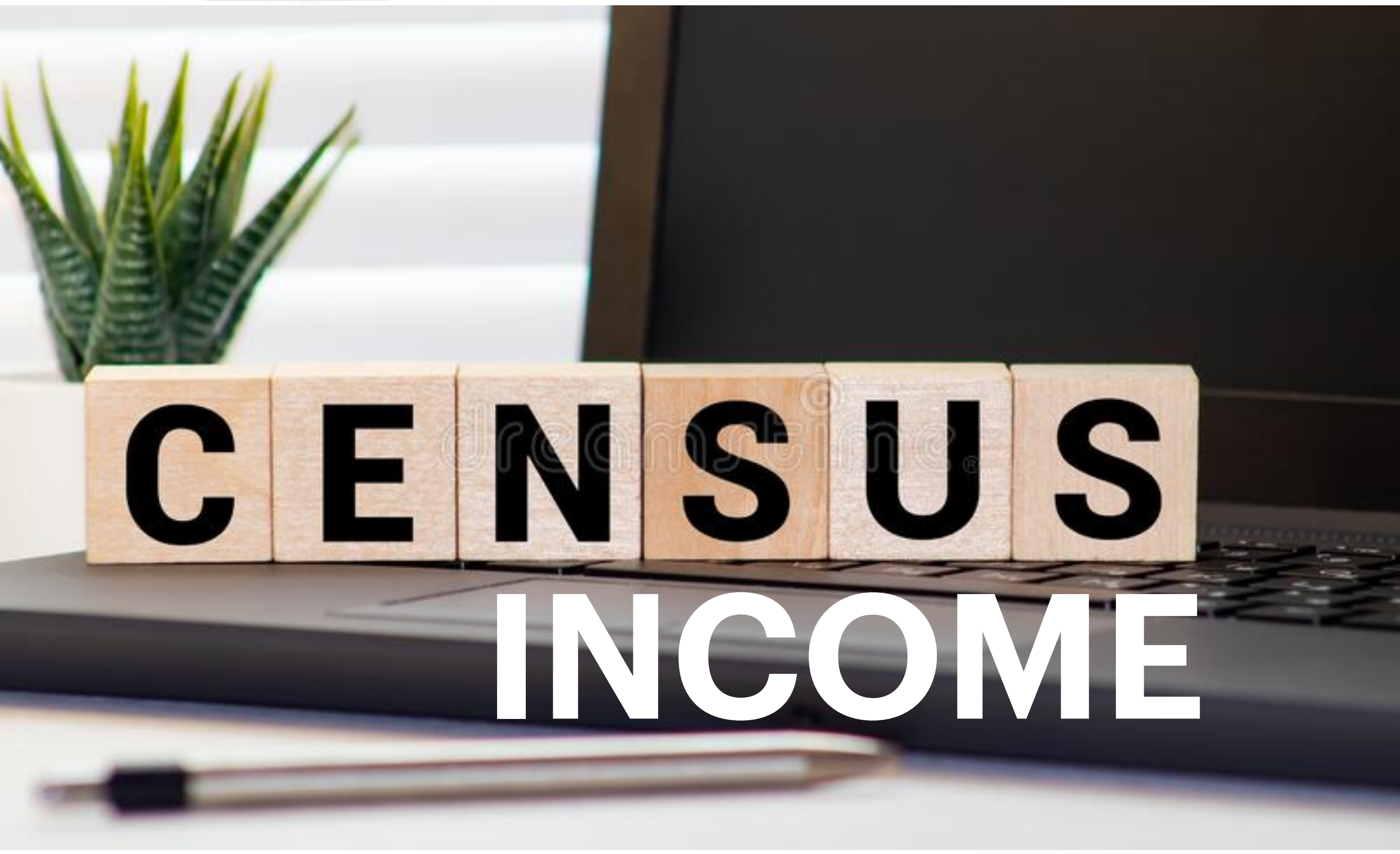
MACHINE LEARNING



Data Source : The Census Income data was extracted from the 1994 Census Bureau Database.

Credits : Rinny Kohavi and Barry Becker

DATASET DESCRIPTION :

The dataset is a collection of multiple samples from country USA , containing different sets of informations about people and their respective age, professional ,educational informations.All these informations altogether contribute to an individual's income.

With these features we need to prepare a machine learning model that predicts an individual's income and classifies them into two categories ,either an individual is earning '>50K' or '<=50K'.

Attribute Information :

- 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, 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.

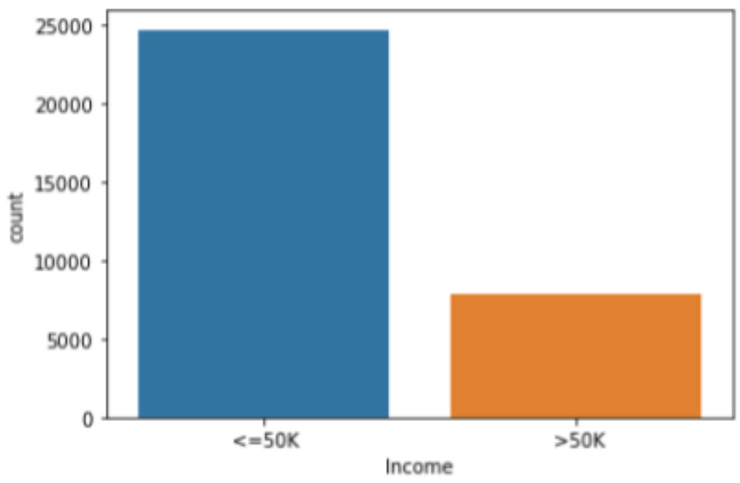
- native-country : United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran,Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico,Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

LIBRARIES USED :

- PANDAS
- NUMPY
- SEABORN
- MATPLOTLIB
- SCIPY

The Census Income dataset is a **Supervised Binary Classification** type of machine learning problem.

- The dataframe has 32560 rows and 15 columns originally.
- The target variable 'Income' has 2 types of results as '<=50K' or '>50K'.
- People with Income '<=50K' are 24719.
- People with income '>50K' are 7841.

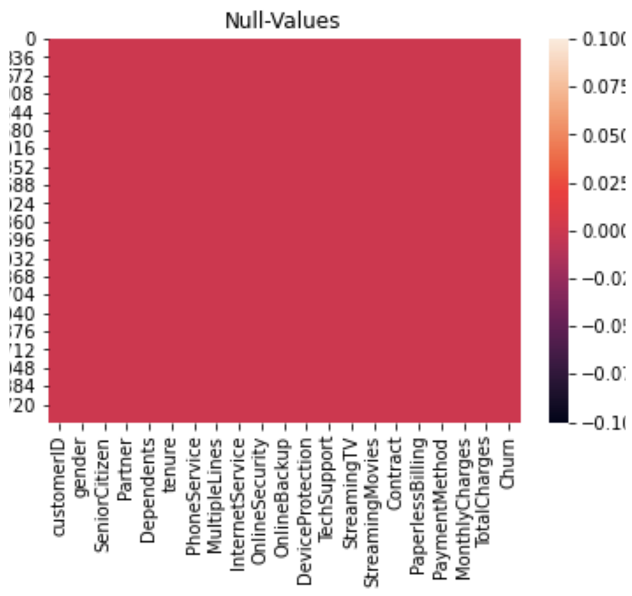


EXPLORING & CLEANING OF DATA :

DATAFRAME-

- duplicate values = '0'

- mis-interpreted values = ‘?'
- null-values =‘0’



When exploring the dataset and it’s values , there were no null-values or duplicate values found iN the graph or record.

Further, on exploring the value counts to get idea about column’s classifications I came across the value ‘?’.

There were 3 columns with mis-interpreted values.

- Workclass - ‘?’
- Occupation - ‘?’
- Native_country - ‘?’

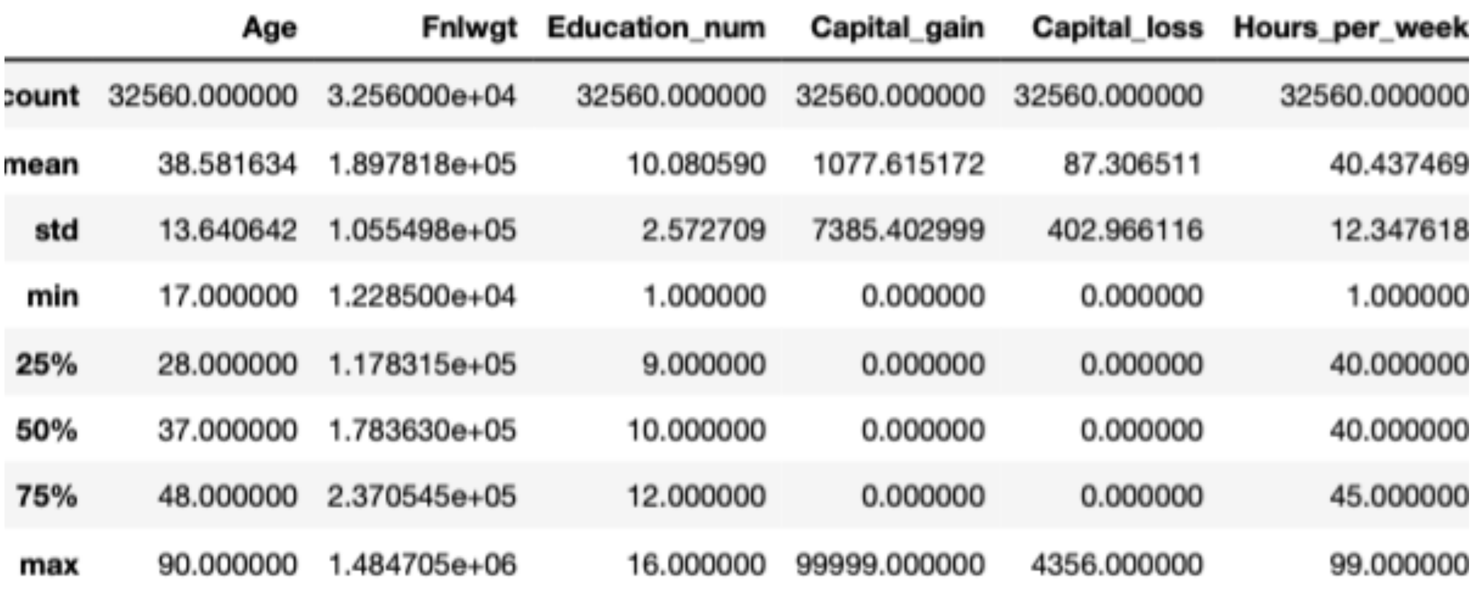
TREATING THE MIS-INTERPRETED VALUES :

The columns containing ‘?’ as values are non-numerical value containing features , So I replaced the string value ‘?’ with the mode i.e most frequent values of the respective features.

