In this blog, we will analyze the Census Dataset, this data was extracted from the [1994 Census bureau database](http://www.census.gov/en.html) by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics).Link to the dataset is here: <https://raw.githubusercontent.com/dsrscientist/dataset1/master/census_income.csv>

**Defining the problem statement**

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.

First, we will import the required libraries for reading and EDA(Exploratory data analysis)

import pandas as pd

import numpy as np

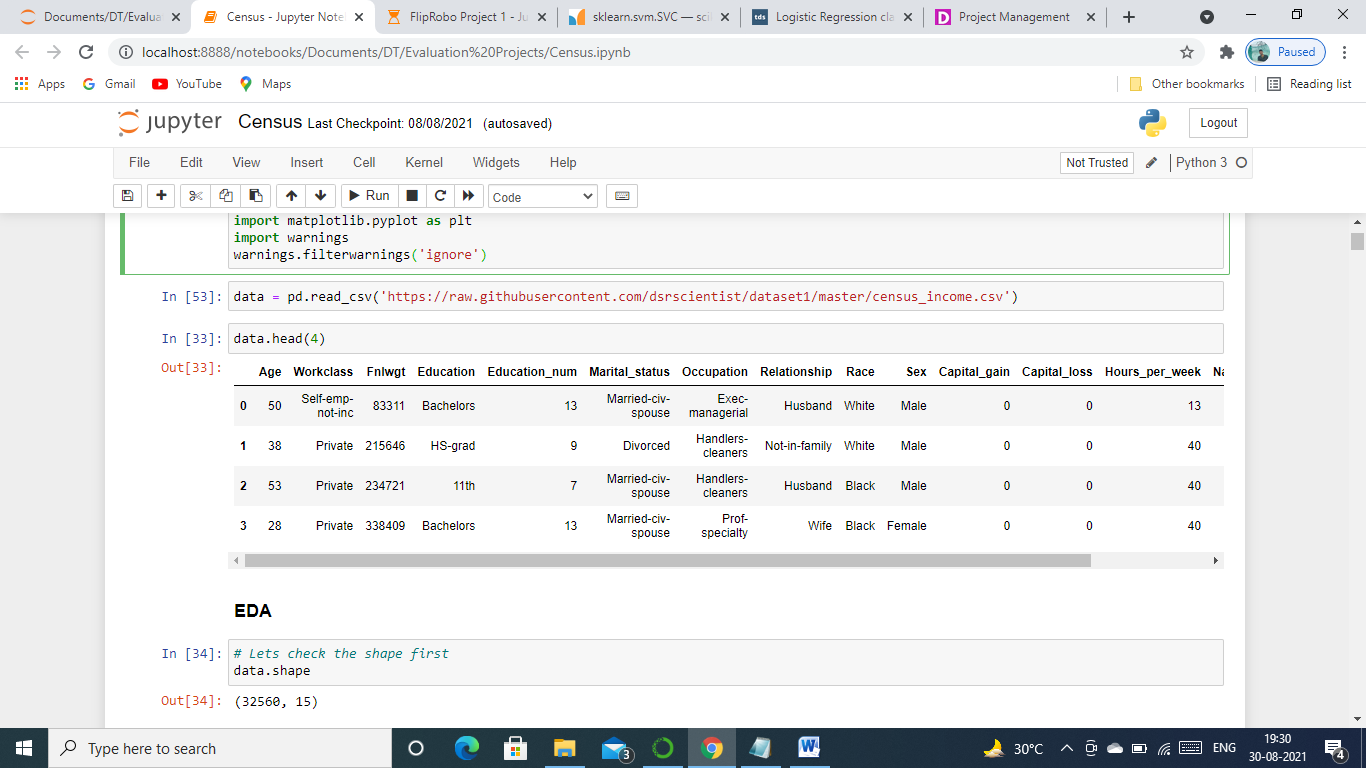
import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

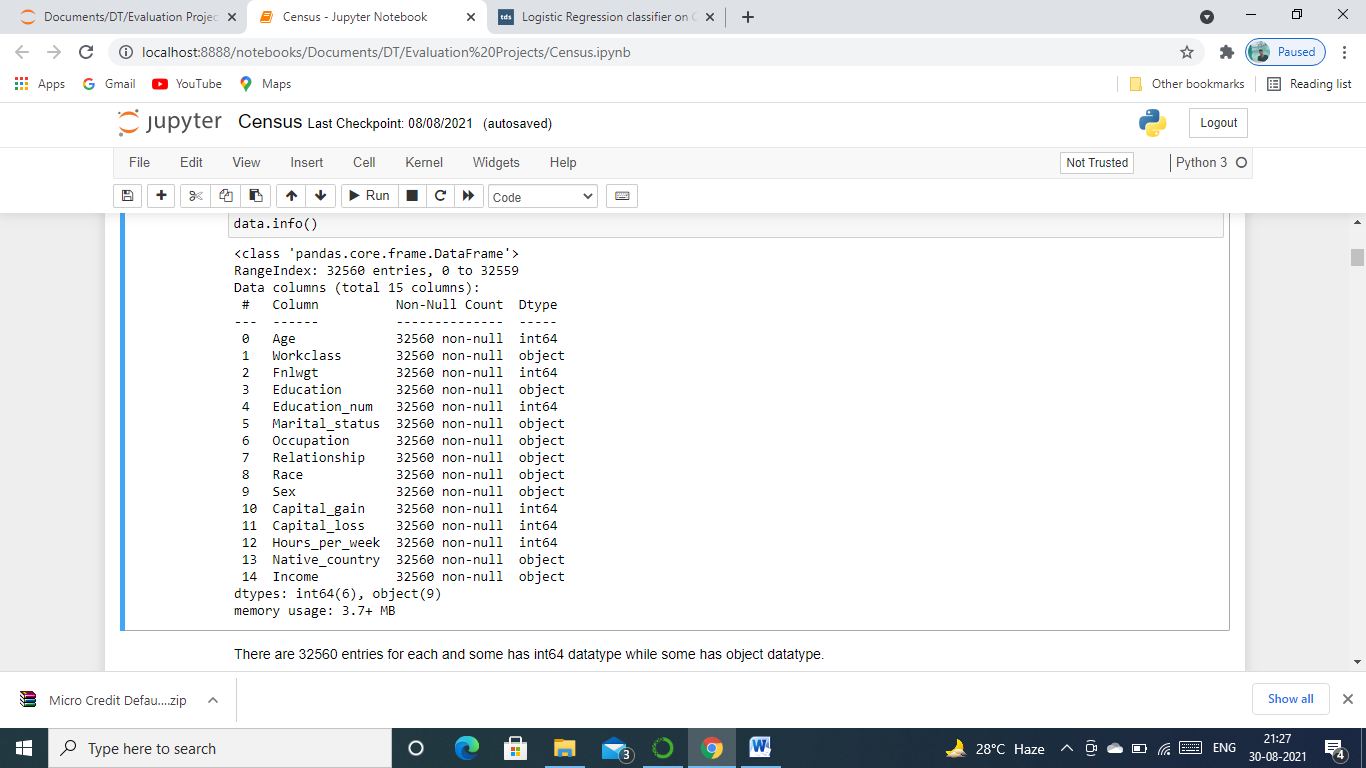
Next, we load the data into a **pandas** dataframe using the **read\_csv** function



We can see above that data is successfully loaded into pandas dataframe, it is stored in variable data which we will use further to do EDA, Cleaning and Model Building.

**Exploratory Data Analysis**

Let’s get more information about the training data using  data.info()



Above figure gave us the detailed information of our dataset. Let us see what we can find from the figure.

* There are 15 columns in the dataset out of which 14 are features and one is label which is our output(Income).
* There are 32560 samples in the dataset.
* There are no missing values in it.
* There are both categorical and numerical columns in the dataset
* Workclass, Education, Marital\_Status, Occupation, Relationship, Race, Sex, Native\_country and Income are categorical and rest of them are Numerical.

More information about the data can be gathered by using data.describe()

|  | **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 |

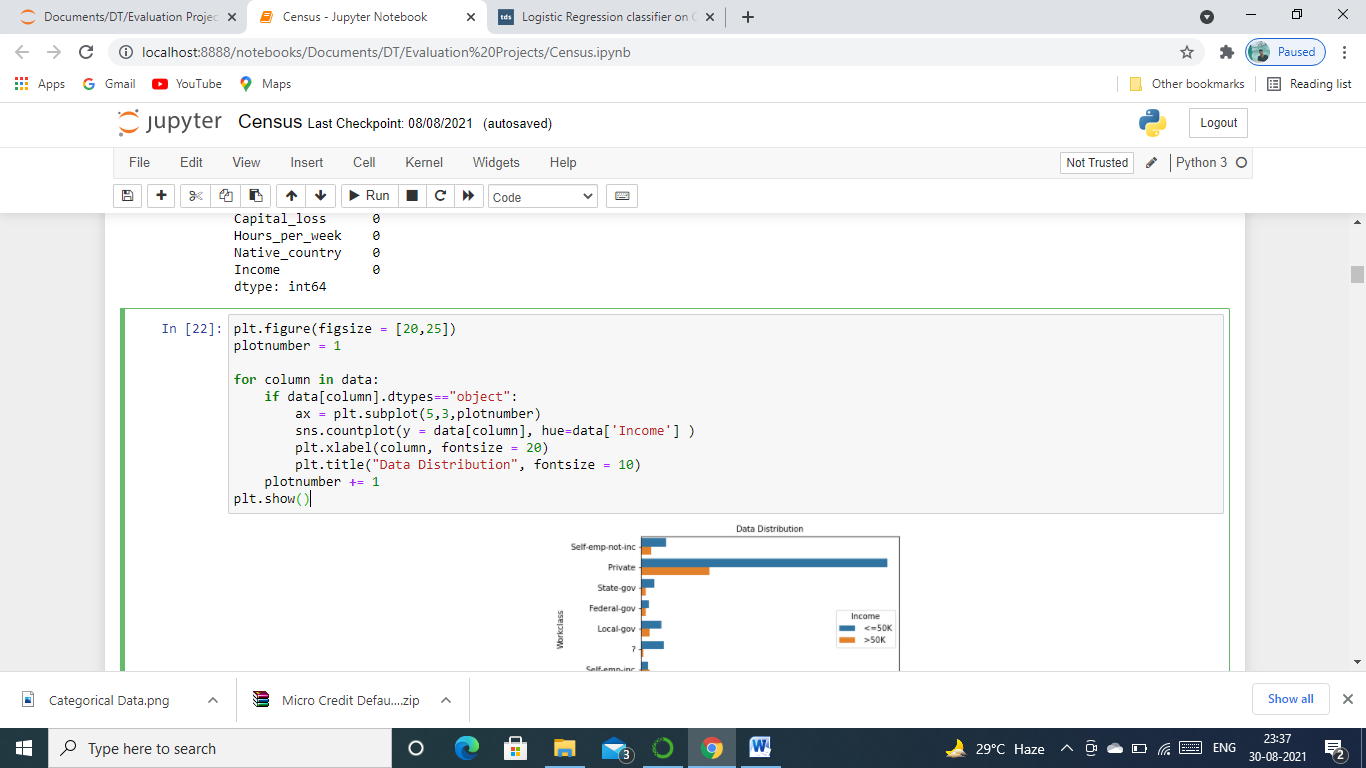
As we can see above it gives information regarding only the numerical data. Let see what we get from it.

* We see the mean, standard deviation and 1st, 2nd and 3rd quartile with it.
* The values are on different scales. Many machine learning models require the values to be on the same scale. To scale the data we will use StandardScaler after cleaning it.

