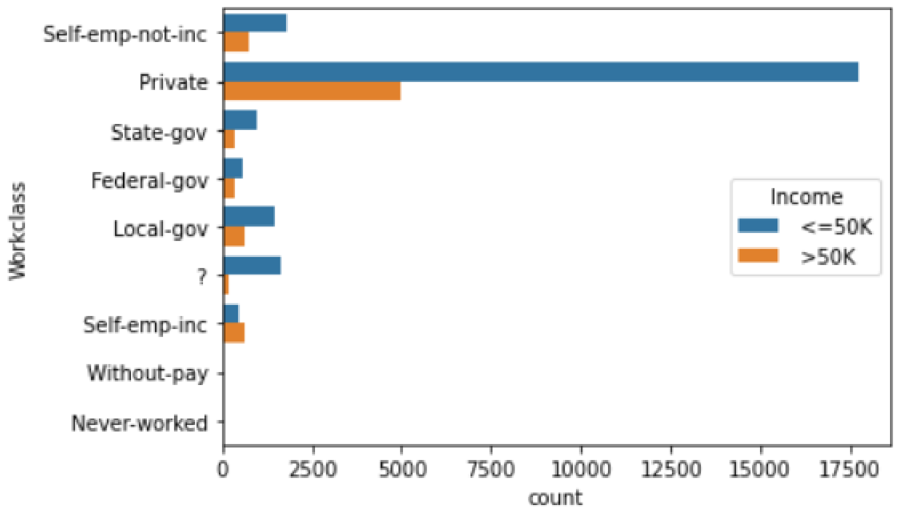
We can see that capital_gain and capital_gain are having difference betw een 50% quartile and mean values, so there is a chance to have outliers in those columns.

Data Visualization :

In data visualization we mostly used countplot from seaborn.

```
sns.countplot(y='Workclass',hue='Income',data=census)
```

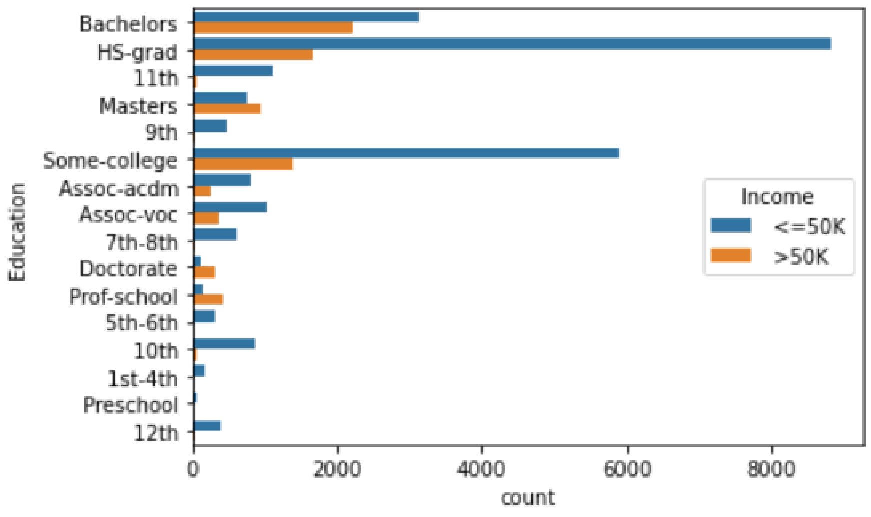
```
<AxesSubplot:xlabel='count', ylabel='Workclass'>
```



Among all the employees, Private employees make more income in both less than 50k and morethan50k.

```
sns.countplot(y='Education',hue='Income',data=census)
```

```
<AxesSubplot:xlabel='count', ylabel='Education'>
```