- Workclass - mode < Private >
- Occupation - mode < Prof-specialty >
- Native_country - mode < United-States >

DATA DESCRIPTION :

Finding out the total count ,mean , standard deviation, quarter percentiles , minimum and maximum values of the continuous type numerical data columns.



	Age	Fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week
count	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581634	1.897818e+05	10.080590	1077.615172	87.306511	40.437469
std	13.640642	1.055498e+05	2.572709	7385.402999	402.966116	12.347618
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

EXPLORATORY DATA ANALYSIS

It is important for us to have insights about data .It helps us find informations which are hidden in the data .By presenting the data features on graph we can observe & draw certain conclusions. For the graph sketching we use libraries 'Seaborn' and 'Matplotlib'.

Visualization

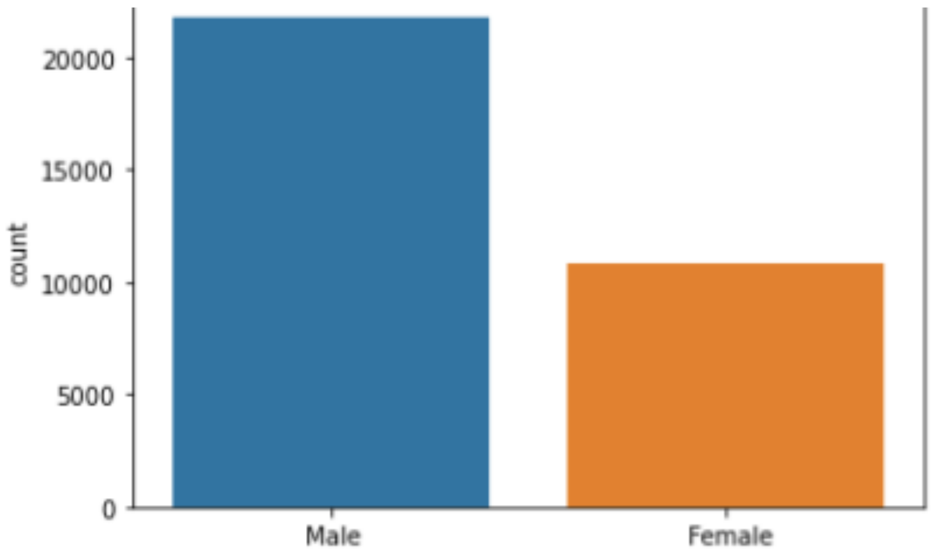
The dataframe includes both categorical and numerical data.

Uni-Variate Analysis :

- On categorical data we perform uni-variate analysis , which gives information about certain category which is popularly occupied and least occupied category on the contrary.For example, drawning graph for the categorical variable 'Sex'.

Males - 21789

Females- 1077



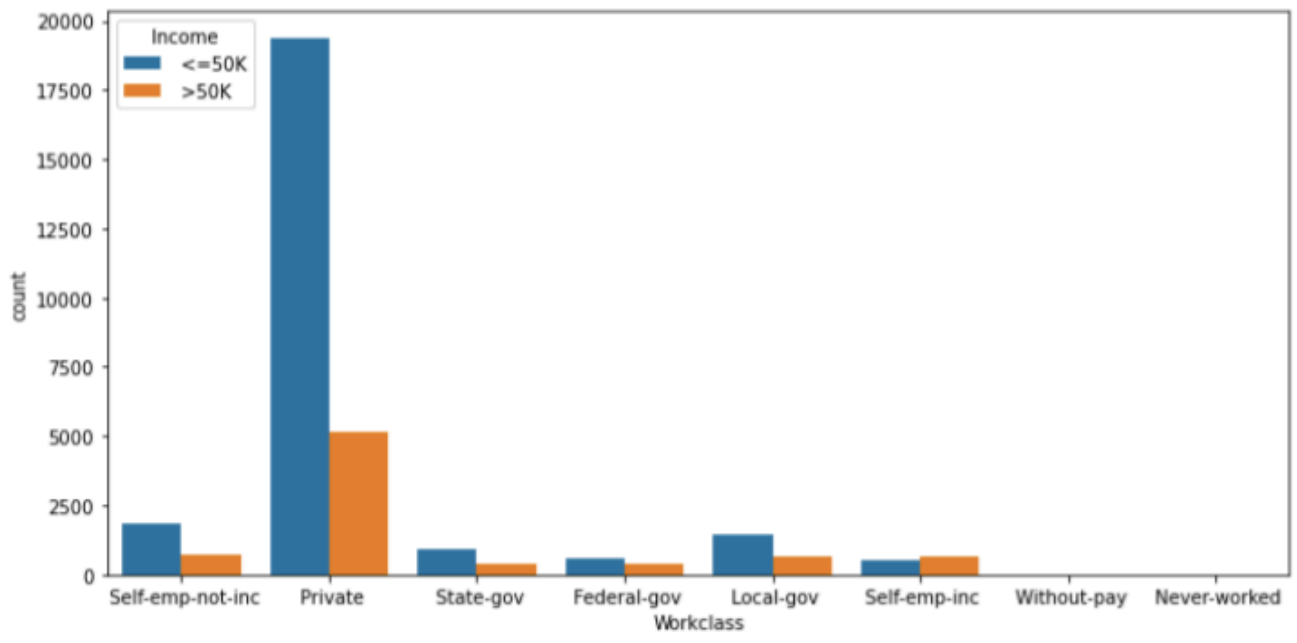
The graph
shows that no of males is higher than no of females in the record.

Bi-Variate Analysis :

In bi-variate analysis , we draw graphs of different features with the target variable to understand which category in the feature columns impact the classification of the result target variable.

Workclass vs Income

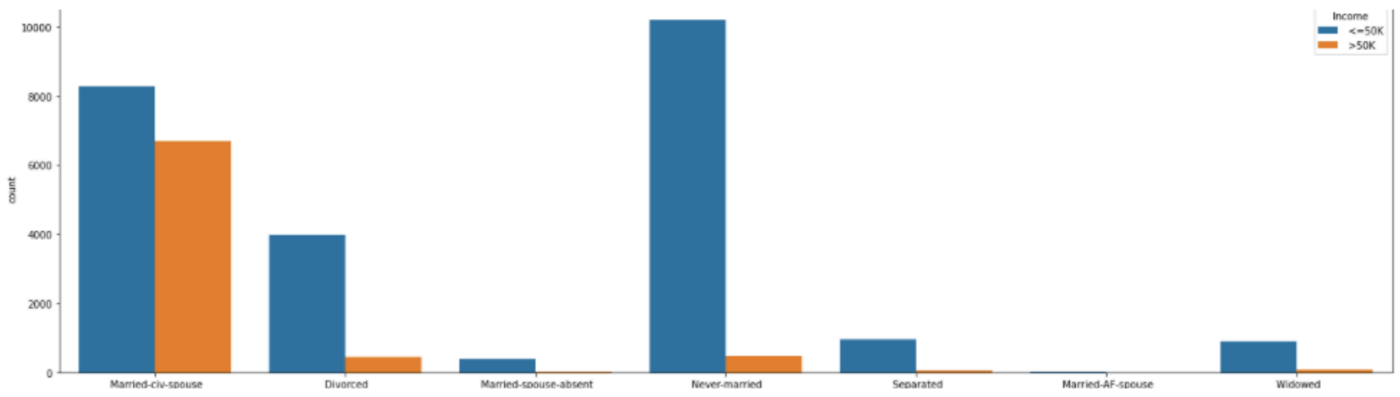
- The graph shows that large part of record, people working under 'Private' workclass are having their Income as <=50K. This suggest that into the target



variable more than half of the population having Income <=50K are from 'Private' workclass.

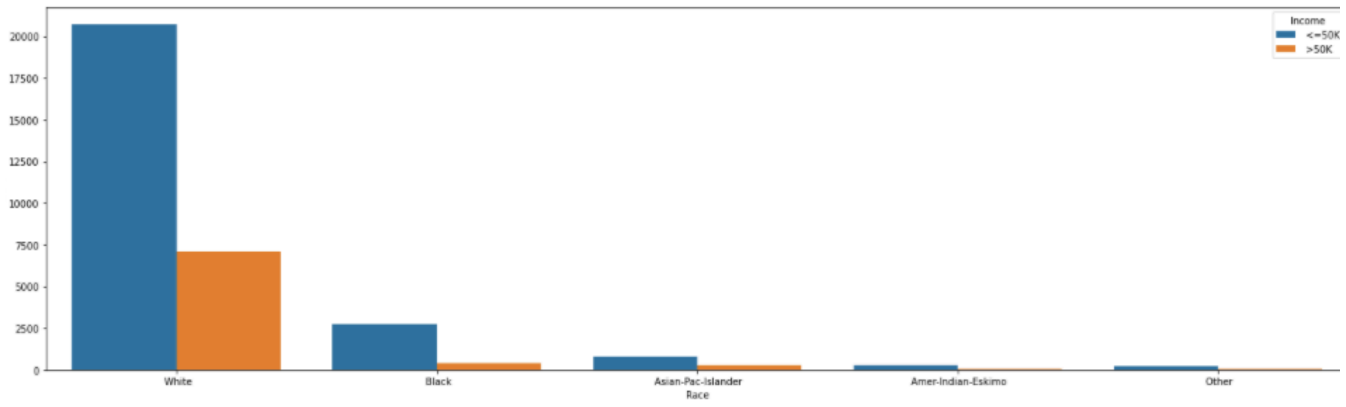
Marital_status vs Income

- Most of the People who are not in family are having salary below 50 K.



- People having Income >50K are mostly having relationship as husband or wife.