**Data Visualization**

**We will use countplot from the seaborn package to analyse the categorical columns.**

**Here is the code which we can use to plot it.**

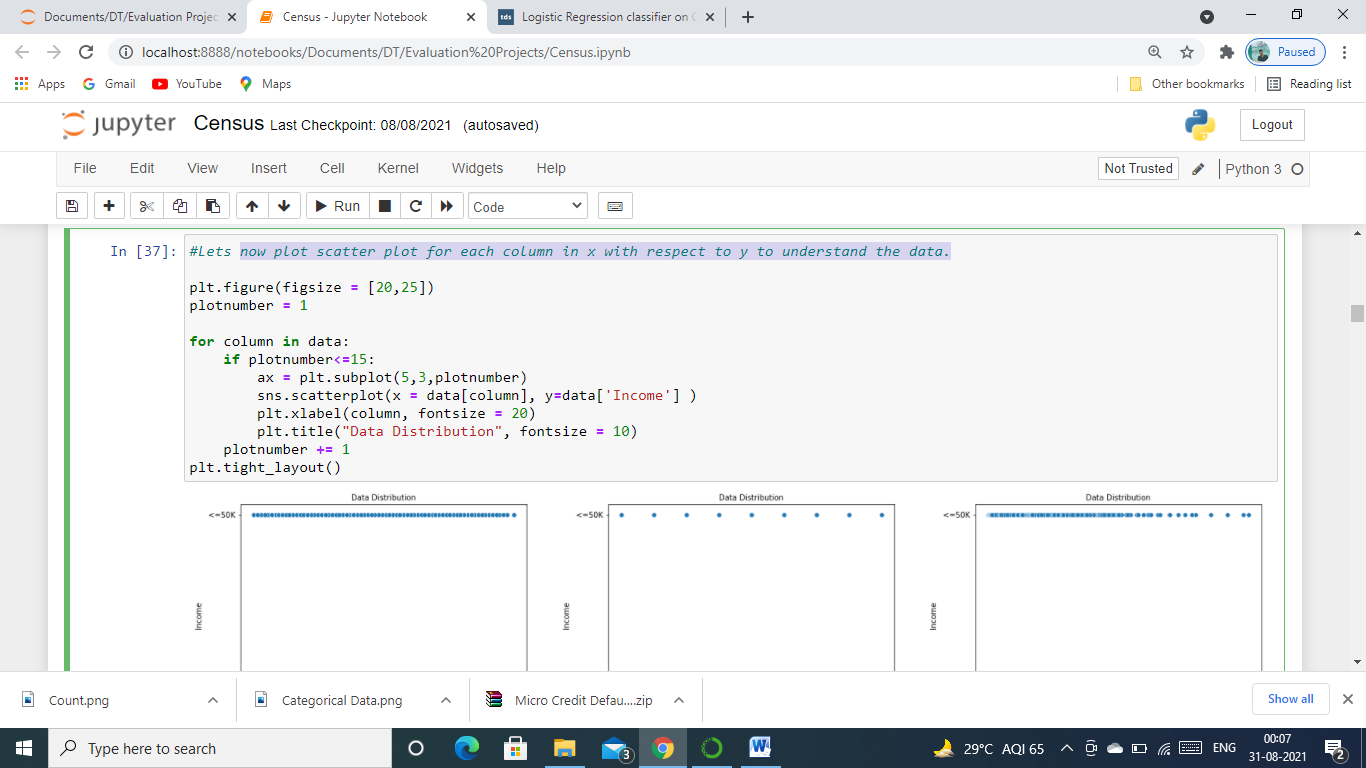




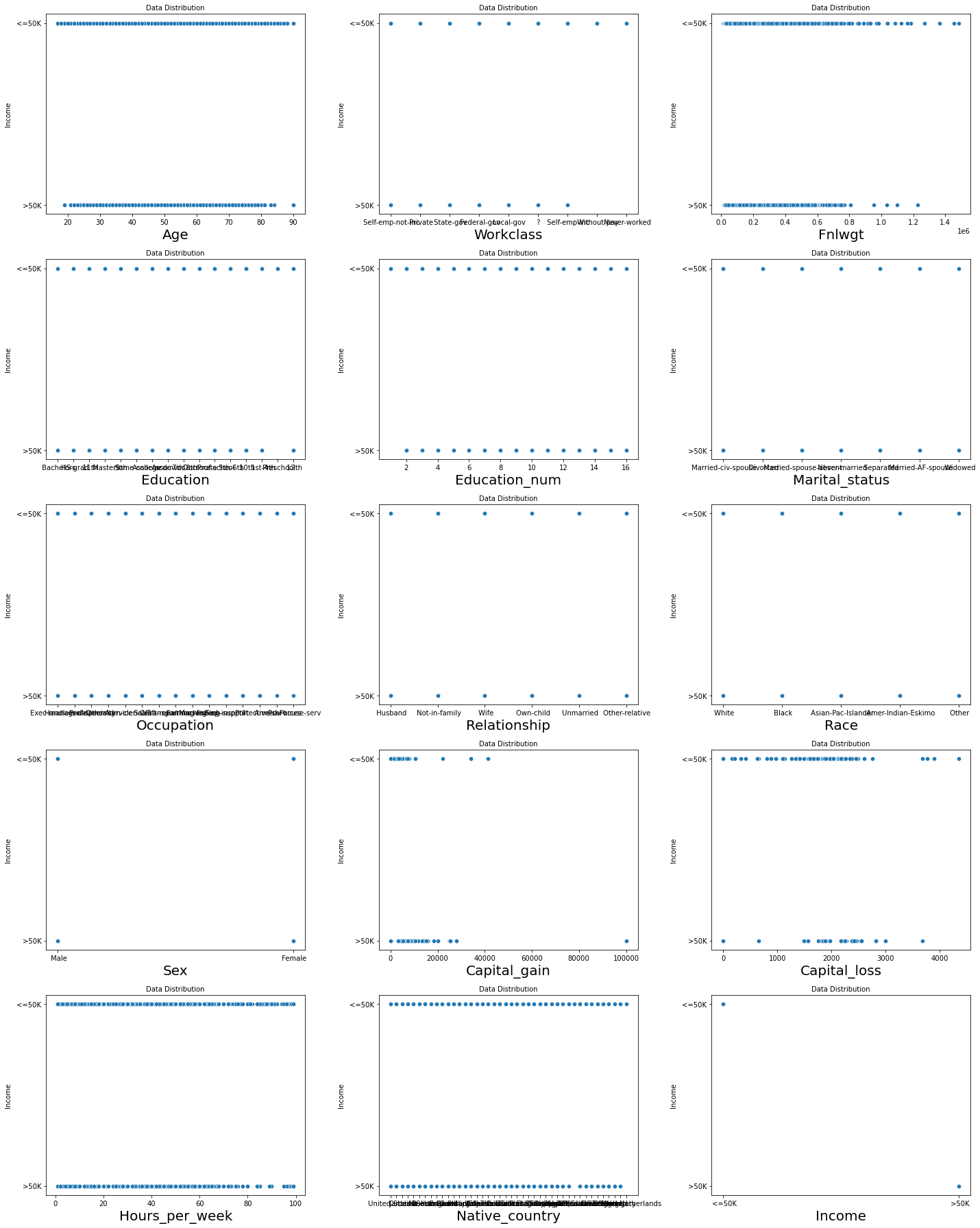
**The above figure gave us the information of how our income is related with the categorical features, And we can from this figure out the following things:**

* Those who never worked and without pay never had income of more than 50k.
* Those who studied upto 10th never got income of more than 50k and have very less income and those who continued their study is getting more than 50k and some have less than 50K.
* **People who are earning more are mostly Unmarried, Divorced and Married-civ-spouse or mostly are husbands, belongs to race White and Males are more as compared to females though some of them are also earning more than 50K.**
* Exec-managerial occupation have highest numbers of people earning more than 50K and Adm-clerical have highest when earning is less than 50K.

**As we saw above the categorical data we will now plot scatter plot from Matplot library for each column in x with respect to y (Income) to understand the data. The code for which is below.**



**Now let’s see the plot and gather some information from it.**



**Below are the observations from scatterplot.**

* We see that there are less number of employees getting income more than 50k between the age group 80-90 otherwise it is equal.
* Person having fnwlgt between 0.75 and 1.50 have less chances of income of more than 50k
* Also we can find that who have more capital gain have chances of income of more than 50k.
* People having less capital loss are having higher chances of income of more than 50k.

**Since there are 9 categorical columns in the data we will first encode them for this we are going to us Label Encoder, we have to first import it and then use for categorical columns.**

#Lets first convert the string values in int for further processing of data with the help of label encoder.

from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()

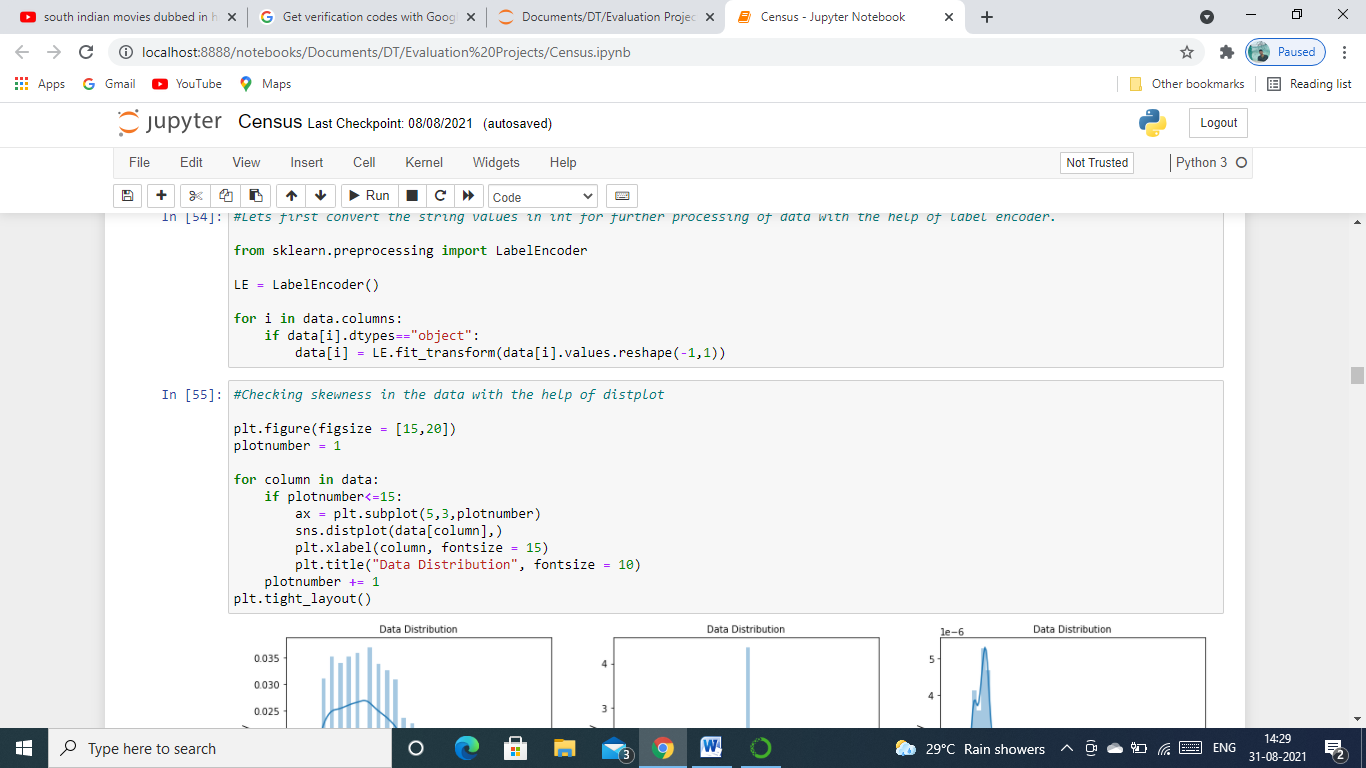
for i in data.columns:

if data[i].dtypes=="object":