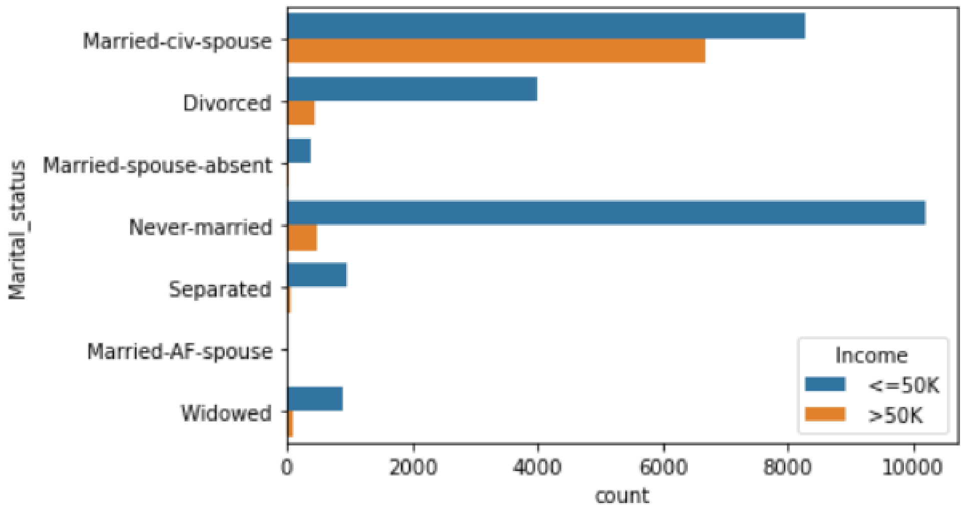


Employees with bachelors degree are more with income greater than $50k

Employees with HS graduation are more with income less than $50k

```
sns.countplot(y='Marital_status',hue='Income',data=census)
```

```
<AxesSubplot:xlabel='count', ylabel='Marital_status'>
```

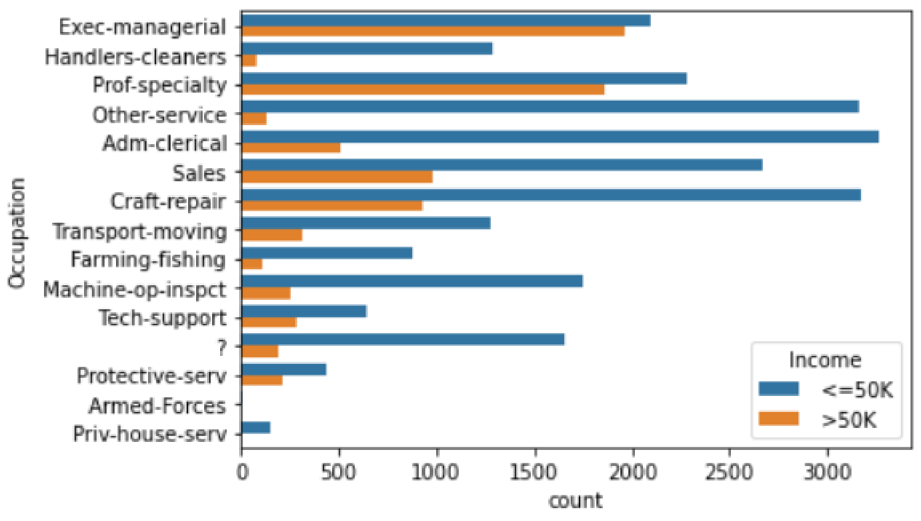


Employees who married-civ-spouse are more with income greaterthan $50k

Never married people are more with income lessthan $50K

```
sns.countplot(y='Occupation',hue='Income',data=census)
```

```
<AxesSubplot:xlabel='count', ylabel='Occupation'>
```

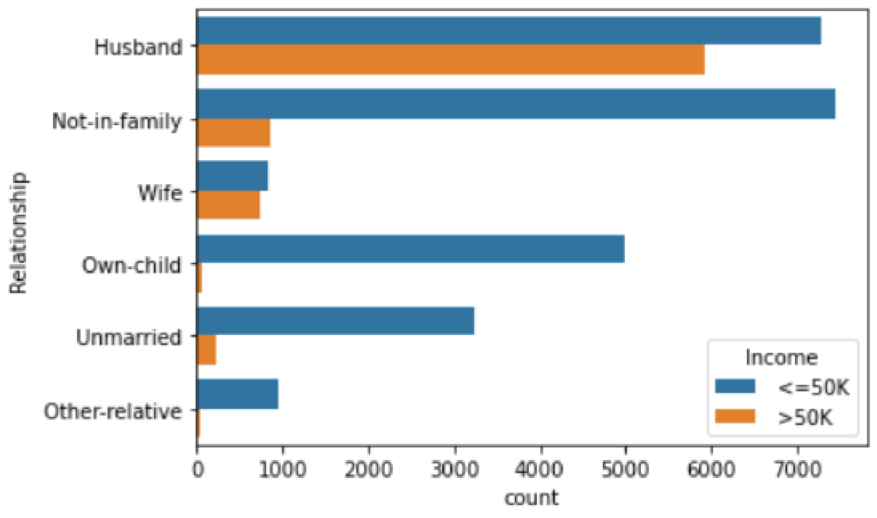


Executive-Managerial employees are more in number who makes income of more than $50k.