Race vs Income



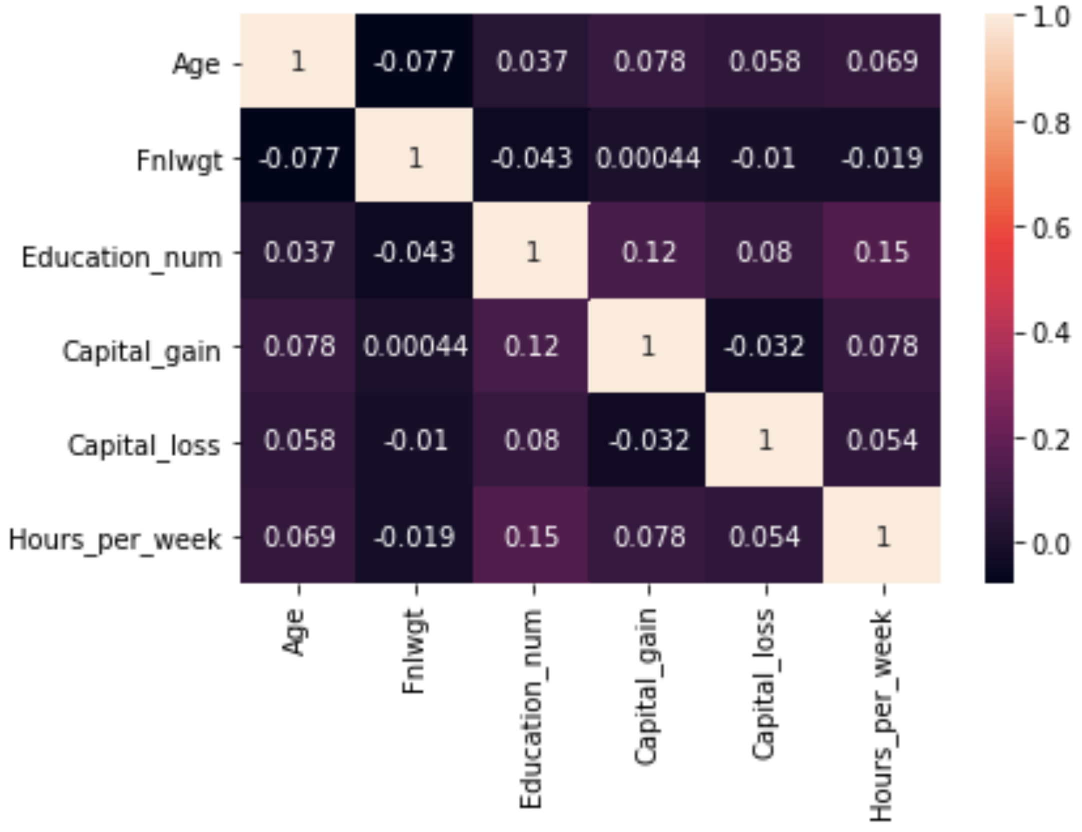
- The record has White-race dominating any other races.

- No of people having Income less than or more than 50K are both higher in white community.

Multi-Variate Analysis :

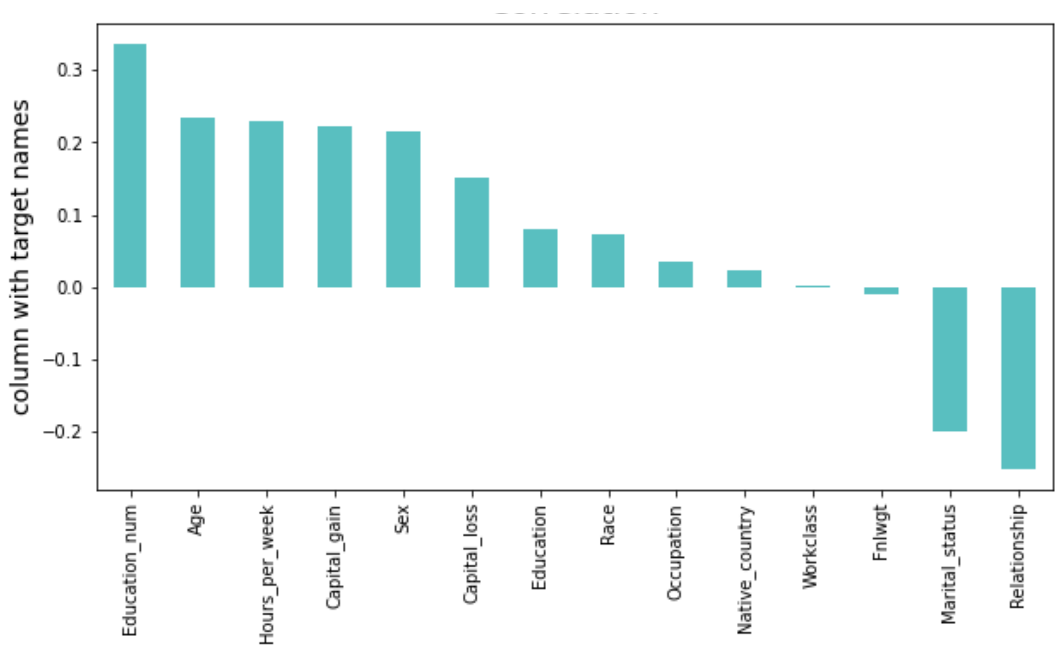
Multi-variate analysis is done on the dataframe with respect to the target variable.It gives us the correlation and other informations of all features with target variable in single picture.

Correlation Graph :



- The bright shades indicate positive correlation.
- The dark shades indicate negative correlation.
- Values written in the check-coloured boxes are the correlation coefficient values of respective columns.

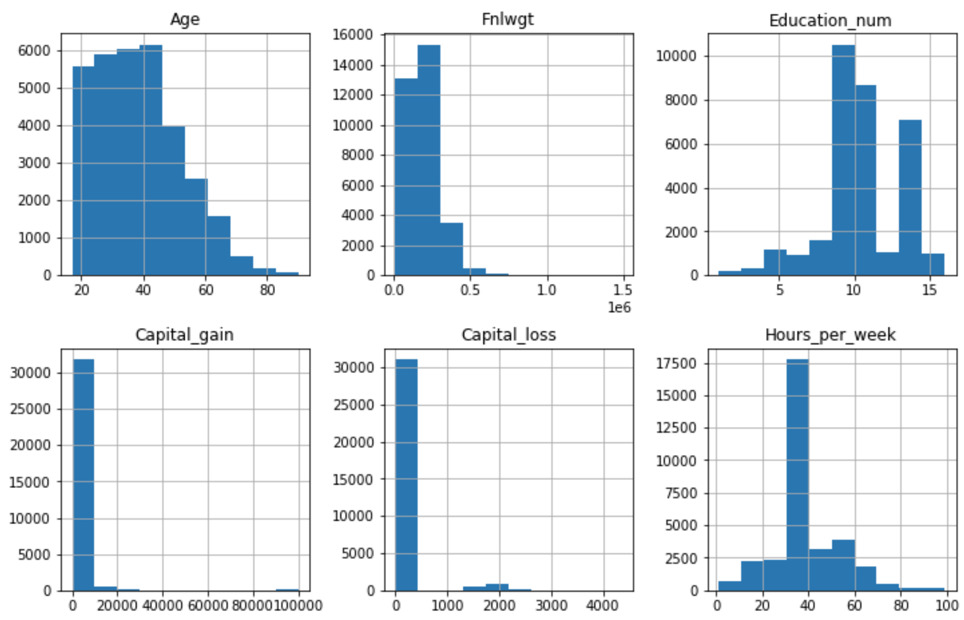
- Not a very strong correlation is observed among variables , which suggests we should look more deep into the data and find if we can drop some columns , making the training data more precise for more precision in testing results.



- Marital_status and Relationship are negatively related with target variable

BAR-PLOT OF DATAFRAME :

Bar-plot for the numeric continuous data columns:

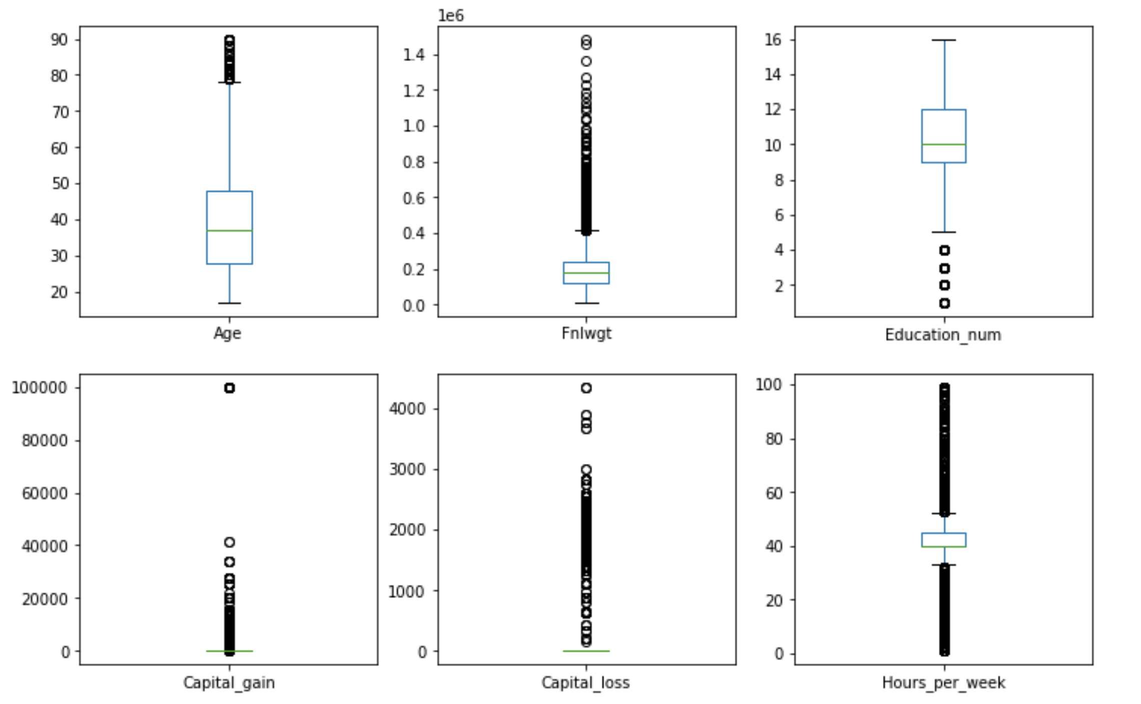


Caption

- The minimum age is 17 and the maximum is 90 years, most of the working age group lies between 20-40
- The minimum hours-per-week is 1 and maximum is 90, with most of the count lying between 30-40
- outliers observed in almost all the numeric features, these are the extreme values that are present in the data.

