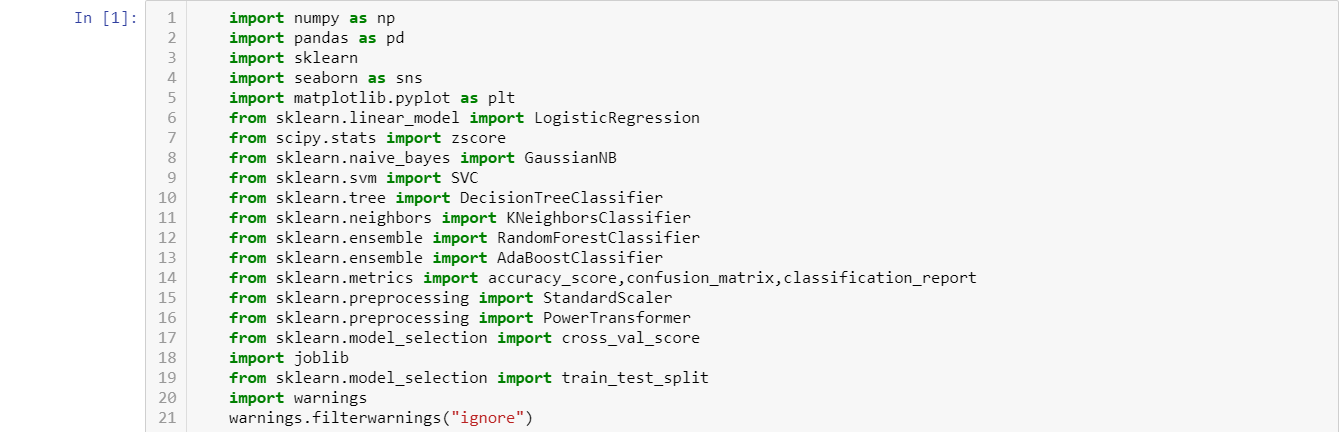
**Census Income Project**

**Defining the problem statement:**

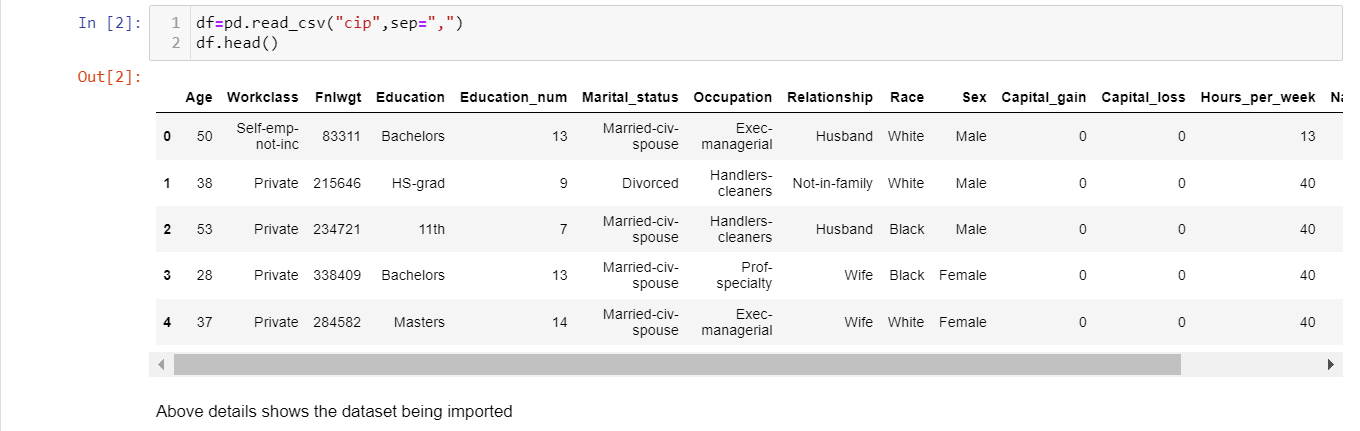
The data contains anonymous information such as age, occupation, education, working class, etc. The goal is to train a binary classifier to predict the income which has two possible values ‘>50K’ and ‘<50K’. There are 32560 instances and 15 attributes in the dataset. The data contains a good blend of categorical and numerical values.