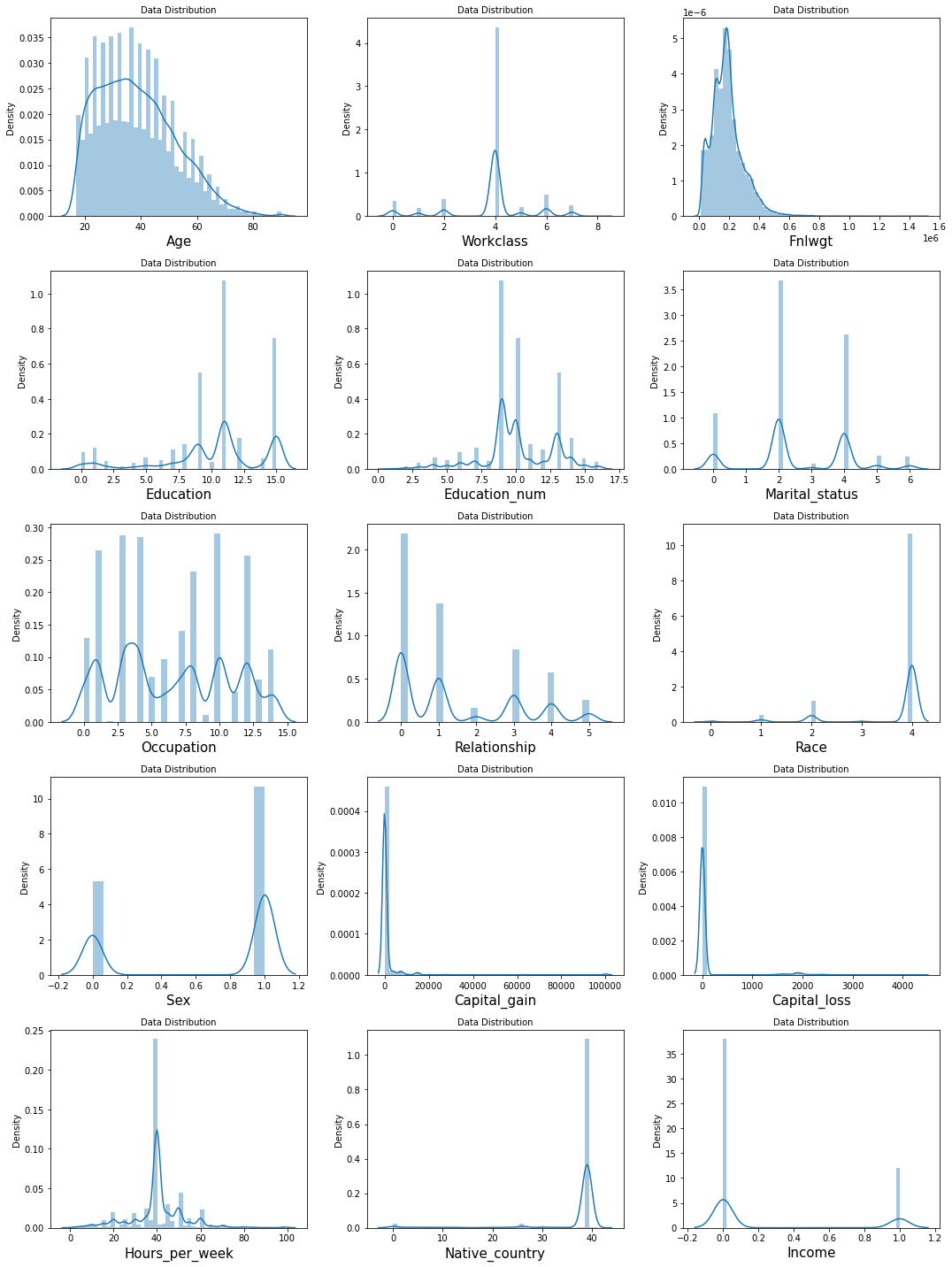
data[i] = LE.fit\_transform(data[i].values.reshape(-1,1))

This code encoded our categorical columns into numericals which will help us to do further analysis and model building.

Let’s us now check for skewness in the dataset, what is it? Skew is the degree of distortion from a normal distribution, normal distribution forms bell curve and is consider good for research. We will use distplot in seaborn library to plot it for the dataset. Below is the code for it.



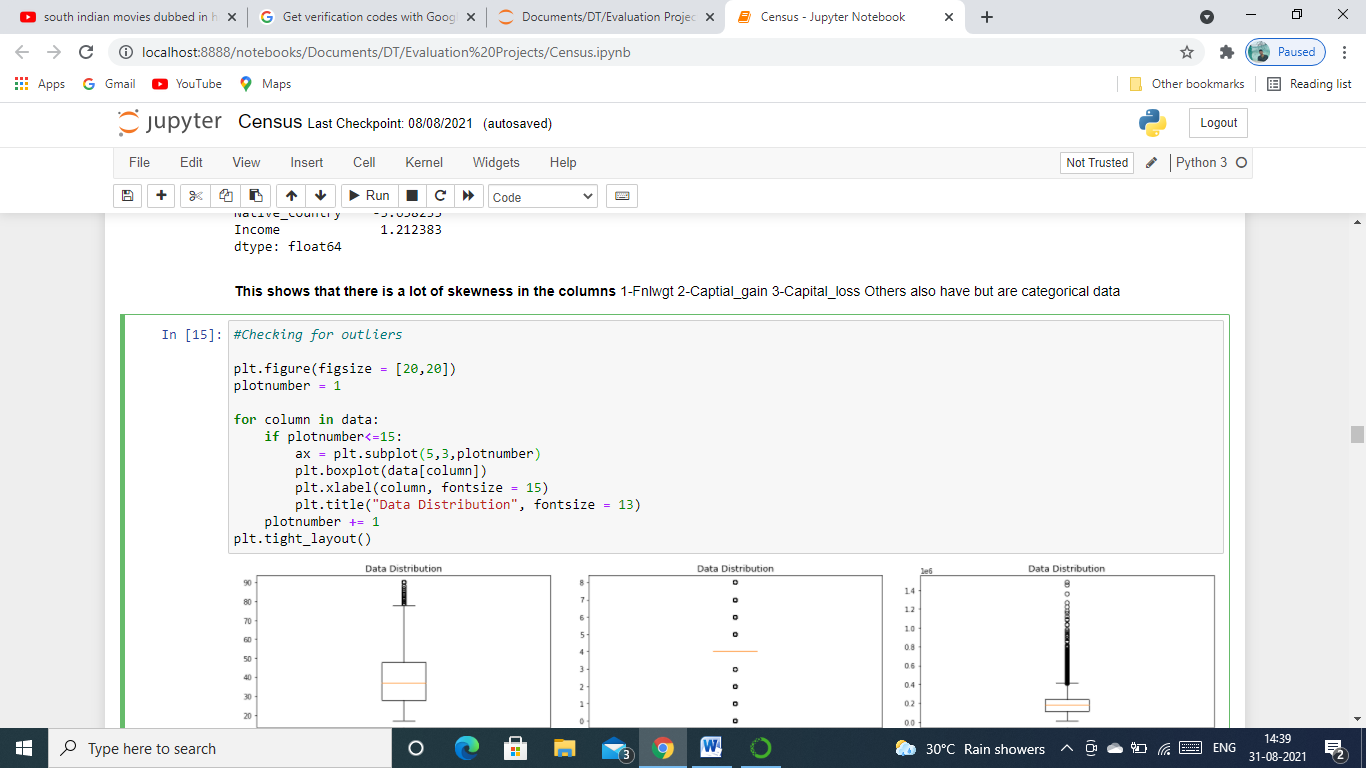
Here is the output of our above code.



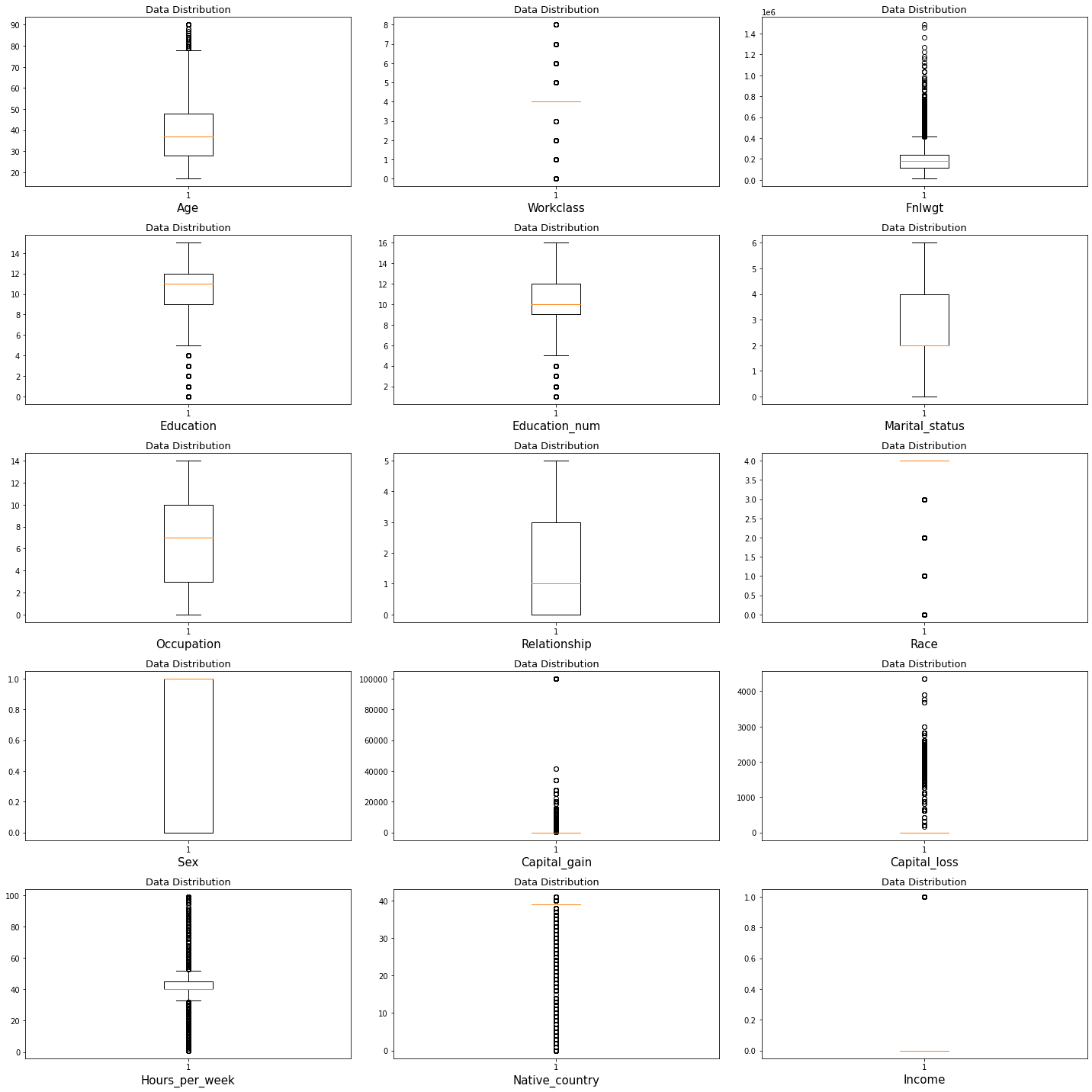
From the above figure we can find the below observations.

1-Fnlwgt 2-Captial\_gain 3-Capital\_loss are continuous data and have skewness which means not normal and is not good for our model, these are skewed towards right as we can see in the figure. Others also have but are categorical data which we do not consider for building a model.