Box-Plot of dataframe :

The box plot graph shows the outliers present in the data , with 25th and 75th percentiles.

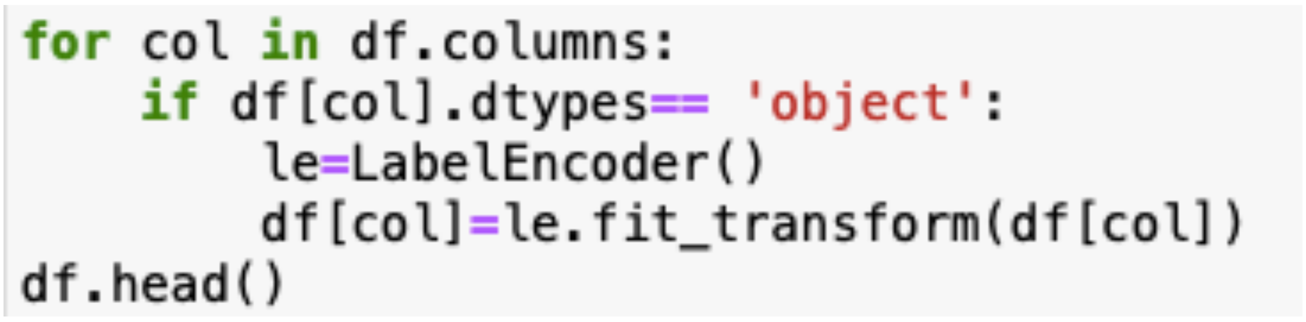


There are outliers present in all the continuous columns.

LABEL ENCODING :

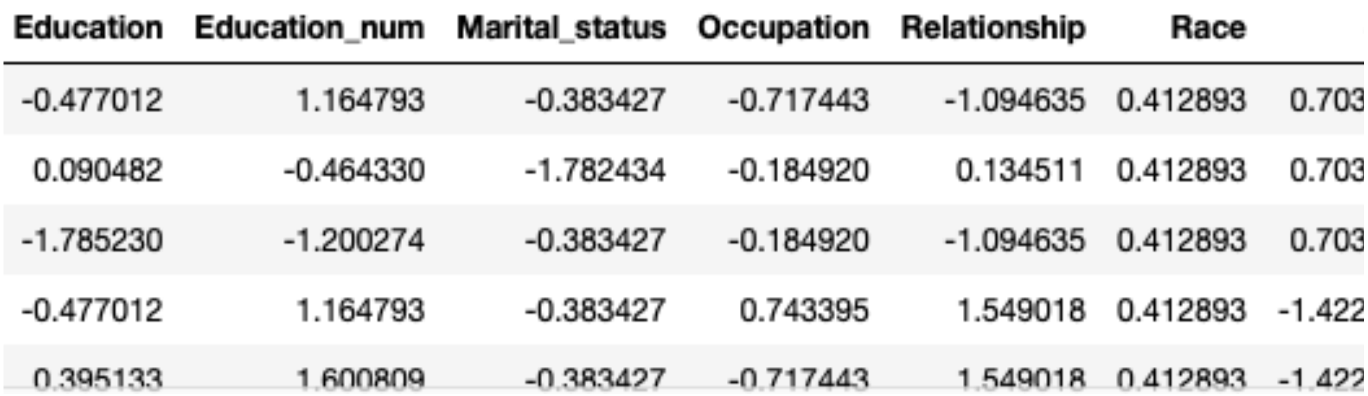
Thorough cleaning of the data has been done but to begin with our work with models training & testing part we need to pre-process the data in order to make it ready to pass through the model.We need to check the data types of values in different columns of the dataset and encode them into numerical format first.

Now, a loop has been introduced in which the dataframe will be passed and whichever column has data-type as ‘object-type’ it will be encoded into numeric format.



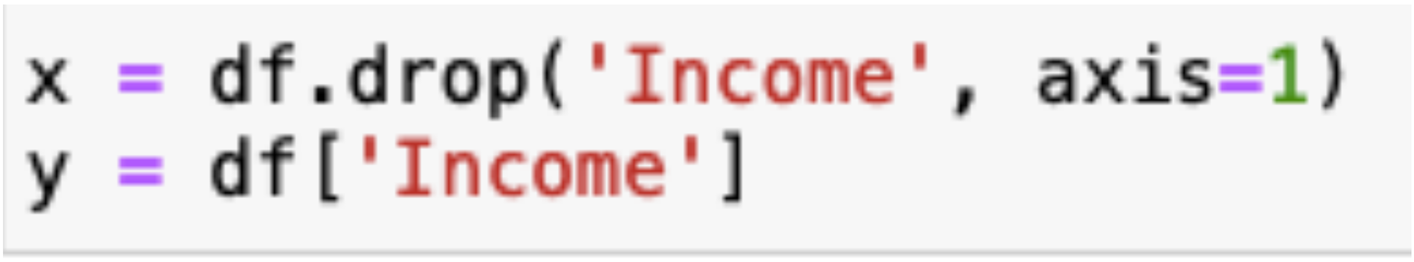
```python
for col in df.columns:
    if df[col].dtypes== 'object':
        le=LabelEncoder()
        df[col]=le.fit_transform(df[col])
df.head()
```

Sample of encoded dataframe :



Education	Education_num	Marital_status	Occupation	Relationship	Race	
-0.477012	1.164793	-0.383427	-0.717443	-1.094635	0.412893	0.703
0.090482	-0.464330	-1.782434	-0.184920	0.134511	0.412893	0.703
-1.785230	-1.200274	-0.383427	-0.184920	-1.094635	0.412893	0.703
-0.477012	1.164793	-0.383427	0.743395	1.549018	0.412893	-1.422
0.395133	1.600809	-0.383427	-0.717443	1.549018	0.412893	-1.422

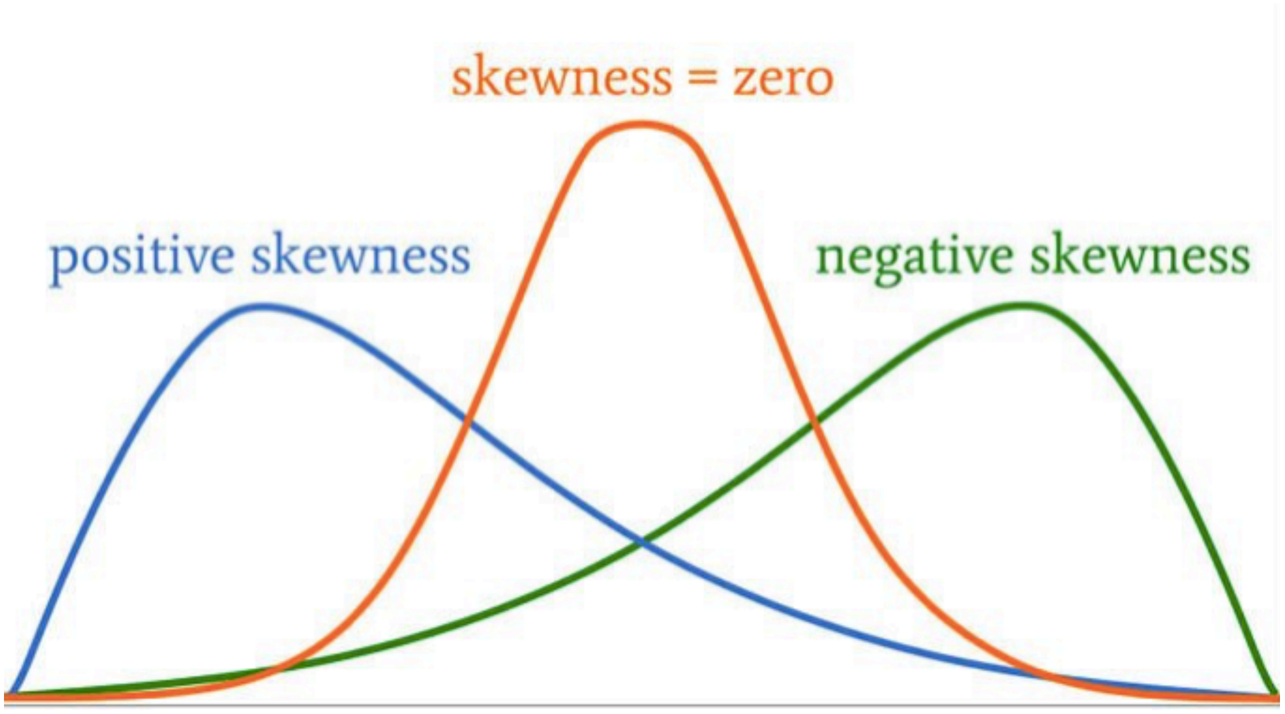
Separating the target variable from the other features.



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

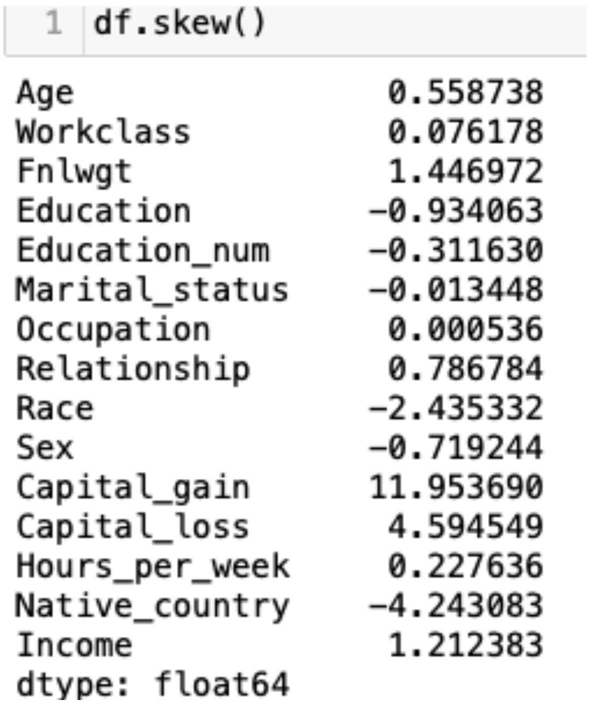
Checking the skewness :

Checking the skewness in the features to see if they all are either right-tailed or left-tailed skewness.