Admin-clerical employees are more in number who makes income lessthan $50k.

```
sns.countplot(y='Relationship',hue='Income',data=census)
```

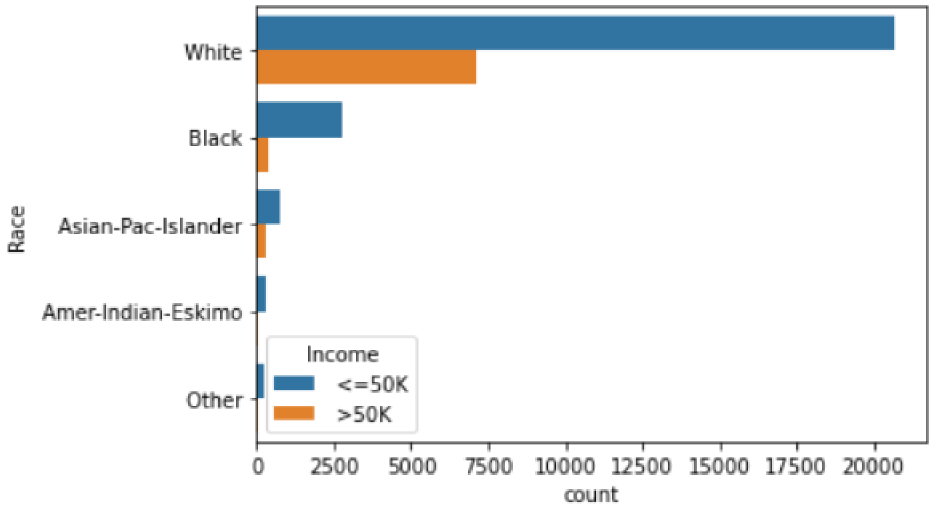
```
<AxesSubplot:xlabel='count', ylabel='Relationship'>
```



Husbands are more in number who makes income more than $50K.

```
sns.countplot(y='Race',hue='Income',data=census)
```

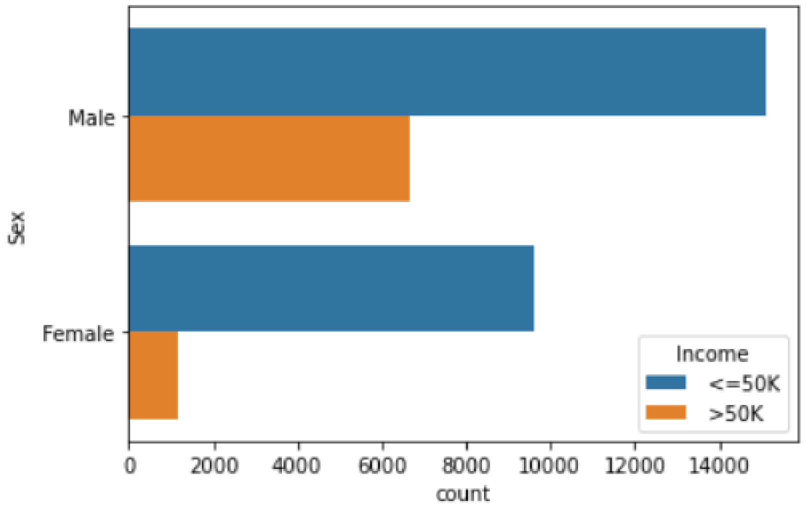
```
<AxesSubplot:xlabel='count', ylabel='Race'>
```



Among all the employees, more number of white people makes income both less than 50k and more than 50k in a year.

```
sns.countplot(y='Sex',hue='Income',data=census)
```

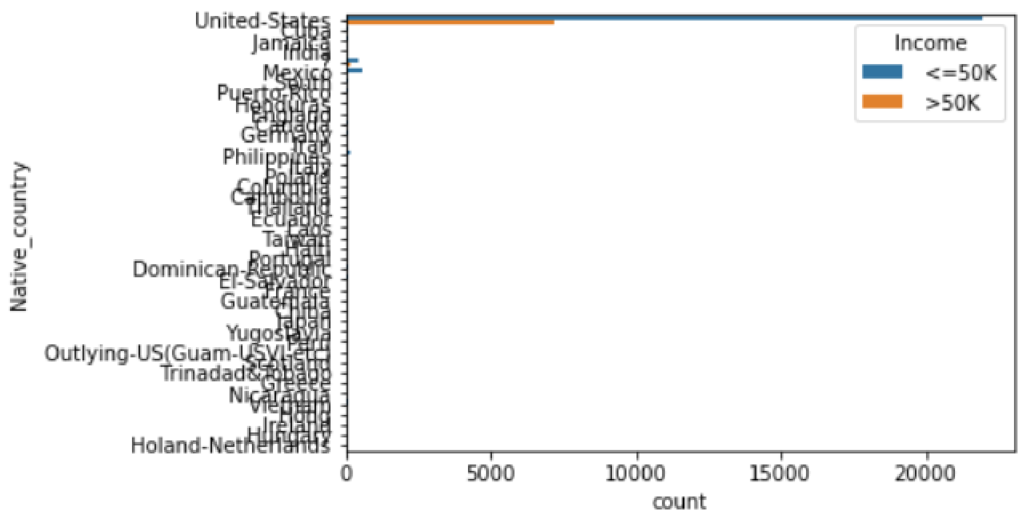
```
<AxesSubplot:xlabel='count', ylabel='Sex'>
```