The next step is to check for outliers, what are these? An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. Outliers are not good for model hence we will check it by plotting boxplot which is in Matplot library with the below code.



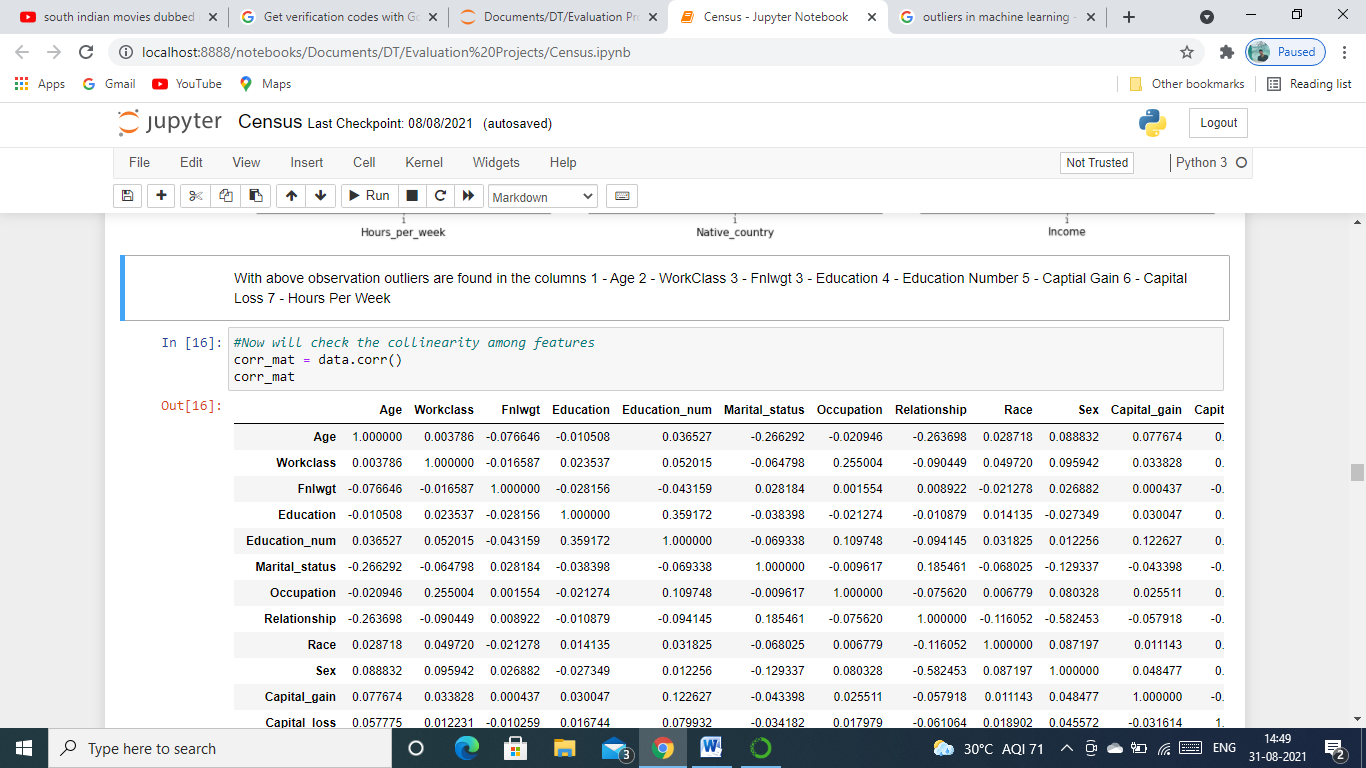
This code will give us the below output.

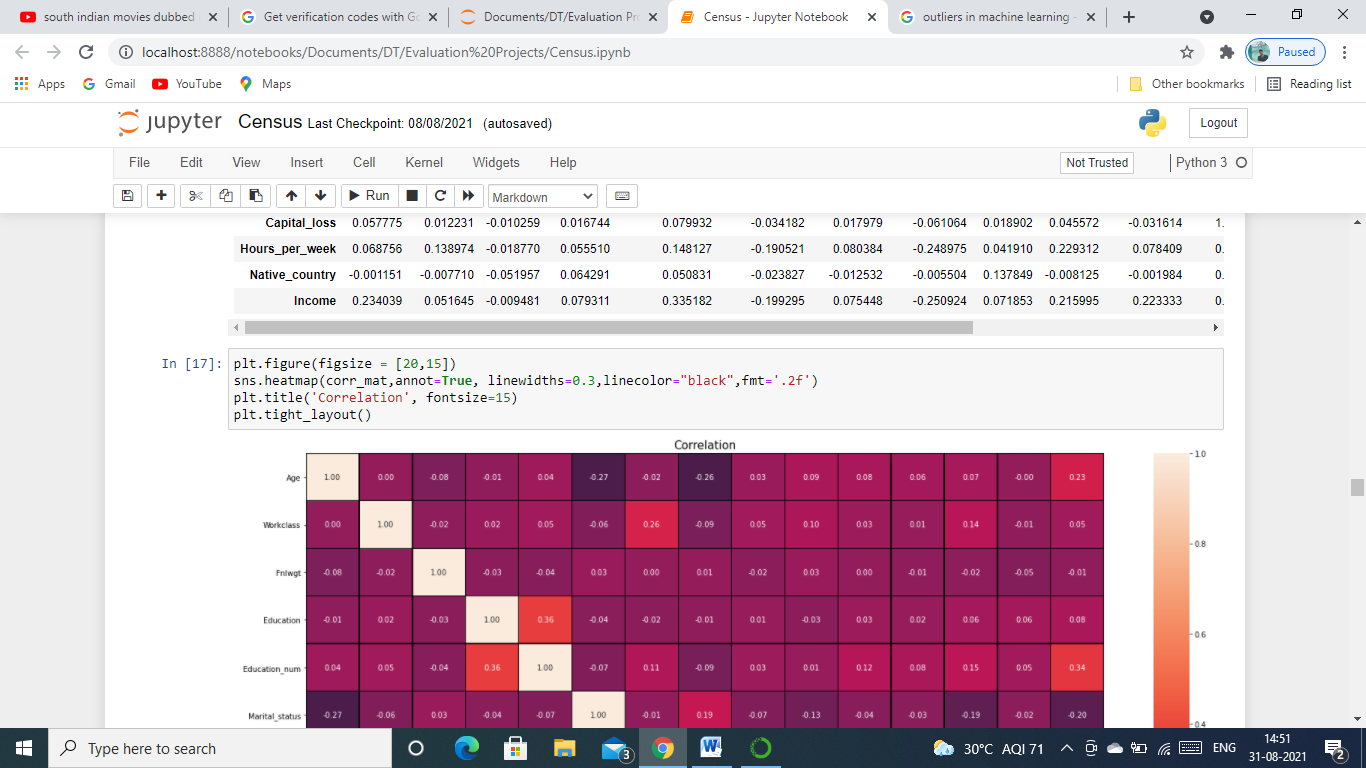


What can observe from the above figure?

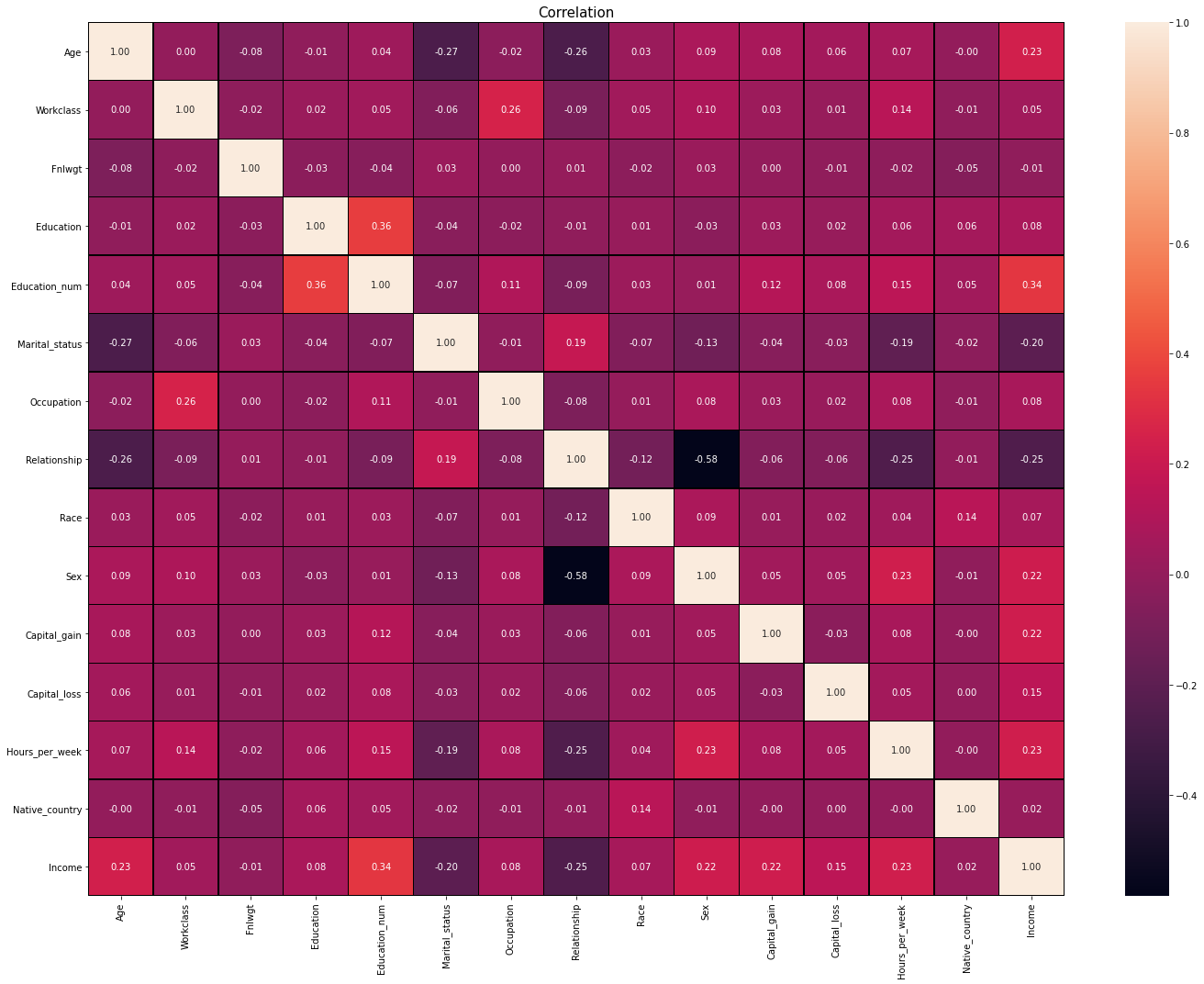
With above observation outliers are found in the columns 1 – Age, 2 – WorkClass, 3 – Fnlwgt, 4 – Education, 5 - Education Number, 6 - Captial Gain, 7 - Capital Loss, 8 - Hours Per Week. These outliers needs to be removed from entire dataset so that we can build a good model for this problem statement.

Now we will look for co-linearity among features, how can we do that? Well for this we will plot heatmap which is also available in seaborn library and is only used to plot the co-linearity to understand it better. Below is the code for it.





This code will give us the below heatmap for us to understand the co-linearity among features in our dataset.



What can be observe from the above figure?