The skewness range is +/- 0.5.

The skewness of the original data lies above the range +/-0.5.

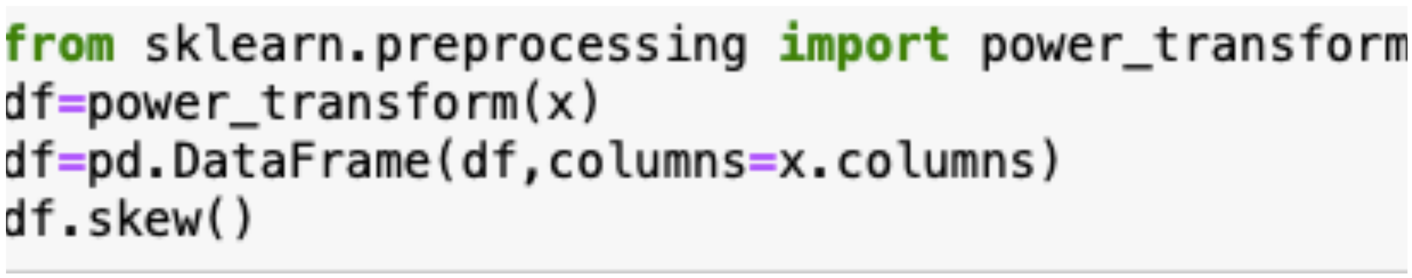


```
df.skew()
```

```
Age                0.558738
Workclass          0.076178
Fnlwgt             1.446972
Education         -0.934063
Education_num     -0.311630
Marital_status    -0.013448
Occupation         0.000536
Relationship       0.786784
Race              -2.435332
Sex               -0.719244
Capital_gain      11.953690
Capital_loss       4.594549
Hours_per_week     0.227636
Native_country    -4.243083
Income             1.212383
dtype: float64
```

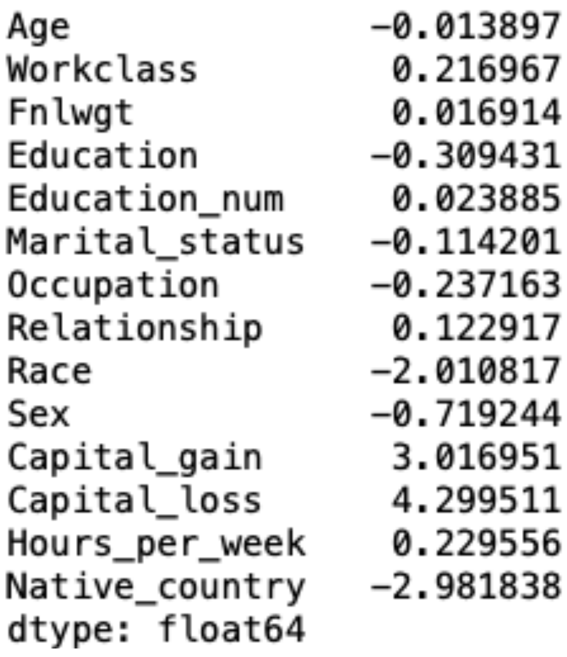
We will perform **Power Transformation** on the features to improve skewness.

Improved Skewness :



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

The improved skewness is aligned on same side tailed.

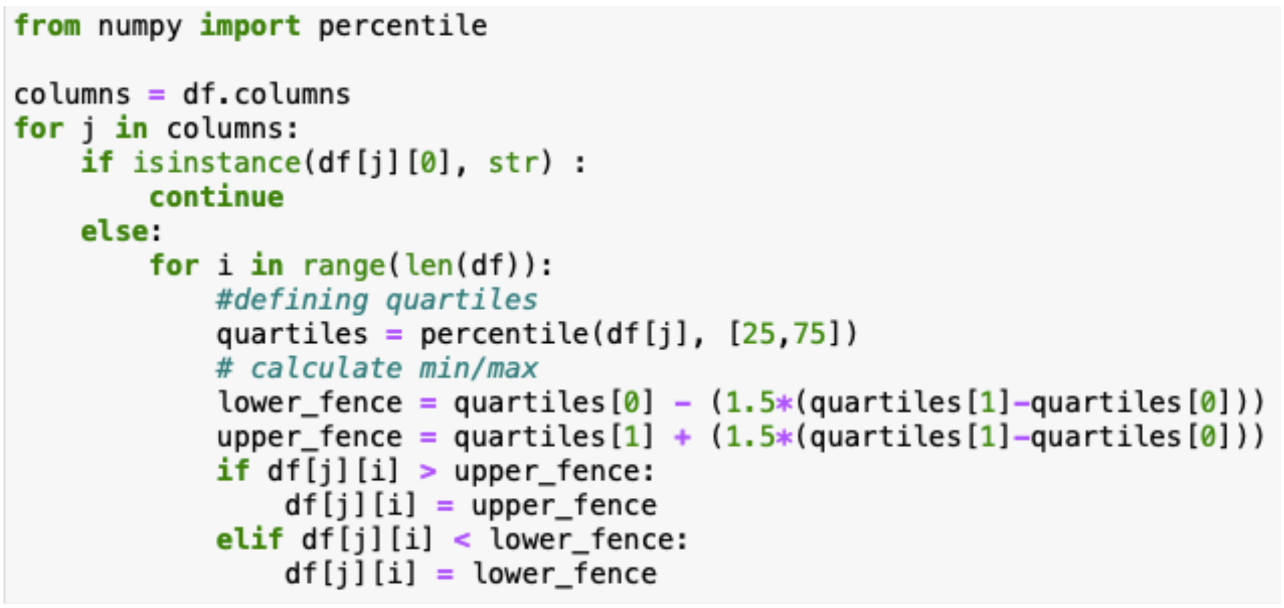


```
Age              -0.013897
Workclass         0.216967
Fnlwgt            0.016914
Education        -0.309431
Education_num     0.023885
Marital_status   -0.114201
Occupation       -0.237163
Relationship      0.122917
Race             -2.010817
Sex              -0.719244
Capital_gain      3.016951
Capital_loss      4.299511
Hours_per_week    0.229556
Native_country   -2.981838
dtype: float64
```

Handling the outliers :

Outliers are the extreme values present in the data , these values are mis-interpreted or have been put by mistake or wrong informations that may occur because human error in data collection or any other mistake.

The outliers impact the machine learning models , the model efficiency decreases as it learns wrong informations.Therefore, we need to treat them before passing them into training phase into the models.



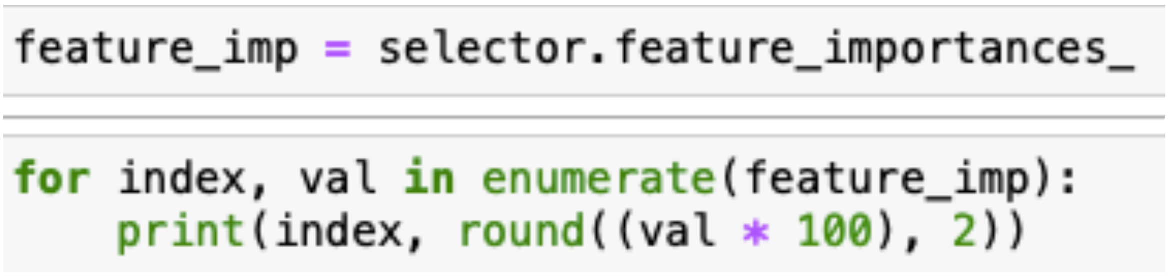
```python
from numpy import percentile

columns = df.columns
for j in columns:
    if isinstance(df[j][0], str) :
        continue
    else:
        for i in range(len(df)):
            #defining quartiles
            quartiles = percentile(df[j], [25,75])
            # calculate min/max
            lower_fence = quartiles[0] - (1.5*(quartiles[1]-quartiles[0]))
            upper_fence = quartiles[1] + (1.5*(quartiles[1]-quartiles[0]))
            if df[j][i] > upper_fence:
                df[j][i] = upper_fence
            elif df[j][i] < lower_fence:
                df[j][i] = lower_fence
```

The outliers can be removed by z-score method or percentile method.

I made a loop for percentile method and passed the dataframe into it , it results in refining the data with reduction of outliers.

Feature Selection :

Based on the scores of the **Extra Tree Classifier** for different attributes the most relevant features have been selected, that are going to be implemented in our model. As a result Features Workclass , Education , Race, Sex , Capital_loss and Native_country have been eliminated as they have the least Extra Trees Classifier Scores.

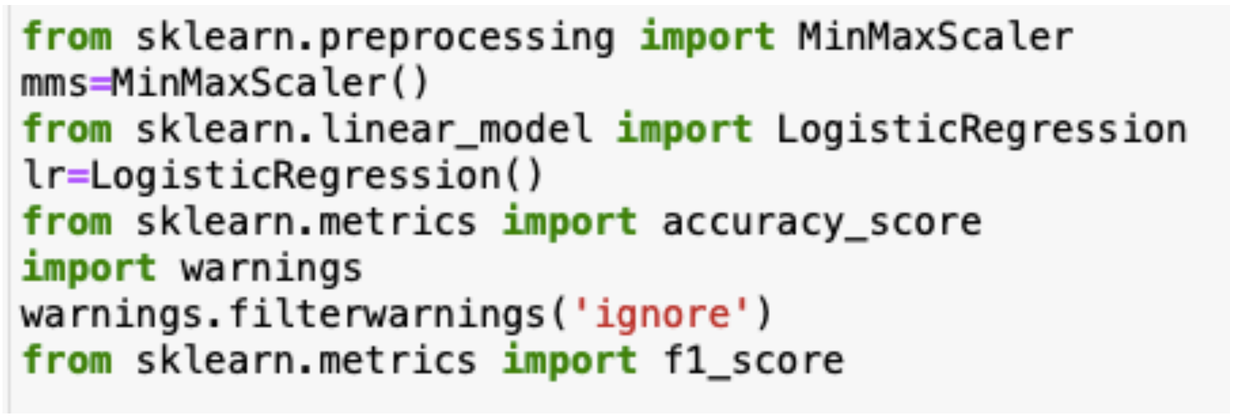


```
feature_imp = selector.feature_importances_
```

```
for index, val in enumerate(feature_imp):
    print(index, round((val * 100), 2))
```

SCALING & BALANCING VARIABLE :

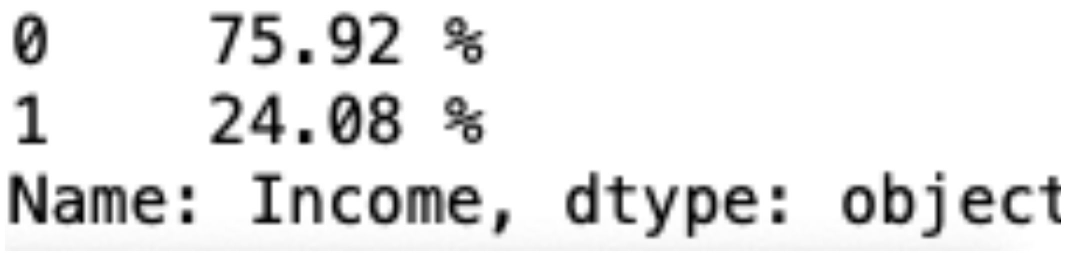
Scaling the data with the MinMax scaler to create uniformness in the data scale.



```python
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import f1_score
```

The target variable is imbalanced ,no of people with income ‘<=50K’ is higher than people with income ‘>50K’.