Among all the employees, more number of male persons makes income both less than $50k and more than 50k in a year.

```
sns.countplot(y='Native_country',hue='Income',data=census)
```

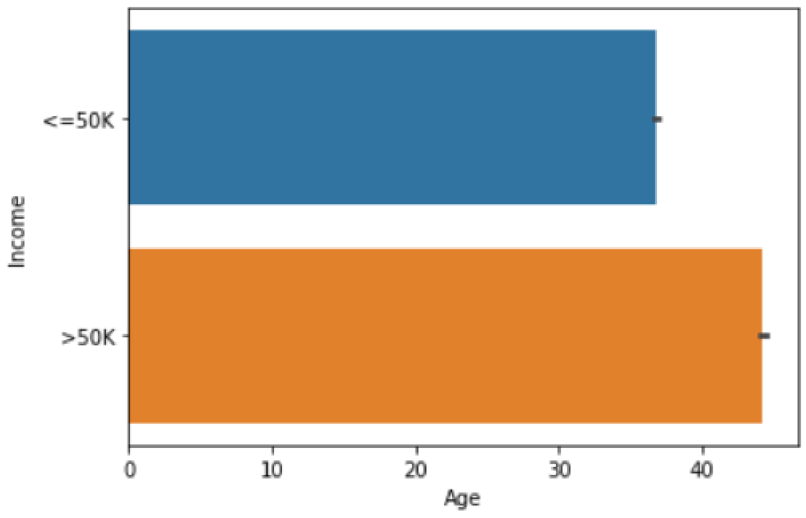
```
<AxesSubplot:xlabel='count', ylabel='Native_country'>
```



Most of the data is taken from United-States.

```
sns.barplot(x='Age',y='Income',data=census)
```

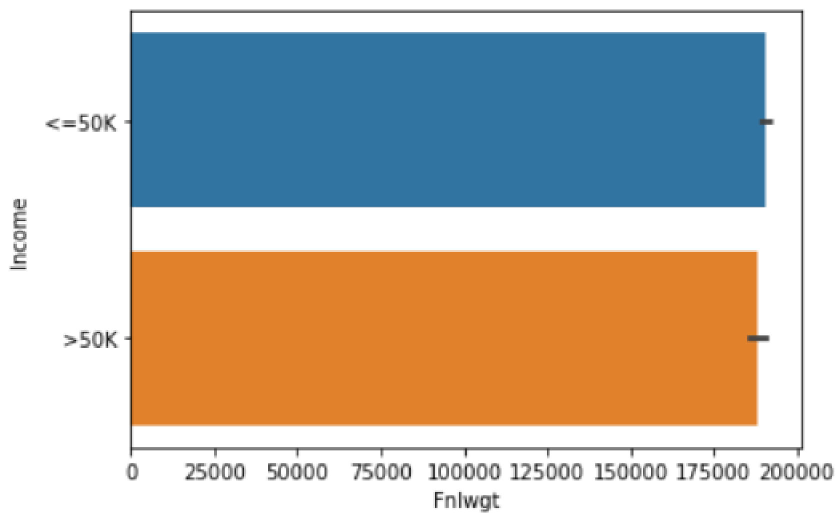
```
<AxesSubplot:xlabel='Age', ylabel='Income'>
```



People with age above 35 are more in number who makes income of more than $50K.

```
sns.barplot(x='Fnlwgt',y='Income',data=census)
```