**First, we will import the required libraries.**



**Importing the Dataset:**

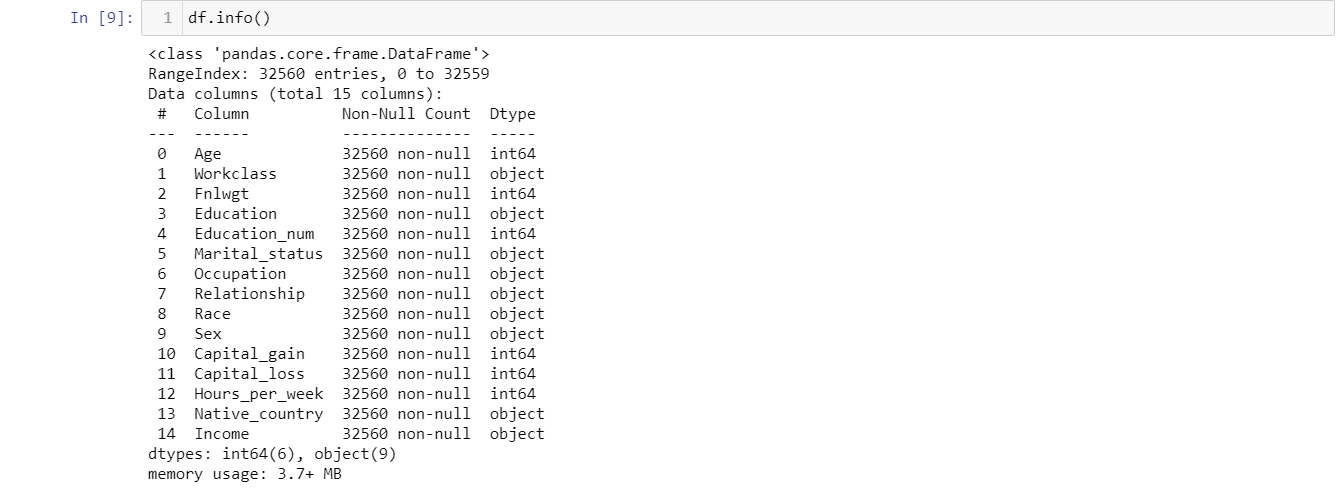
We load the data into a pandas’ data frame using the read\_csv function. There are some whitespaces before and after the data values. To trim all the whitespaces, we use the separator ‘\*, \*’.



Next, we will explore the data. This is an important step before going building the model.

# Exploratory Data Analysis:

# Let us get more information about the dataset using df.info().



Observations

* There are **32560** samples in the training dataset.
* There are both categorical and numerical columns in the dataset.

Let us look the numerical and the categorical data with the help of some visualizations.

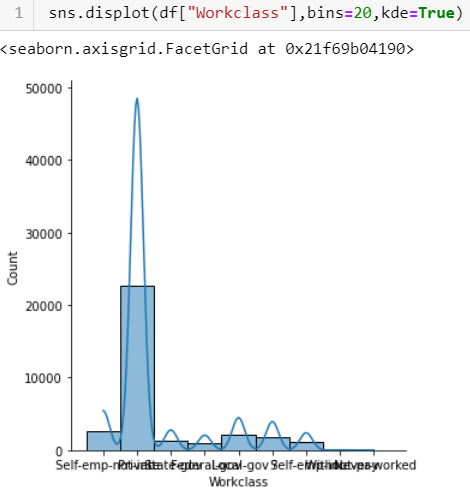
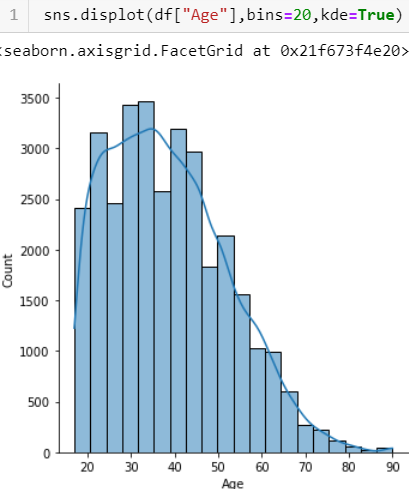
The variables **age**, **hours-per-week** are self-explanatory.