If we train our model with imbalanced data than the model might go biased results can be imperfect , precision will be compromised.



```
0     75.92 %
1     24.08 %
Name: Income, dtype: object
```

To overcome this bias issue of model we resample the training data.

```
import six
import sys
sys.modules['sklearn.externals.six'] = six

from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler


from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
```

The data is re-sampled using **Random Under Sampler** with **SMOTE**.

```
ros = RandomOverSampler(random_state=42)
```

```
ros.fit(x,y)
```

```
RandomOverSampler(random_state=42)
```

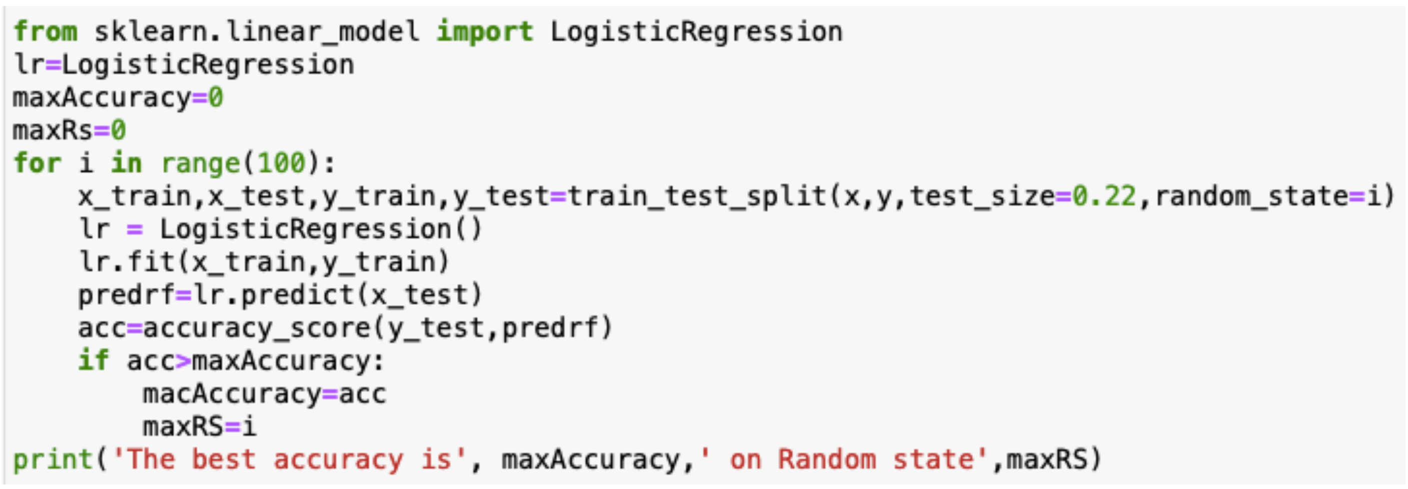
```
x_resampled,y_resampled=ros.fit_resample(x,y)
```

We can use this re-sampled data now to train our machine learning models .

Best Random State :

Before training the model we find a best random state where we get the best accuracy score.

Making a loop with **Logistic Regression** , in this we will pass the training data and print the best random state where accuracy score will be highest.

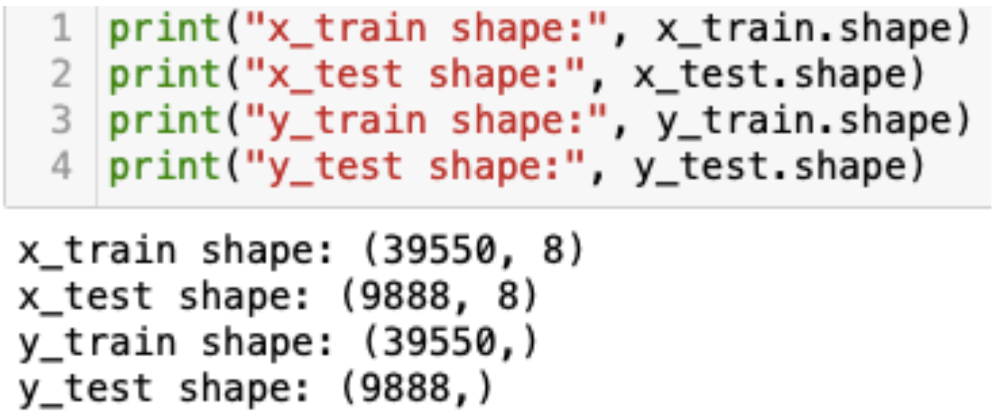


```python
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression
maxAccuracy=0
maxRs=0
for i in range(100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.22,random_state=i)
    lr = LogisticRegression()
    lr.fit(x_train,y_train)
    predrf=lr.predict(x_test)
    acc=accuracy_score(y_test,predrf)
    if acc>maxAccuracy:
        macAccuracy=acc
        maxRS=i
print('The best accuracy is', maxAccuracy,' on Random state',maxRS)
```

```
The best accuracy is 0 on Random state
99.
```

TRAIN , TEST & SPLIT :

Creating train ,test and split with random state 99 and test_size (0.2).



```
print("x_train shape:", x_train.shape)
print("x_test shape:", x_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
x_train shape: (39550, 8)
x_test shape: (9888, 8)
y_train shape: (39550,)
y_test shape: (9888,)
```

Checking the shapes of x_train ,x_test ,y_train and y_test.

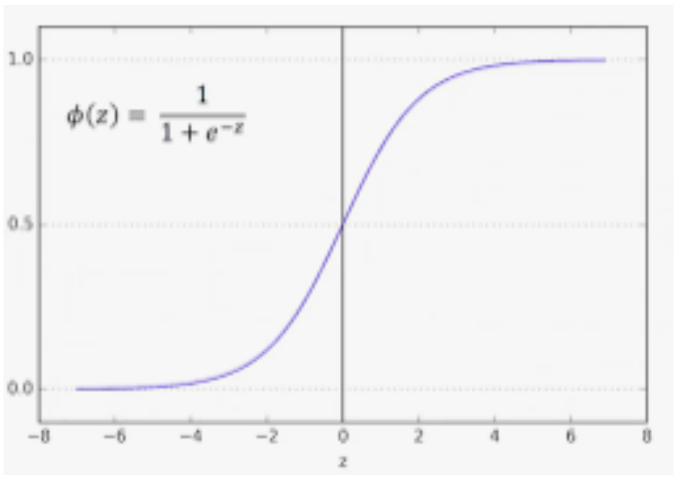
Testing Models :

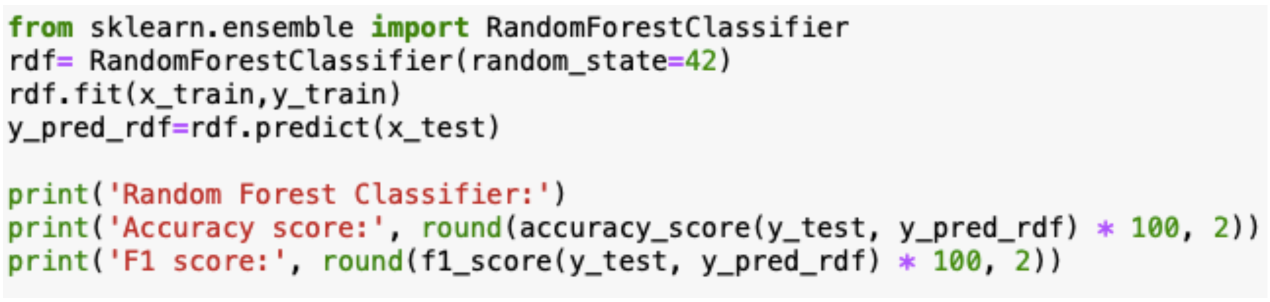
The data has been pre-processed and features have been selected , now we can train and test our models .

Logistic Regression Model :

After passing the data from the Logistic Regression model on random state 99.

Accuracy Score is 68.46.