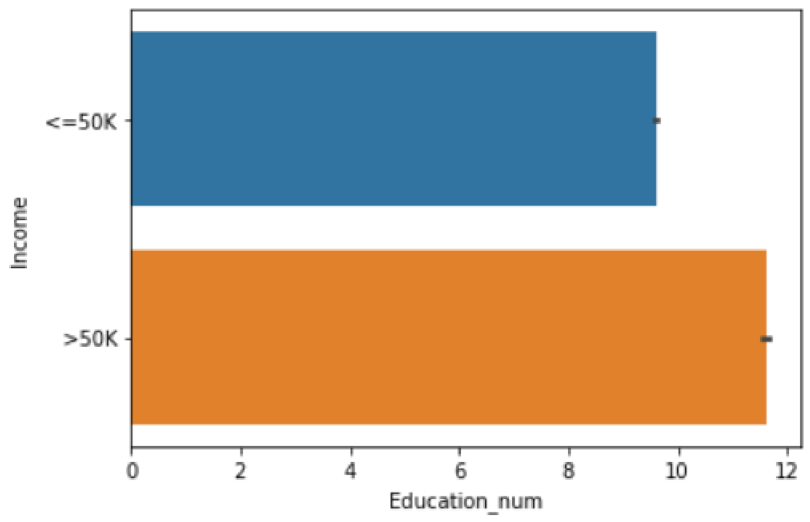
```
<AxesSubplot:xlabel='Fnlwgt', ylabel='Income'>
```



As fnlwgt is sampling weight we can see that data is almost equally distributed.

```
sns.barplot(x='Education_num',y='Income',data=census)
```

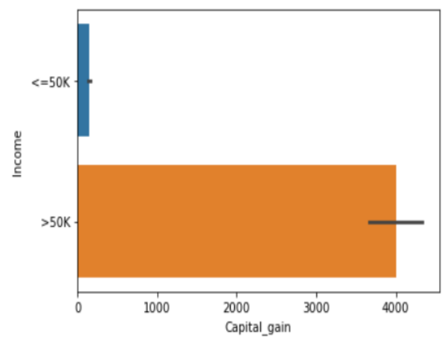
```
<AxesSubplot:xlabel='Education_num', ylabel='Income'>
```



People who studied for more than 9years are having more chances to make $50k per year.

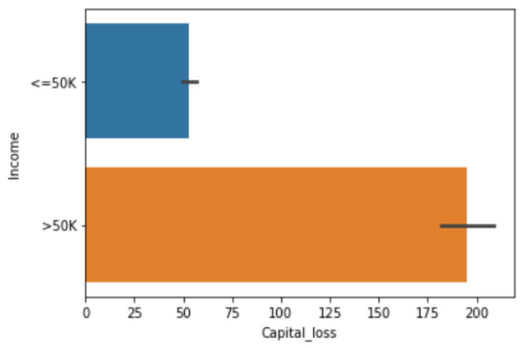
```
sns.barplot(x='Capital_gain',y='Income',data=census)
```

```
<AxesSubplot:xlabel='Capital_gain', ylabel='Income'>
```



```
sns.barplot(x='Capital_loss',y='Income',data=census)
```

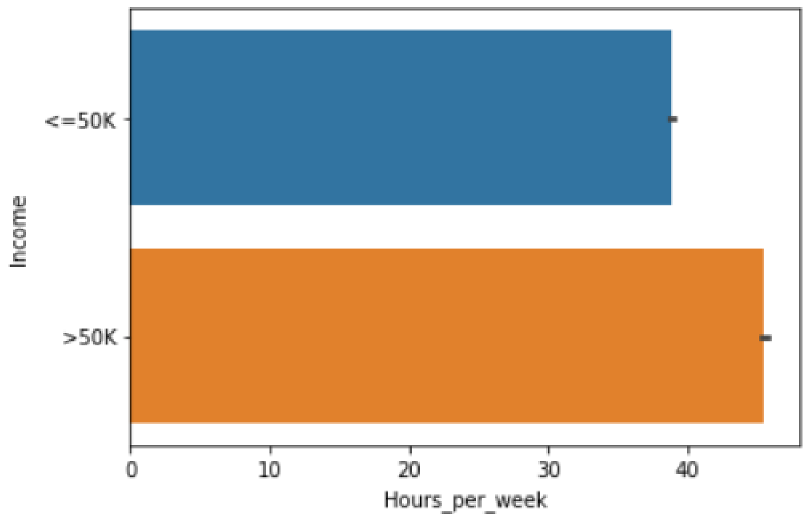
```
<AxesSubplot:xlabel='Capital_loss', ylabel='Income'>
```



We can see that people with income less than 50K are less in number in both capital_gain and capital_loss.

```
sns.barplot(x='Hours_per_week',y='Income',data=census)
```

```
<AxesSubplot:xlabel='Hours_per_week', ylabel='Income'>
```



People who work for more than 38 hours per week are having more chances to fall under category with income greater than $50k.