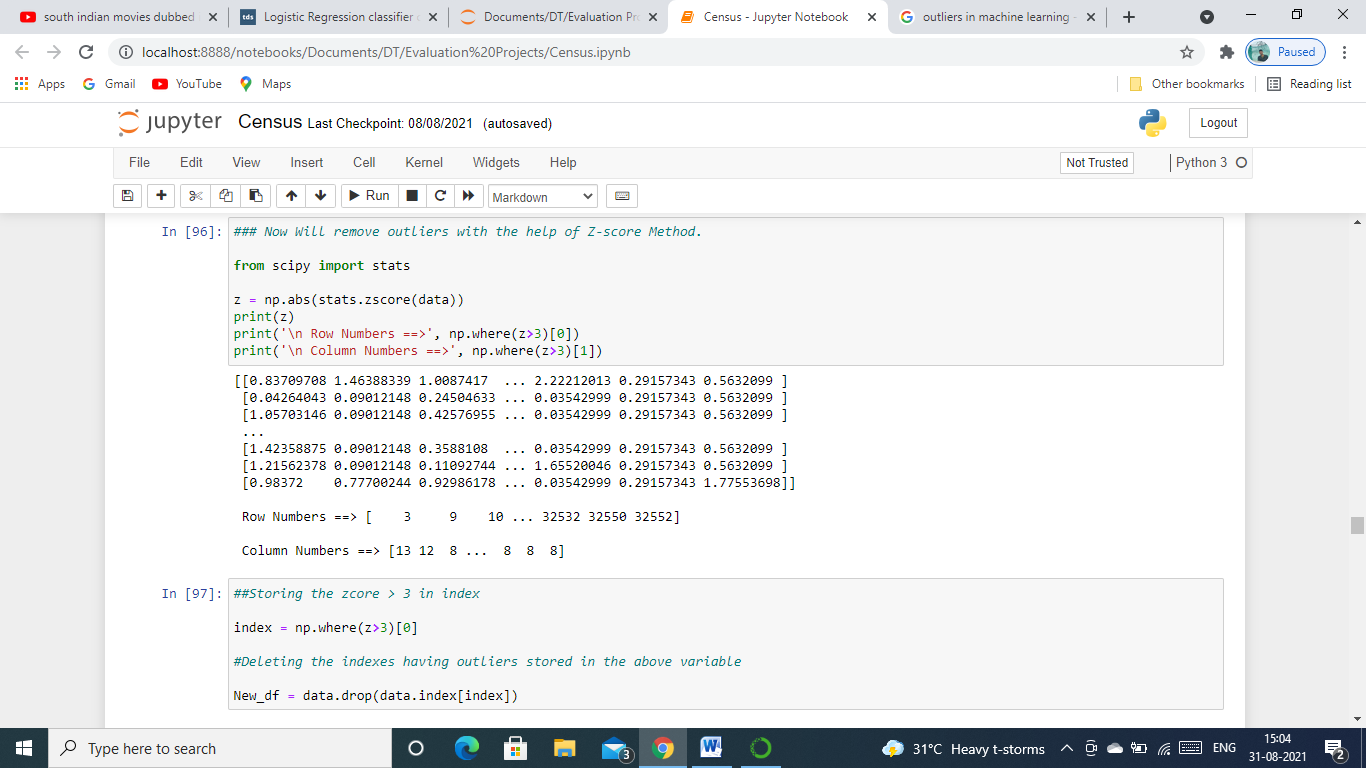
* Max Correlation of the output with the features columns is with Education number(34%).
* Min Correlation of the output with the features columns is with Fnlwgt(-1%).
* Also we can see that there are not much corelation among the features.

Since there are not much co-linearity among feature we do not have to clean or remove any feature from the dataset for building our model.

Now since we visualized all the important factors, we will clean it for building a good model.

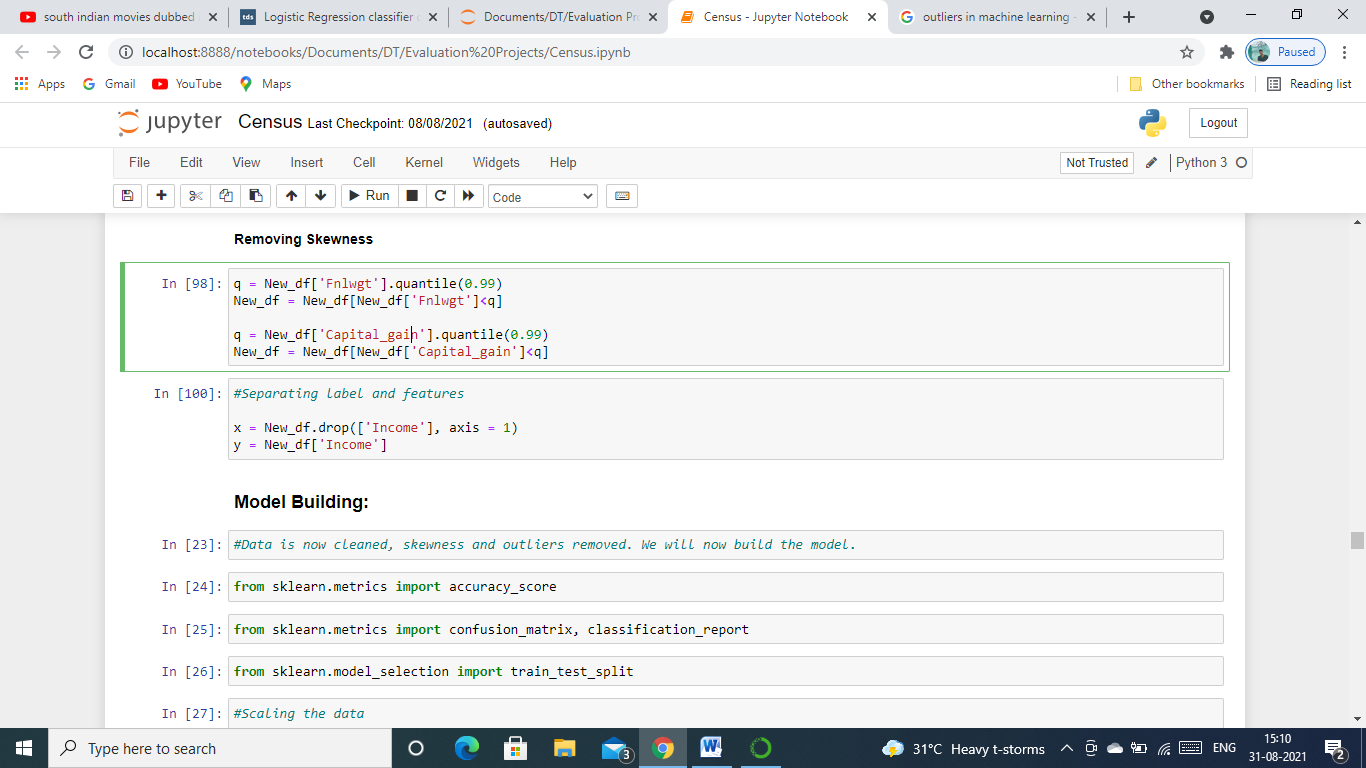
**Data Cleaning**

As we saw above that there were outliers in the dataset, we will remove these with the help of zscore method , we will import the necessary library required for it and then we will apply zscore on the dataset to remove using data.drop() and store it in a new variable New\_df. Below is the code and output for removing the outliers.



Since we have stored the data without outliers in a different variable New\_df, the next step for us would be to remove the skewness as we observed before that we had skewness in two of the features, Fnlwgt and Capital\_gain. We will remove skewness with the help of quantile method.

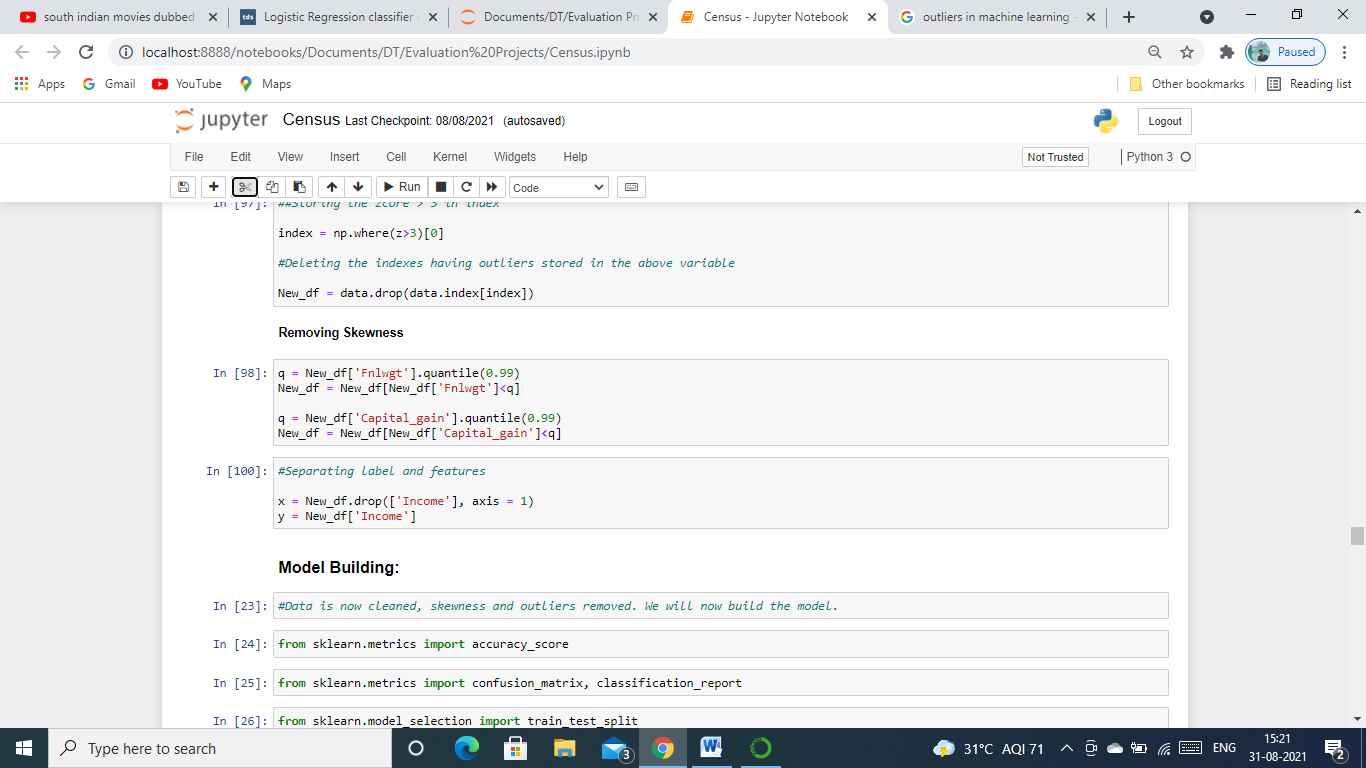
Code for it is below.