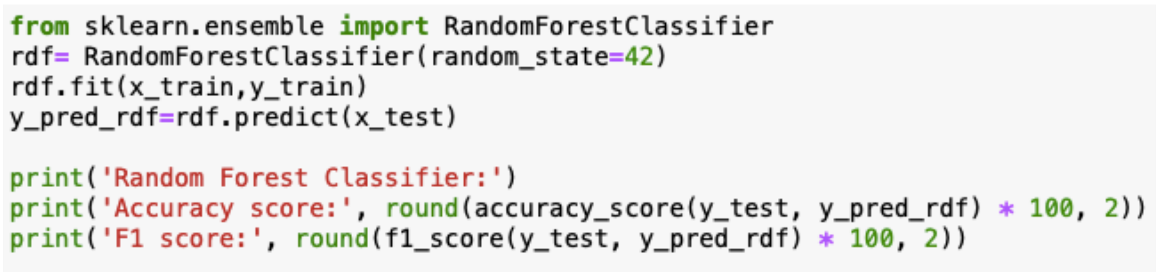




```
from sklearn.ensemble import RandomForestClassifier
rdf= RandomForestClassifier(random_state=42)
rdf.fit(x_train,y_train)
y_pred_rdf=rdf.predict(x_test)

print('Random Forest Classifier:')
print('Accuracy score:', round(accuracy_score(y_test, y_pred_rdf) * 100, 2))
print('F1 score:', round(f1_score(y_test, y_pred_rdf) * 100, 2))
```

Random Forest Classifier Model :

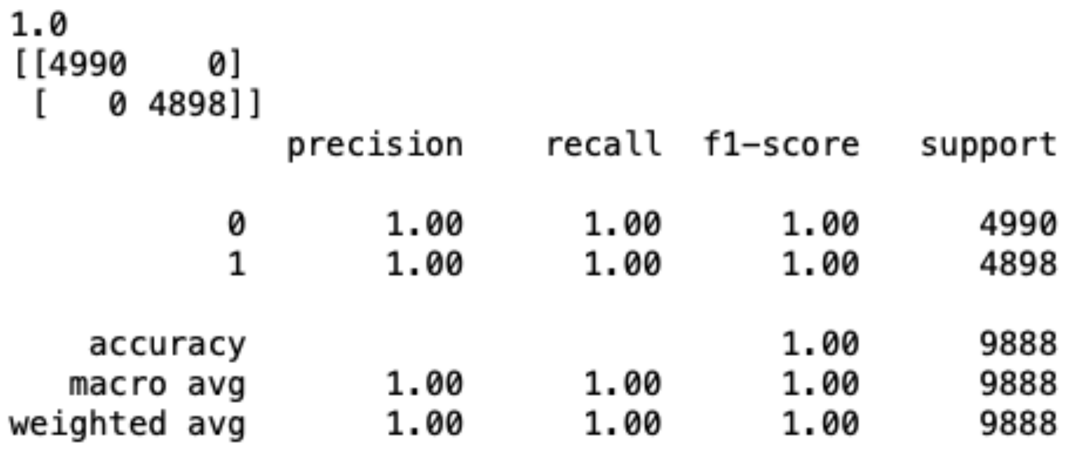
Accuracy score is 92.77.



```
from sklearn.ensemble import RandomForestClassifier
rdf= RandomForestClassifier(random_state=42)
rdf.fit(x_train,y_train)
y_pred_rdf=rdf.predict(x_test)

print('Random Forest Classifier:')
print('Accuracy score:', round(accuracy_score(y_test, y_pred_rdf) * 100, 2))
print('F1 score:', round(f1_score(y_test, y_pred_rdf) * 100, 2))
```

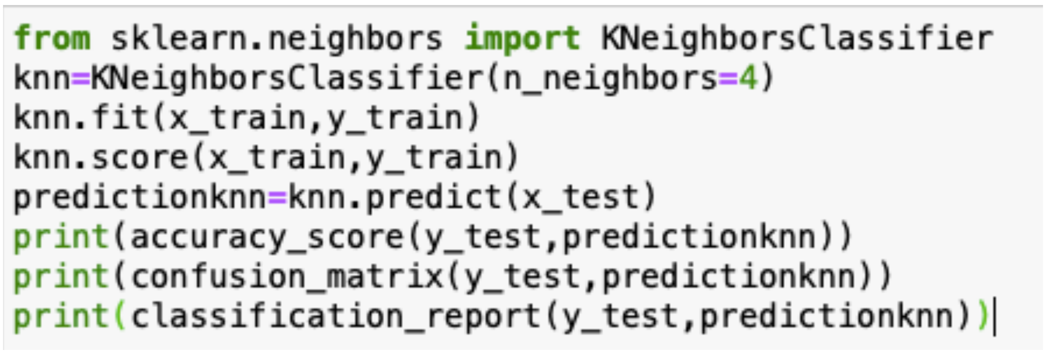
Decision Tree Classifier Model :

Accuracy score is 1.0.



```
1.0
[[4990    0]
 [   0 4898]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4990
           1       1.00      1.00      1.00      4898

    accuracy                           1.00      9888
   macro avg       1.00      1.00      1.00      9888
weighted avg       1.00      1.00      1.00      9888
```

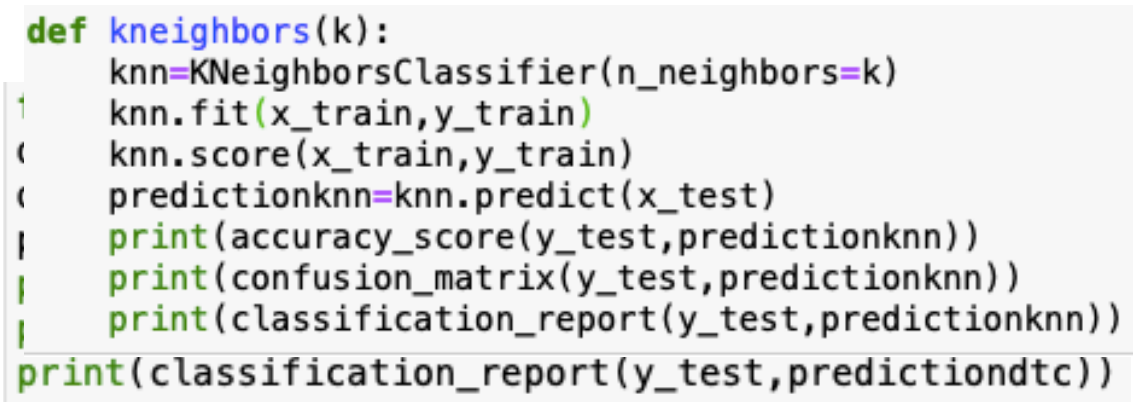
KNeighbors Classifier :



```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=4)
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
predictionknn=knn.predict(x_test)
print(accuracy_score(y_test,predictionknn))
print(confusion_matrix(y_test,predictionknn))
print(classification_report(y_test,predictionknn))
```

After passing the parameter of 4 values as neighbors for the model to obtain results.The accuracy score , confusion matrix and classification reports are as follows.-

The accuracy with 4 nearest neighbors is 0.75.Now ,we will a loop in which we will pass different parameter of nearest neighbor and check at which number the model gives highest accuracy.

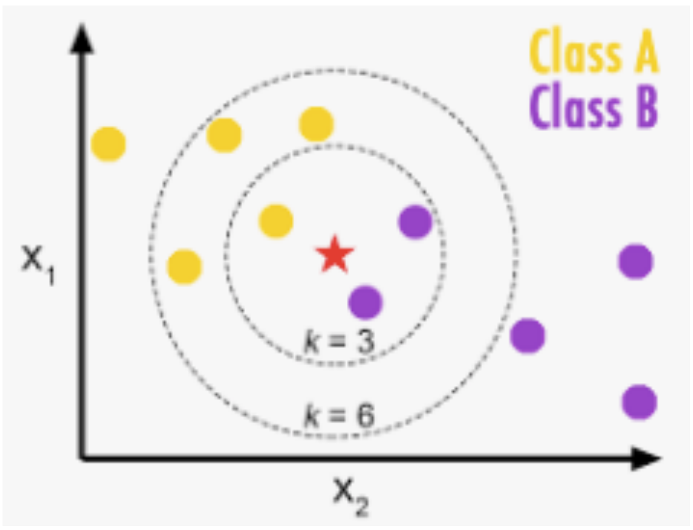


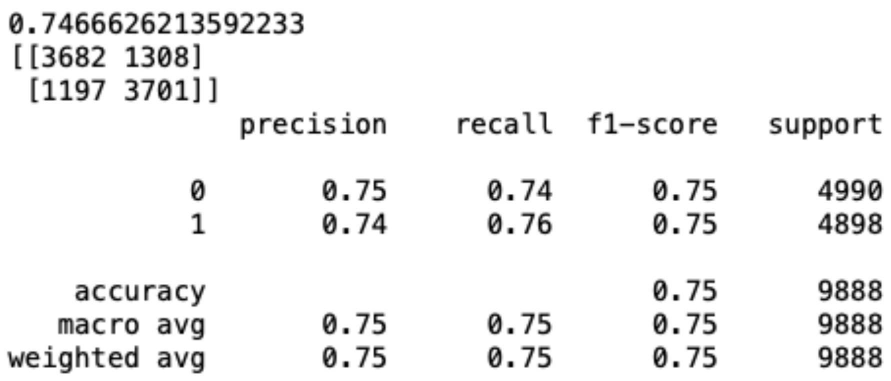
```
def kneighbors(k):
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train,y_train)
    knn.score(x_train,y_train)
    predictionknn=knn.predict(x_test)
    print(accuracy_score(y_test,predictionknn))
    print(confusion_matrix(y_test,predictionknn))
    print(classification_report(y_test,predictionknn))
print(classification_report(y_test,predictiondtc))
```

kneighbors(3) - Accuracy score is ‘0.76’.

kneighbors(5)- Accuracy score is ‘0.71’.

kneighbors(6)- Accuracy score is ‘o.70’.





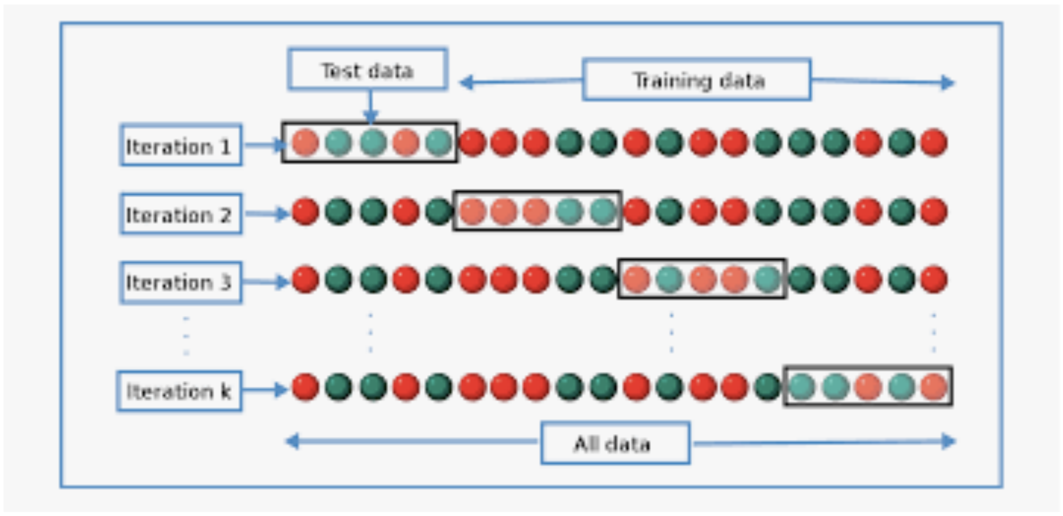
```
0.7466626213592233
[[3682 1308]
 [1197 3701]]
              precision    recall  f1-score   support

           0       0.75      0.74      0.75      4990
           1       0.74      0.76      0.75      4898

    accuracy                           0.75      9888
   macro avg       0.75      0.75      0.75      9888
weighted avg       0.75      0.75      0.75      9888
```

Observation - On passing different no as the neighbors parameter in the model there wasn’t a high difference of accuracy score observed.