In the next step of preprocessing, we will convert all categorical columns to numerical columns using label encoder.

```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
census['Workclass'] = le.fit_transform(census['Workclass'])
census['Education'] = le.fit_transform(census['Education'])
census['Marital_status'] = le.fit_transform(census['Marital_status'])
census['Occupation'] = le.fit_transform(census['Occupation'])
census['Relationship'] = le.fit_transform(census['Relationship'])
census['Race'] = le.fit_transform(census['Race'])
census['Sex'] = le.fit_transform(census['Sex'])
census['Native_country'] = le.fit_transform(census['Native_country'])
census['Income'] = le.fit_transform(census['Income'])
```

We can see that education and education_num are having almost same data.

Education tells the highest education qualification a person is having.

Education_num gives the number of years a person spent on education, so we can drop one of them.

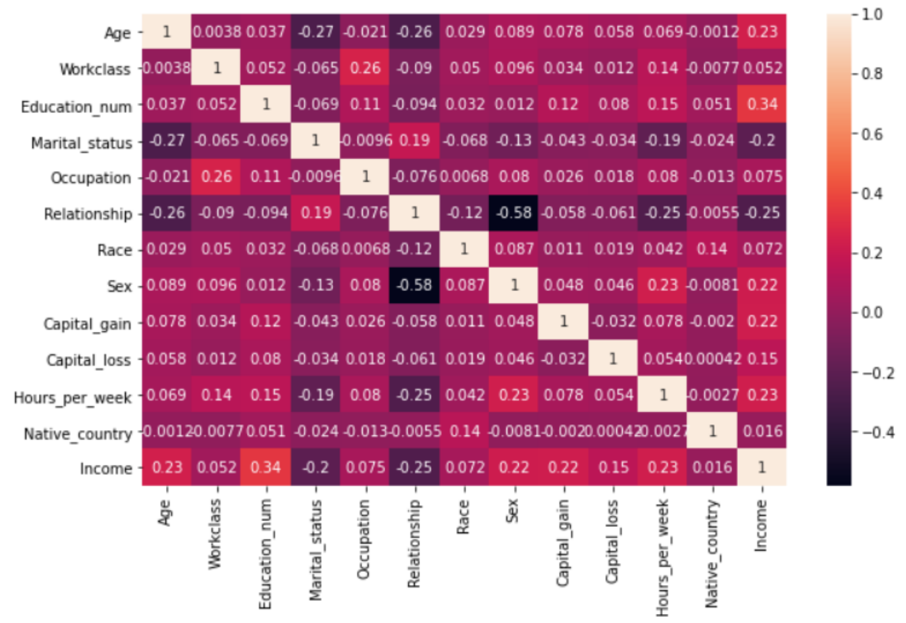
fnlwgt description tells us that it gives information about sampling weight, so it will not create impact on model building, so we can drop fnlwgt.

```python
census.drop('Education',inplace=True,axis=1)
census.drop('Fnlwgt',inplace=True,axis=1)
```

From correlation matrix, we can observe that education_num is having strong relation with income column.

```
corr_mat = census.corr()
```

```
plt.figure(figsize=[10,6])
sns.heatmap(corr_mat,annot=True)
plt.title = ('Correlation Matrix')
plt.show()
```



```
corr_target = census.corrwith(census['Income'], axis=0)
corr_target
```

```
Age               0.234039
Workclass         0.051645
Education_num     0.335182
Marital_status   -0.199295
Occupation        0.075448
Relationship     -0.250924
Race              0.071853
Sex               0.215995
Capital_gain      0.223333
Capital_loss      0.150523
Hours_per_week    0.229690
Native_country    0.015845
Income            1.000000
dtype: float64
```

Every column is having some relation with target column after removing those two columns.

```
census.plot(kind='density',subplots=True, sharex=False, legend=False, fontsize=1, figsize=[4,20])
plt.show()
```

```
sns.pairplot(census)
plt.show()
```

Density plot and pairplot tell us that there is some skewness in few parts of data, so we use power transform to remove skewness to some extent.

```
x = census.drop('Income',axis=1)
y = census['Income']
```

```
x.skew()
```

```
Age                0.558738
Workclass         -0.752280
Education_num     -0.311630
Marital_status    -0.013448
Occupation         0.114540
Relationship       0.786784
Race              -2.435332
Sex               -0.719244
Capital_gain      11.953690
Capital_loss       4.594549
Hours_per_week     0.227636
Native_country    -3.658235
dtype: float64
```

```
from sklearn.preprocessing import power_transform
census_new = power_transform(x)
census_new = pd.DataFrame(census_new,columns = x.columns)
```

```
census_new.skew()
```

```
Age               -0.013897
Workclass          0.061629
Education_num      0.023885
Marital_status    -0.114201
Occupation        -0.187152
Relationship       0.122917
Race              -2.010817
Sex               -0.719244
Capital_gain       3.016951
Capital_loss       4.299511
Hours_per_week     0.229556
Native_country    -2.725467
dtype: float64
```

Till now we have completed all the data preprocessing steps that are required to filter our data.