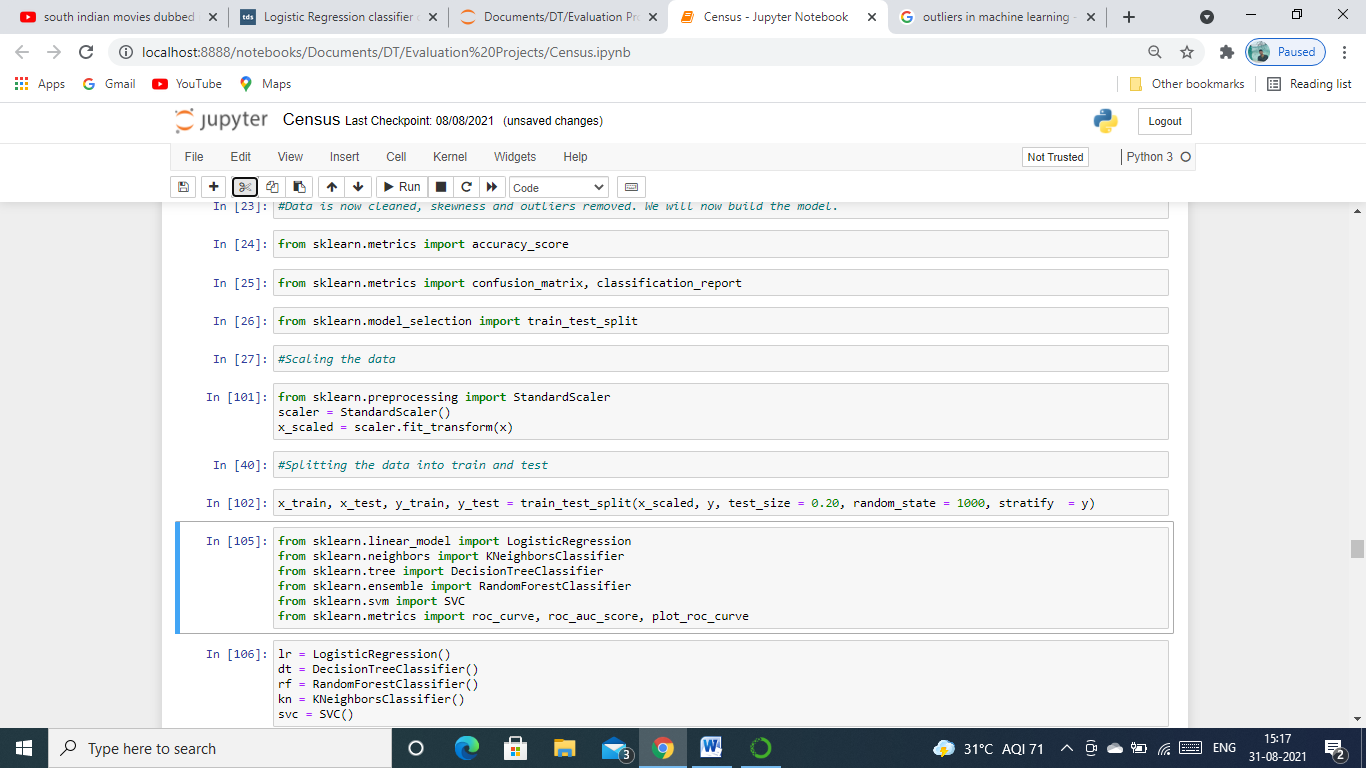
Now that our data is clean, removed all the outliers and skewness from it, Next step we finally can say that our data is good for building model.

**Model Building:**

Let us now first separate our features and label into X and Y variable to build our model.

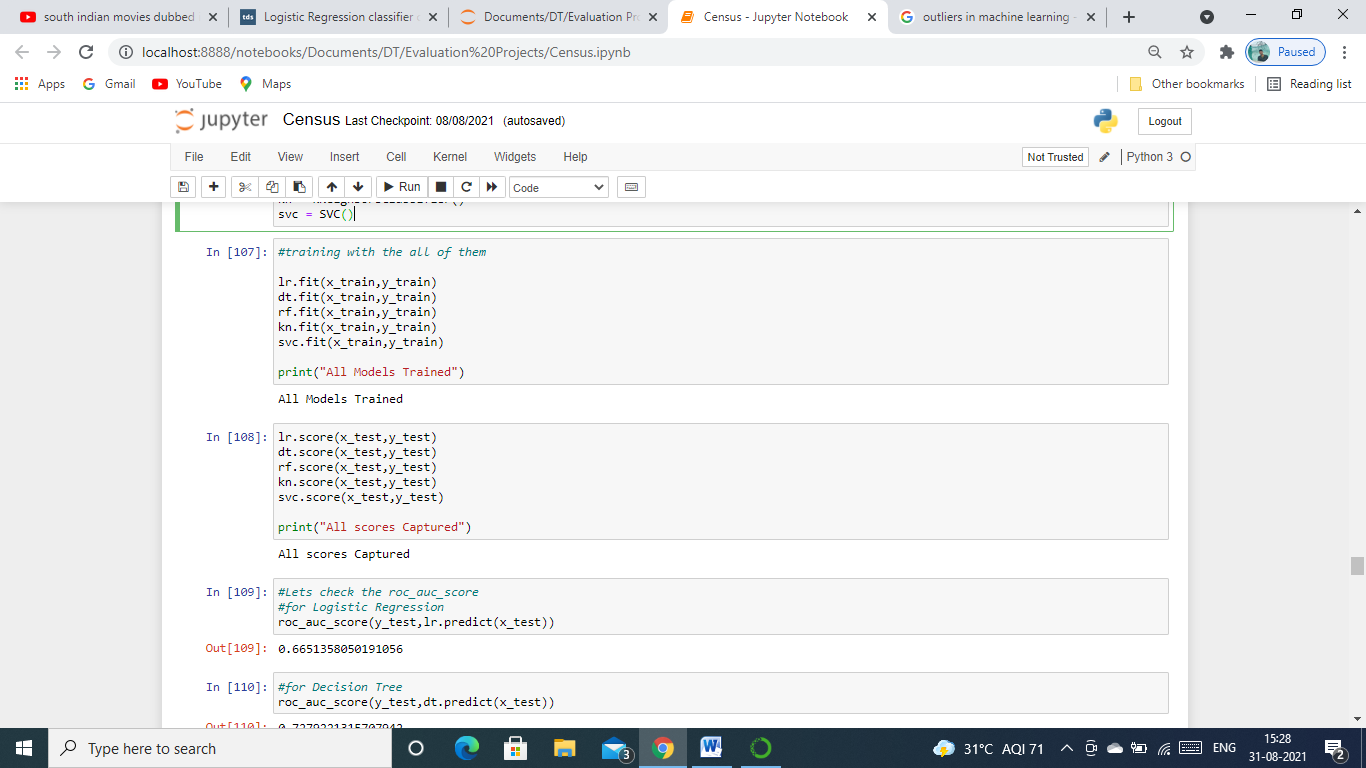


Now we are going to import the required algorithms and metrics which we will use to train and test the model and store them in a variable and then scale our features with the help of StandardScaler as we saw previously that they were on different scale. Below are the codes for all this.

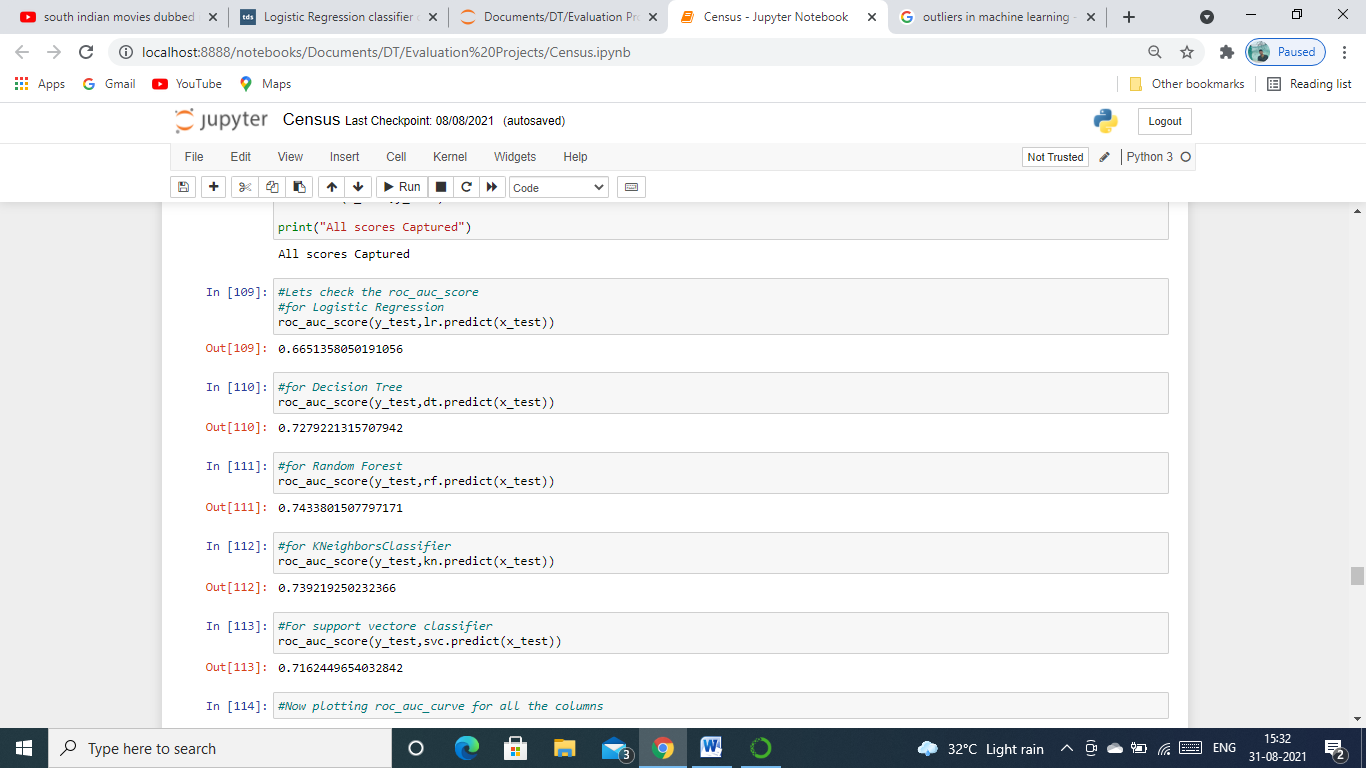


Above we can see that dataset into train and test sets using train\_test\_split method, kept the test size to 20% and we used Stratify=y to balance the label.

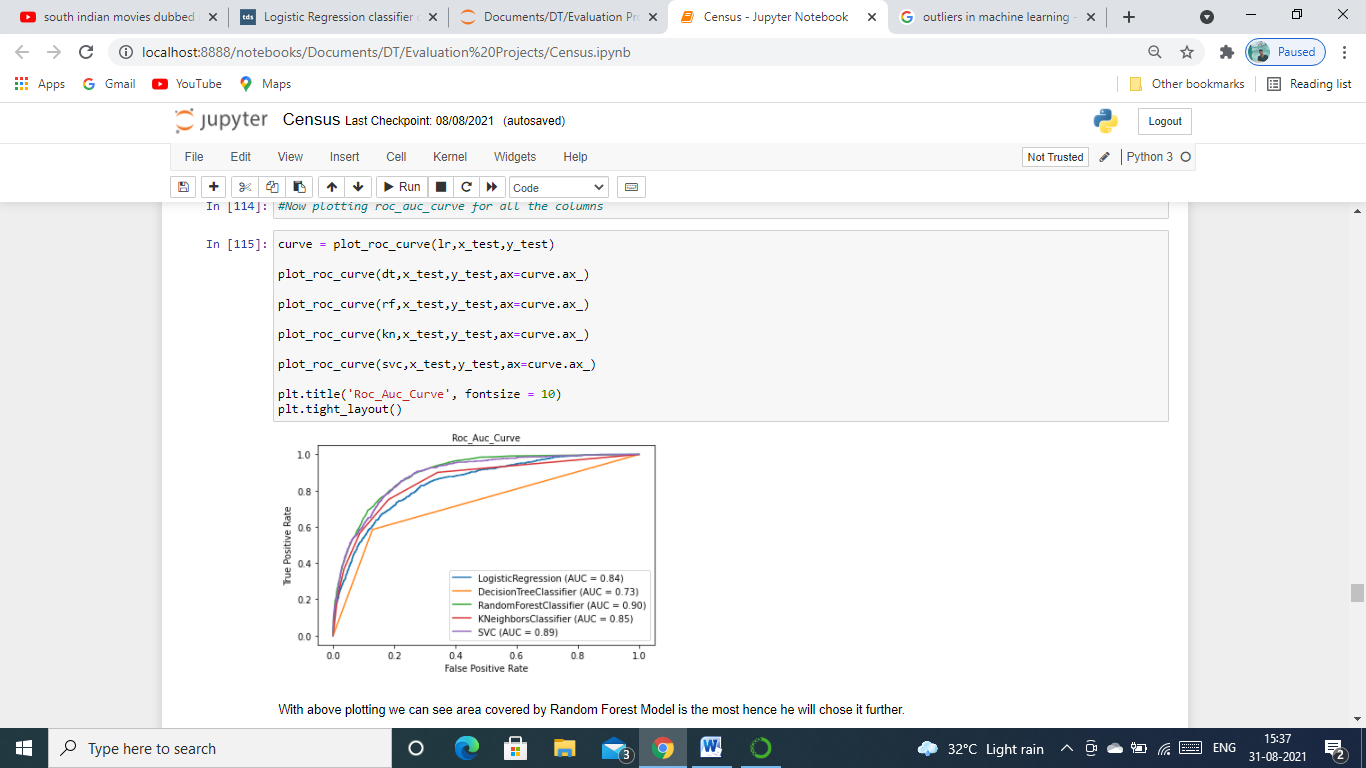
Let’s train the model and capture all the scores. We can do that using below codes.