We have tested the data on different classification models , but to come to a conclusion of best model we don’t consider the accuracy score parameter only. We will check the **K-fold** values so that if there has been any over-fitting or under-fitting biased pattern followed by the models.After this we can conclude the best model for our data.

Cross-Validation Score

Repeated k-fold cross-validation provides a way to improve the estimated performance of a machine learning model. ... This mean result is expected to be a more accurate estimate of the true unknown underlying mean performance of the model on the dataset, as calculated using the standard error.Hence,



we will find out the cross validation score for each tested model and compare them.

Models comparison :

MODEL NAME	ACCURACY SCORE	K-FOLD VALUE	DIFFERENCE
LOGISTIC REGRESSION	68.46	0.79	67.66
RANDOM FOREST CLASSIFIER	92.77	91.9	91.97
DECISION TREE CLASSIFIER	1.0	0.79	0.02
KNEIGHBORS CLASSIFIER	0.74	0.05	0.02

RESULT :

The minimum difference between the cross-validation score and accuracy score is for the **DecisionTree Classifier** model.

Cross-Validation score for each model :-

```python
from sklearn.model_selection import cross_val_score
```

Logistic Regression Model :-

```python
cvs=cross_val_score(dtc,x,y,cv=5)
print('The cross validation score for LinearRegression model is :',cvs.mean())
print('The difference between accuracy and crossvalidation score is :67.661812039312
```

```
The cross validation score for LinearRegression model is : 0.7978808353808353
The difference between accuracy and crossvalidation score is :67.66181203931204.
```

Decision Tree Classifier Model¶

```python
cvs=cross_val_score(dtc,x,y,cv=5)
print('The cross validation score for DecisionTreeClassifier model is :',cvs.mean())
print('The difference between accuracy and crossvalidation score is :0.200859950859
```

```
The cross validation score for DecisionTreeClassifier model is : 0.797972972972973
The difference between accuracy and crossvalidation score is :0.2008599508599509.
```

Random Forest Classifier Model :-

```python
cvs=cross_val_score(dtc,x,y,cv=5)
print('The cross validation score for DecisionTreeClassifier model is :',cvs.mean())
print('The difference between accuracy and crossvalidation score is :91.971289926289
```

```
The cross validation score for DecisionTreeClassifier model is : 0.797911547911548
The difference between accuracy and crossvalidation score is :91.97128992628993.
```

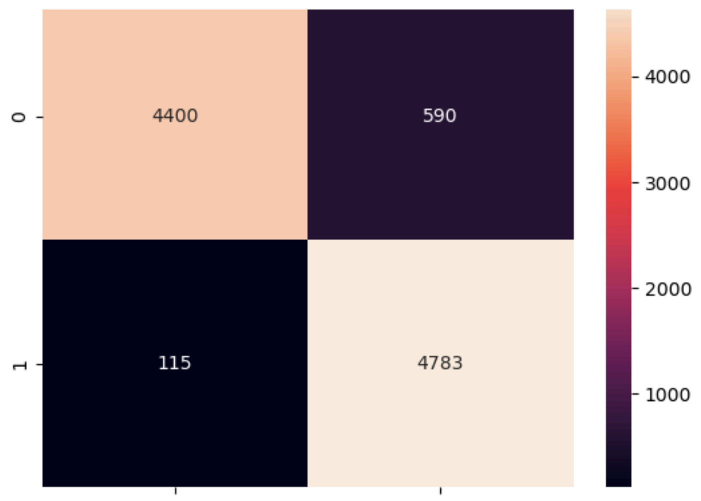
Nearest Neighbors Classifier Model : -

```python
cvs=cross_val_score(dtc,x,y,cv=5)
print('The cross validation score for Nearest Neighbors Classifier model is :',cvs.m
print('The difference between accuracy and crossvalidation score is :0.0536137914171
```

```
The cross validation score for Nearest Neighbors Classifier model is : 0.7994471744471
745
The difference between accuracy and crossvalidation score is :0.05361379141718936.
```

HyperParameter Tuning :

As we have selected **DecisionTree Classifier** model as the best model



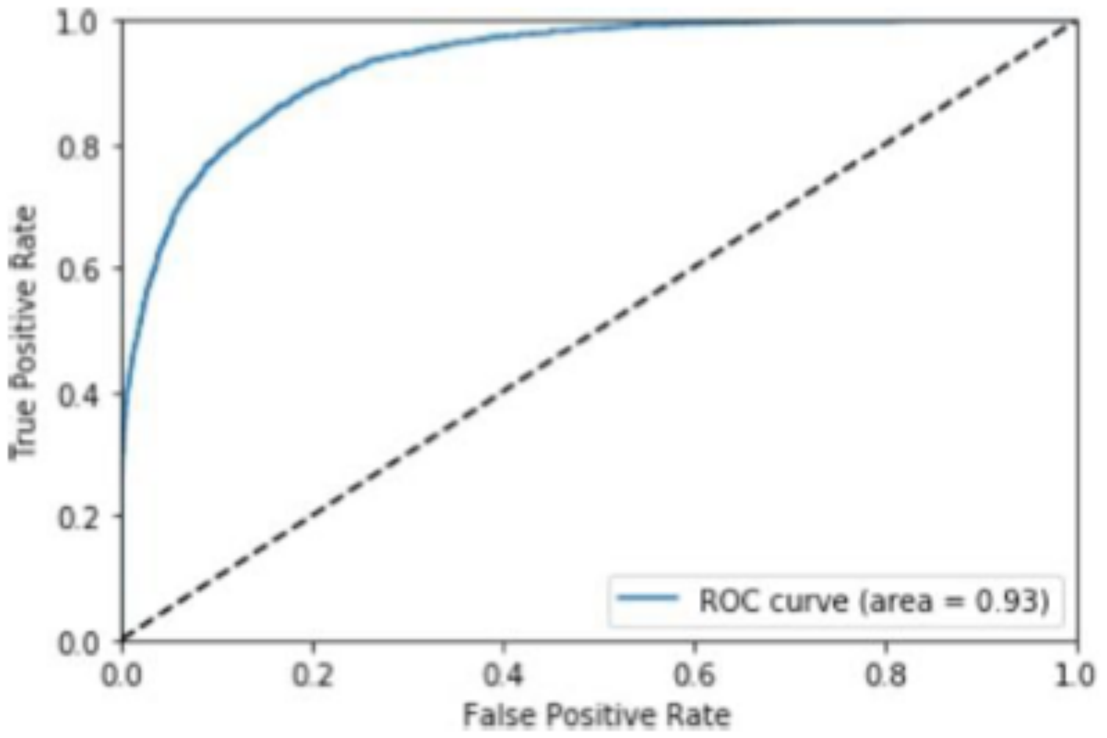
Predicted label vs True Label

we will tune and check the confusion matrix.

```
              precision    recall  f1-score   support

           0       0.97      0.88      0.93      4990
           1       0.89      0.98      0.93      4898

    accuracy                           0.93      9888
   macro avg       0.93      0.93      0.93      9888
weighted avg       0.93      0.93      0.93      9888
```

ROC Curve for area under the curve :

Drawing graph for the ROC curve.



-According to the obtained Training and Validation Accuracy, it can be concluded that the model is a good fit.

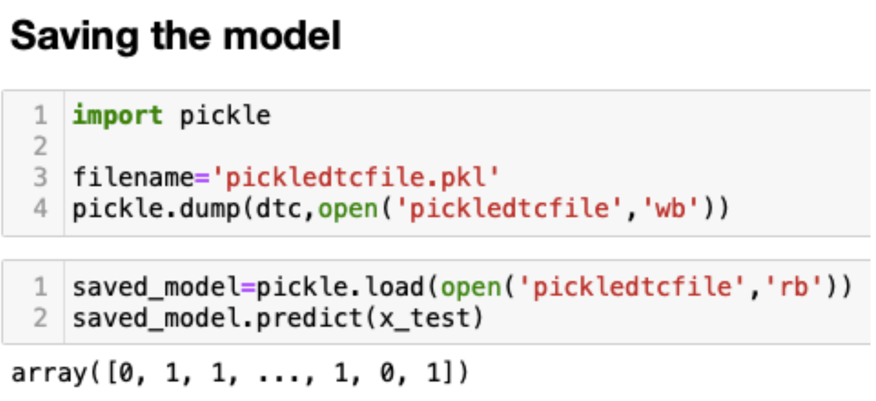
- The Area Under the Receiver Operator Characteristic in the graph is a descent one, as more the AUROC (towards 1.0), better the performance of the model.

Saving the model :

We can save the best model with pickle method or joblib.

I have used pickle method to dump and save my model.

The best model is **DecisionTree Classifier** with accuracy score as 1.0 and cross-validation score 0.02.



Saving the model

```
import pickle

filename='pickledtcfile.pkl'
pickle.dump(dtc,open('pickledtcfile','wb'))
```

```
saved_model=pickle.load(open('pickledtcfile','rb'))
saved_model.predict(x_test)
```

```
array([0, 1, 1, ..., 1, 0, 1])
```

Conclusion :

In this project , after building various machine learning models , I choose DecisionTree Classifier as the best suited model for prediction of the classification of income as '<=50K' or '>50K'.With the help of the selected model we can predict income of any population with the selected features for analyzing.We will be giving suggestions based on the result obtained which level of qualification can lead to a higher income and people of which age group are earning more.

Jupyter Notebook (github-link) :

https://github.com/ShivaniKataria05/Analysis-Projects/blob/main/Datatrained-Census-project.ipynb

Declaration :

I am obliged to the **Datatrained** team, teachers ,project mentors, technical & non-technical support teams for their guidance and continuous support.