We started with checking for null values, proceeded with data visualization and concluded by removing skewness.

Model Building:

Now it's time to start our model building process.

We will start by separating target column from our dataset.

```
x = census.drop('Income',axis=1)
y = census['Income']
```

Now we will split our data into training and test data.

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25, random_state=15)
```

We are now ready to start our model building.

First we will use Logistic Regression Model,

Logistic Regression Model:

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative distribution function of logistic distribution.

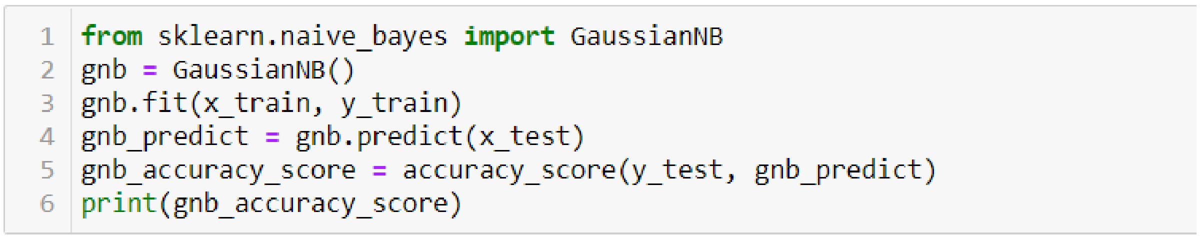
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
lr_predict =lr.predict(x_test)
lr_accuracy = accuracy_score(y_test, lr_predict)
print(lr_accuracy)
```

```
0.8141277641277641
```

We will use accuracy score to check the accuracy of our model.

Gaussian Naive Bayes Classifier:

The Gaussian Processes Classifier is a classification machine learning algorithm. Gaussian Processes are a generalization of the Gaussian probability distribution and can be used as the basis for sophisticated non-parametric machine learning algorithms for classification and regression.

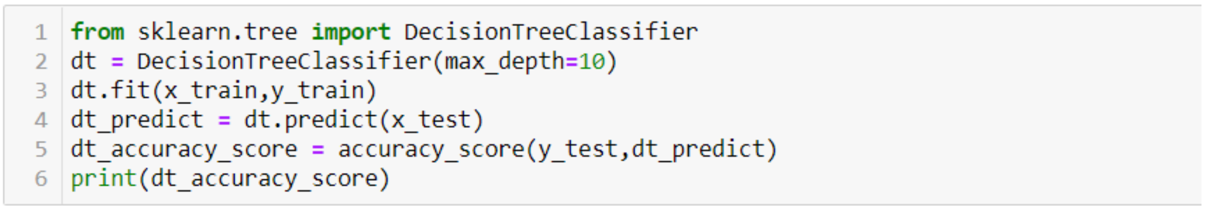


```python
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train, y_train)
gnb_predict = gnb.predict(x_test)
gnb_accuracy_score = accuracy_score(y_test, gnb_predict)
print(gnb_accuracy_score)
```

```
0.8045454545454546
```

Decision Tree Classifier:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

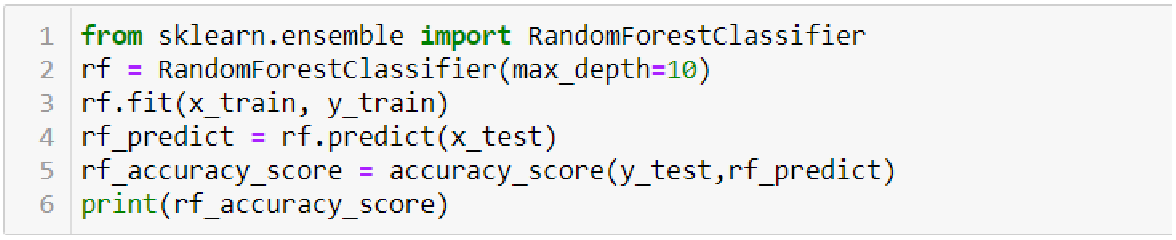


```python
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth=10)
dt.fit(x_train,y_train)
dt_predict = dt.predict(x_test)
dt_accuracy_score = accuracy_score(y_test,dt_predict)
print(dt_accuracy_score)
```

```
0.8606879606879607
```

Random Forest Classifier:

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.



```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=10)
rf.fit(x_train, y_train)
rf_predict = rf.predict(x_test)
rf_accuracy_score = accuracy_score(y_test,rf_predict)
print(rf_accuracy_score)
```