Now that our model is trained using all the all the algorithms, we will now see how good it performs using roc\_auc metric which we imported previously. Below are the codes and output for each algorithm.

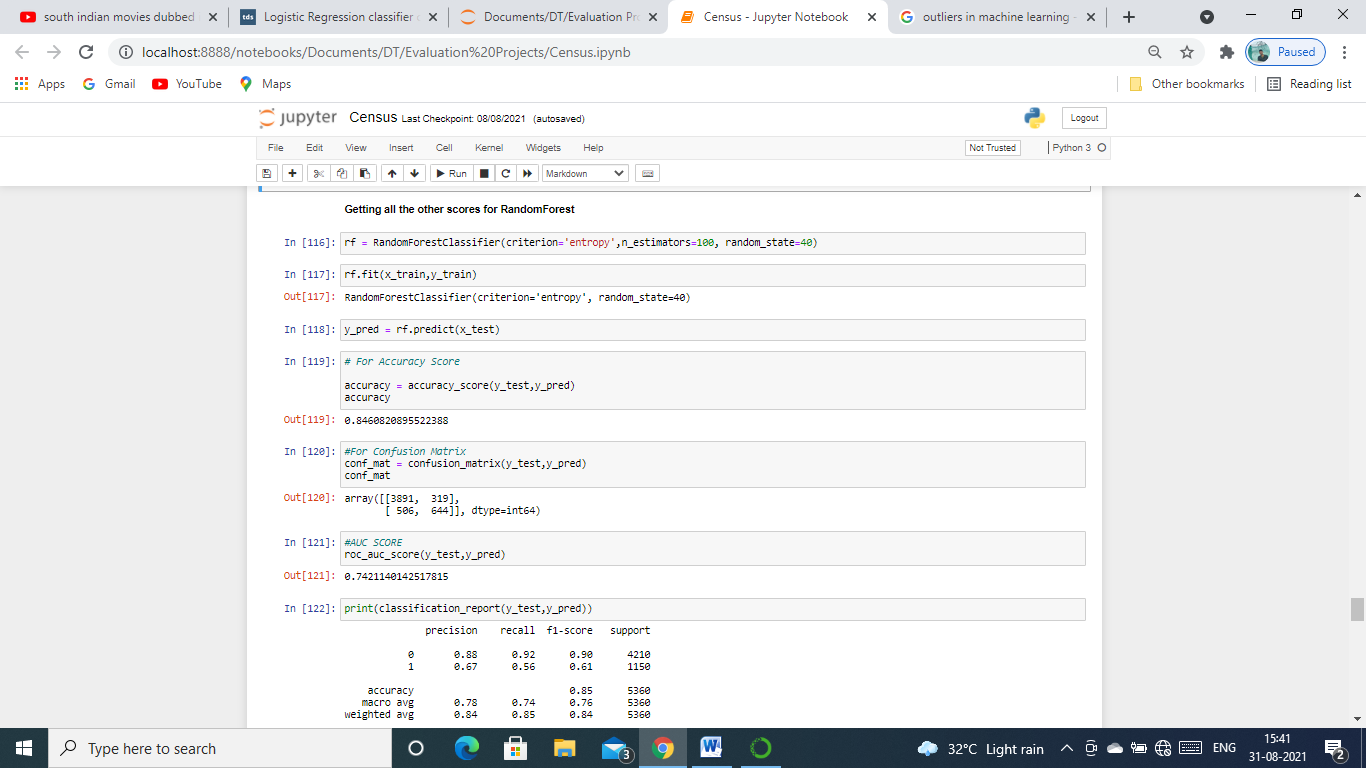


As we can see above that Random Forest model is giving us the best predictions which is of 74.33% on this dataset. We will further plot an auc\_roc curve for all the algorithms and see which of them is covering most of the area, model which covers most of the area is considered to be good for classification problems. Below is the code for plotting it.

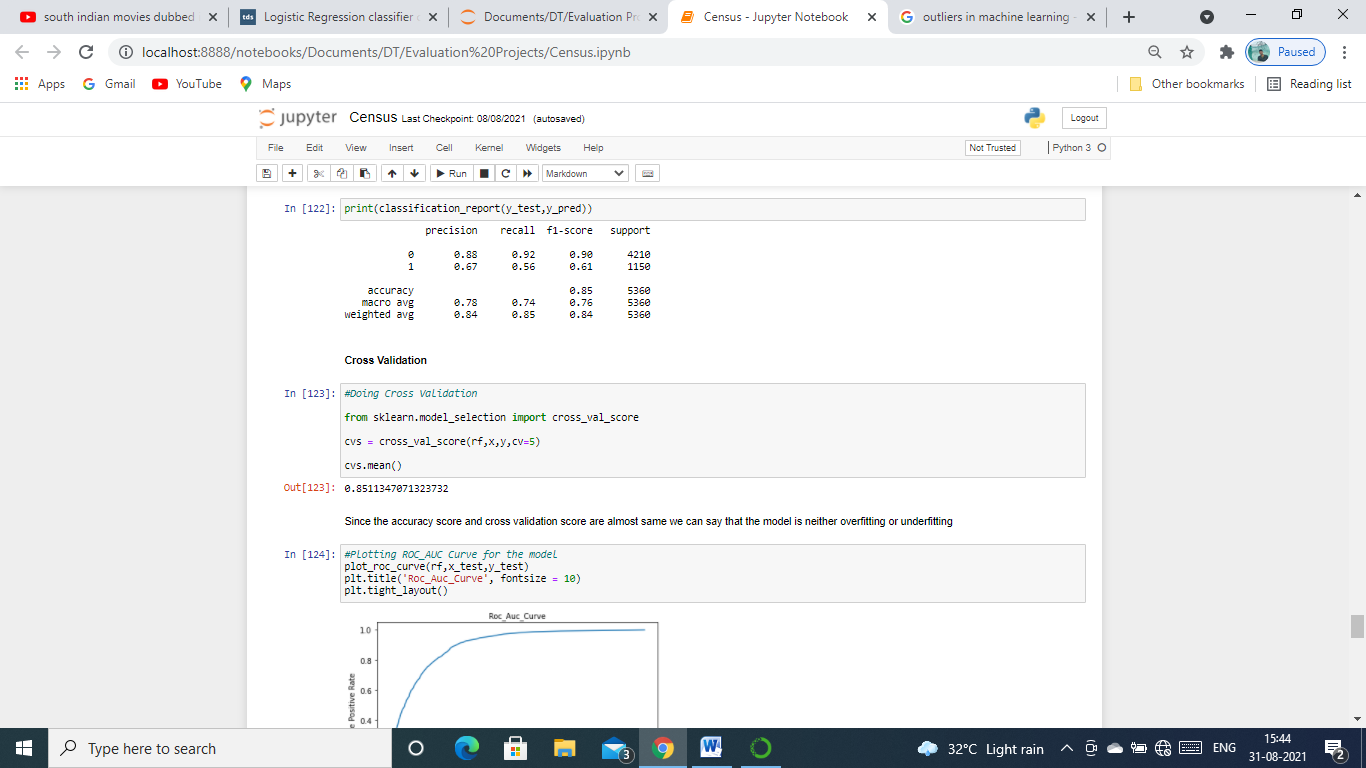


With above plotting we can see area covered by Random Forest Model is the most hence he will chose it for further evaluation and tuning it.

Let’s now see all the scores of our RandomForest model, below are the codes and scores for it.



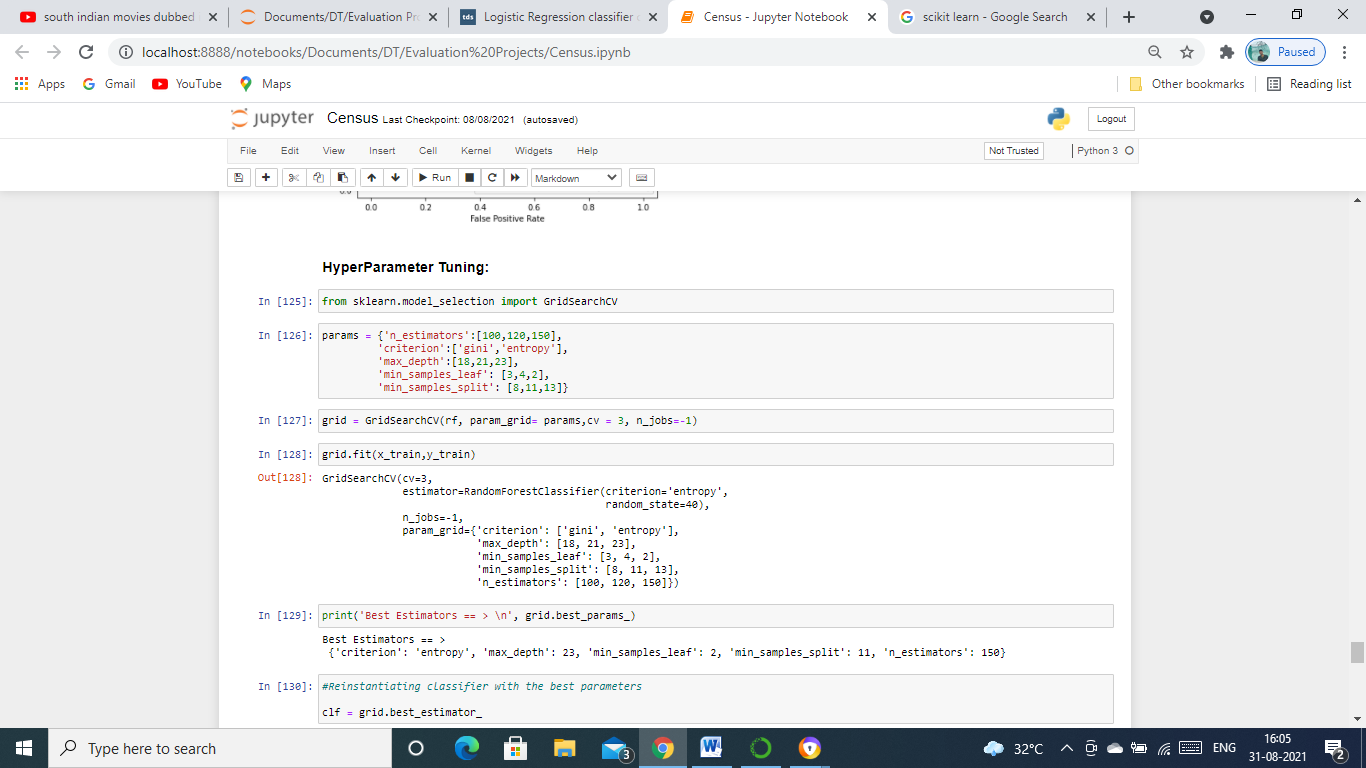
After this we will cross validate our model to see if it is overfitting or underfitting, we will import required library and then apply it on the features and label to get the mean of the score, We can do this with the following code.



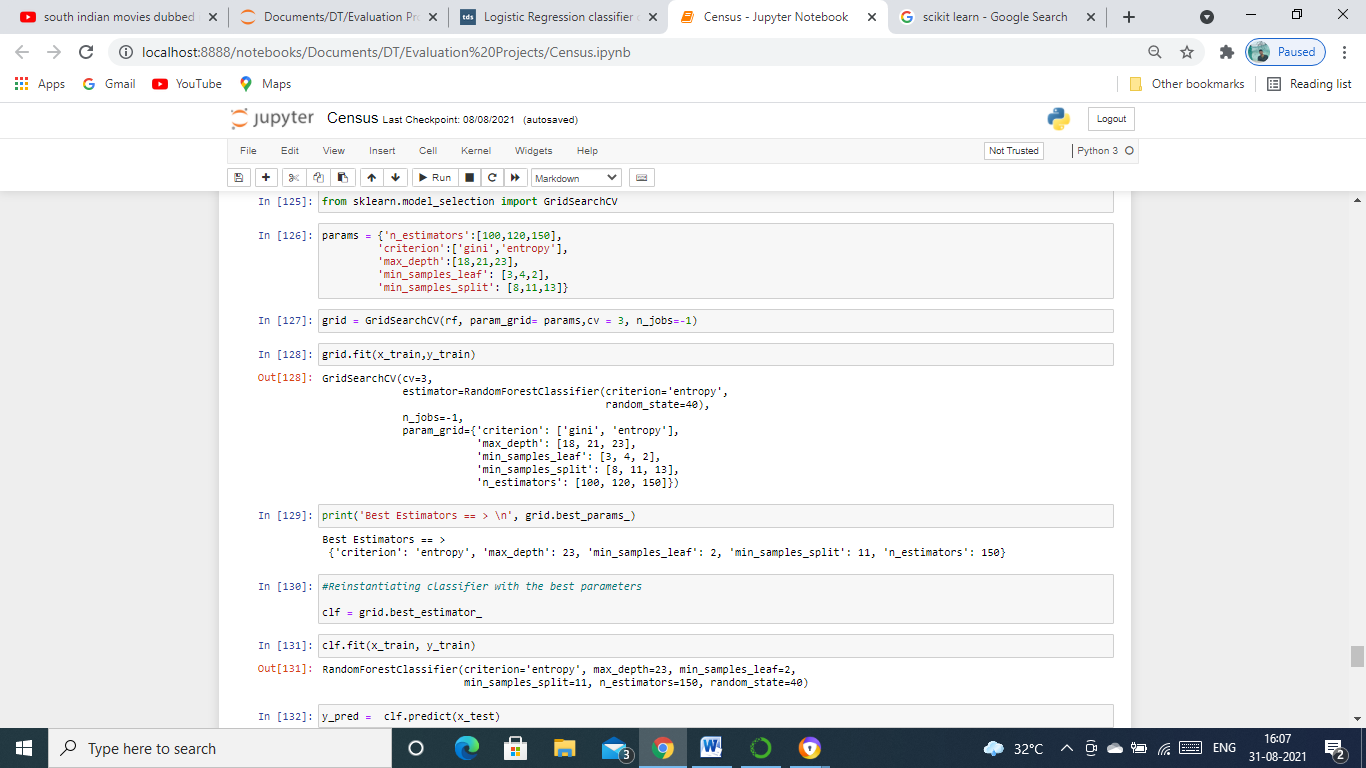
As we saw that there is hardly any difference between our cross validation score and accuracy score we can now say that our model is good, it is neither overfitting nor underfitting.

Next step would be to do Hyperparameter tuning to improve our scores. For this we will use GridSearchCV from Sciket-Learn library to do an exhaustive search over specified parameter values for an estimator and playing around with the parameters which we can find from the documentation of [Random Forest Classifier](sklearn.ensemble.RandomForestClassifier%20—%20scikit-learn%200.24.2%20documentation.html) available on Scikit Learn Website.

Code for importing GridSearch is below and we will use the following parameters.

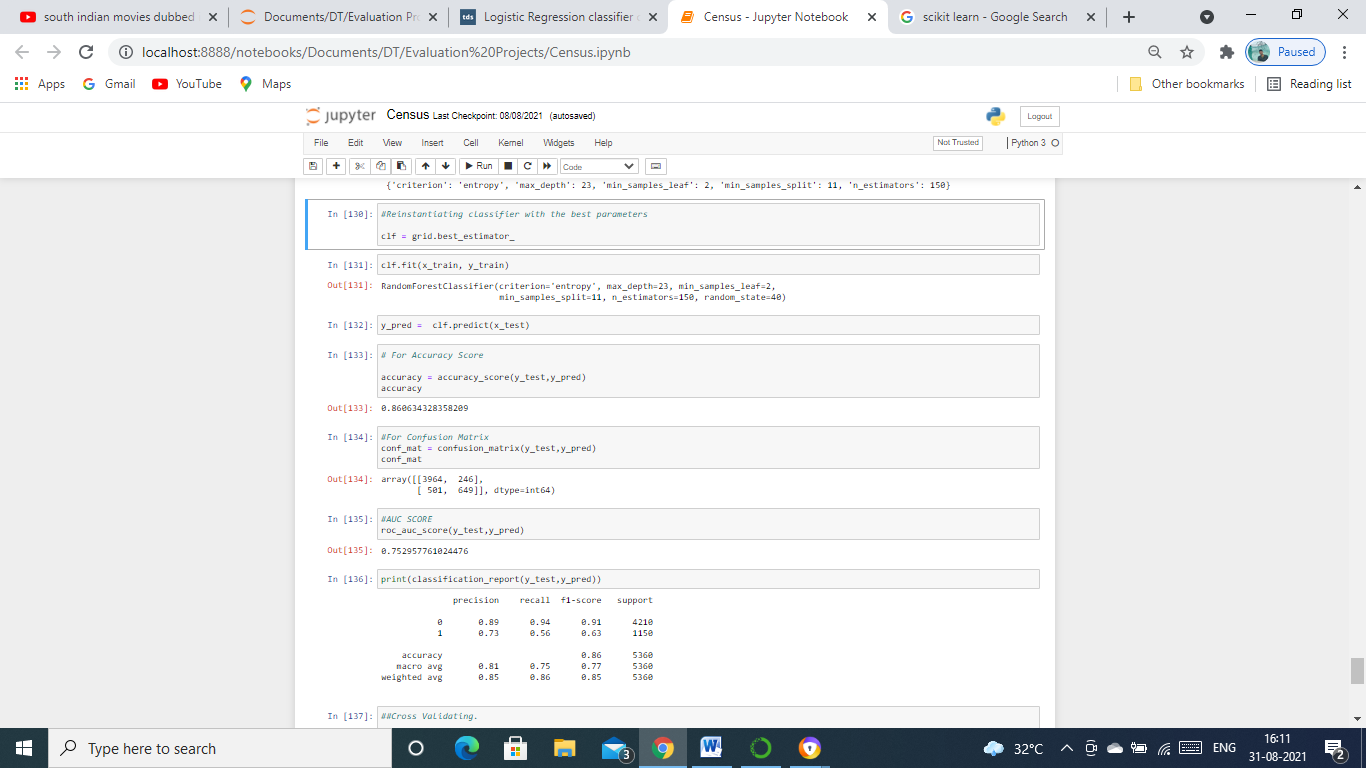


Using GridSearchCV to find the optimal parameters, below are the codes for it.



Above we can see the optimal parameters for this dataset. We will now re-instantiate with the best parameters that we got and find the all the scores using metrics imported previously.

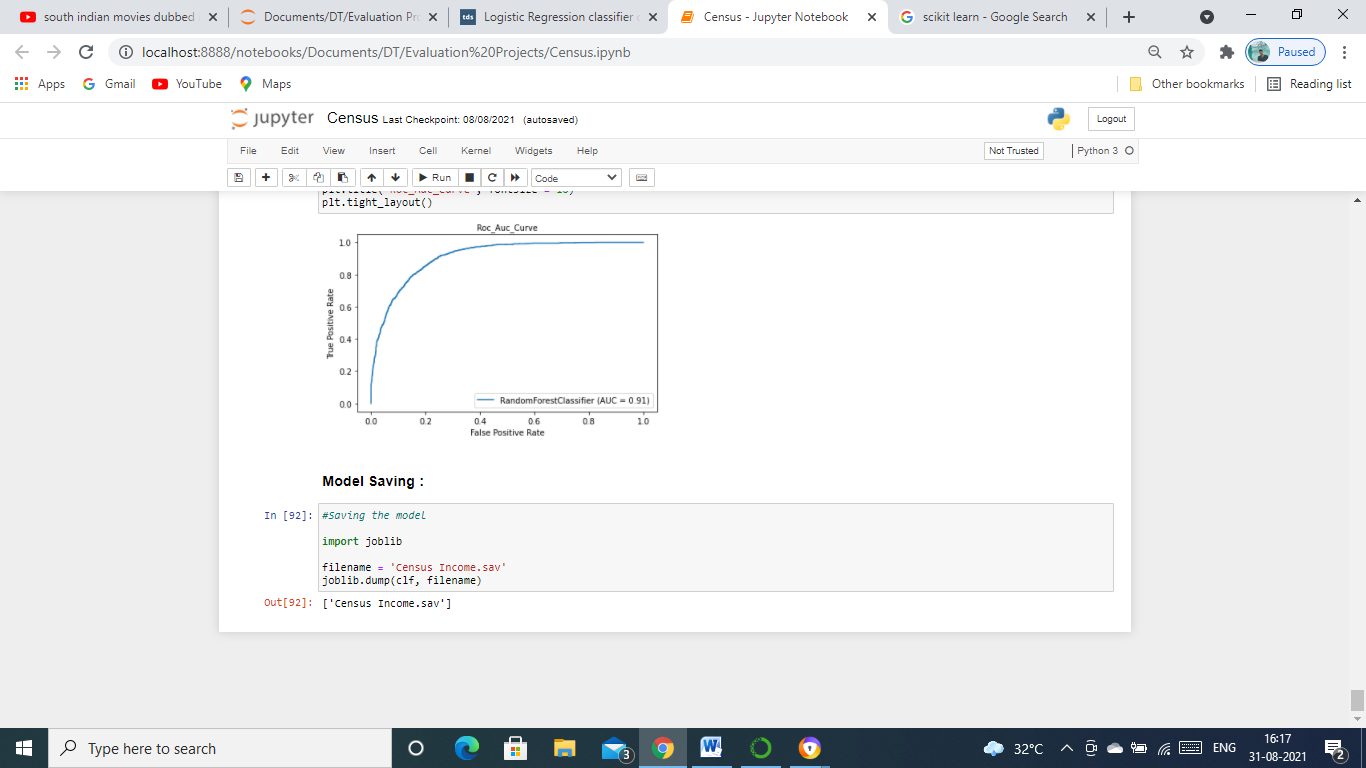
Below are the codes and scores for using all the metrics.



After tuning it with the best parameters we achieved an accuracy score of 86% on this dataset which is better than previously achieved score.

**Saving the model**

We have done all the hard work of creating and testing the model. It would be good if we could save the model for future use rather than retrain it, for this we will first import [Joblib](https://joblib.readthedocs.io/en/latest/) and then save our model.



# Final Remarks

We have learned to build a complete machine learning model. In the process, We visualised, cleaned and built a model which will predict whether a person income is more or less than 50k with an accuracy of 86% for us, taking various independent variables or we can say features as input . We also learned to improve the accuracy score by tuning our model using various parameters and save it for further use.

I hope this blog helped you in anyway and feel free to let me know if there are other changes that could be done to improve the scores and building a better model.

Thank you for reading the blog!