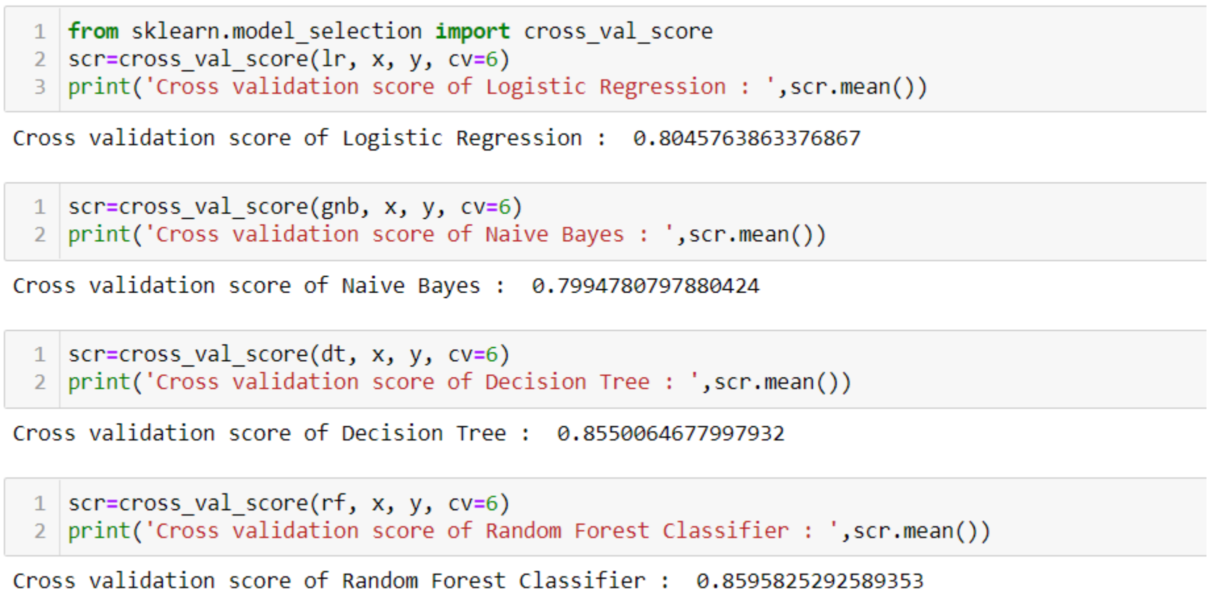
```
0.8614250614250615
```

By building our model using various algorithms we can select the best model which will give predictions with good accuracy, but by looking at the accuracy score we can’t conclude that which is the best model.

We need to perform cross validation to check the best model among all the models built.

Cross Validation:

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).



```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(lr, x, y, cv=6)
print('Cross validation score of Logistic Regression : ',scr.mean())
```

```
Cross validation score of Logistic Regression :  0.8045763863376867
```

```
scr=cross_val_score(gnb, x, y, cv=6)
print('Cross validation score of Naive Bayes : ',scr.mean())
```

```
Cross validation score of Naive Bayes :  0.7994780797880424
```

```
scr=cross_val_score(dt, x, y, cv=6)
print('Cross validation score of Decision Tree : ',scr.mean())
```

```
Cross validation score of Decision Tree :  0.8550064677997932
```

```
scr=cross_val_score(rf, x, y, cv=6)
print('Cross validation score of Random Forest Classifier : ',scr.mean())
```

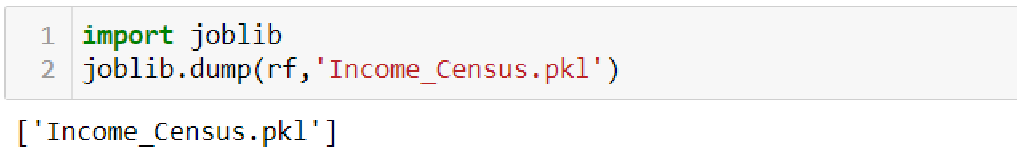
```
Cross validation score of Random Forest Classifier :  0.8595825292589353
```

From above cross validation, we can observe that Random Forest Classifier is having least difference between accuracy score and cross validation.

So, Random Forest Classifier with accuracy score of 86.14% is the best model

Deployment:

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data.



```python
import joblib
joblib.dump(rf,'Income_Census.pkl')
```

```
['Income_Census.pkl']
```

Conclusion:

As part of our study, we understood how to build a proper machine learning model and what are the points that are to be considered and systematic workflow.

I will be happy to take any kind of suggestions for the betterment of project.

Hope, you liked my work and thanks for reading