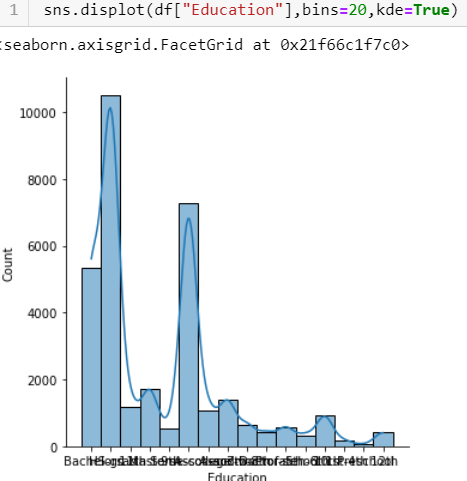
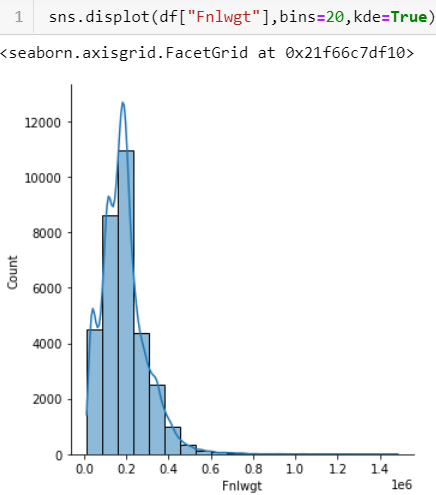
* **fnlwgt**: sampling weight
* **education-num**: number of years of education in total
* **capital-gain/capital-loss**: income from investment sources other than salary/wages.

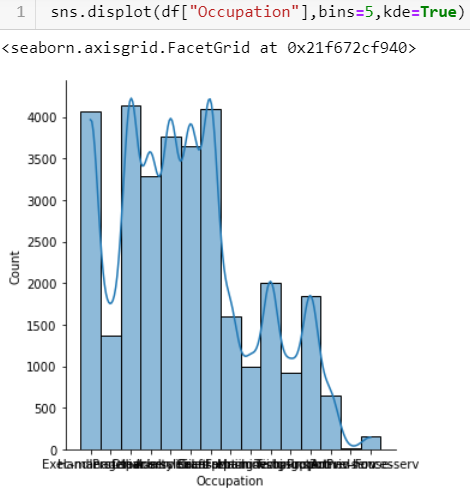
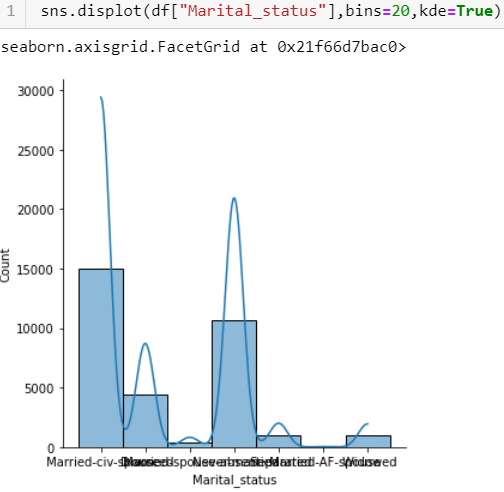
**fnlwgt** is not related to the target variable **income** and will be removed before building the model.

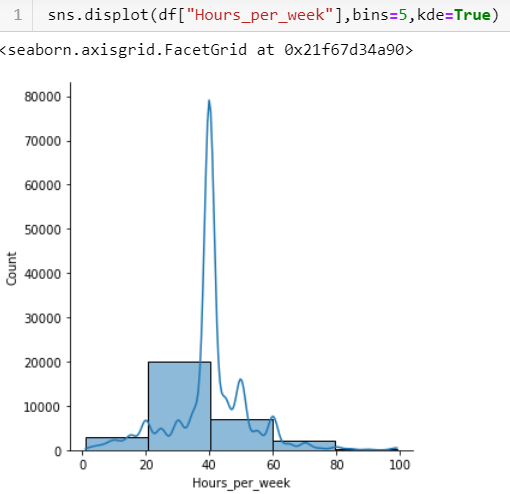
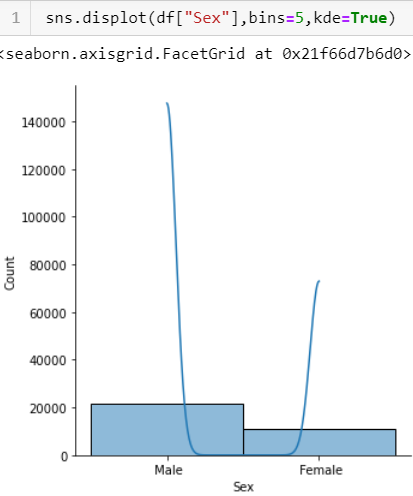
**Data Visualizations:**

**The below figures shows the distribution plots of few columns with respect to target variable.**

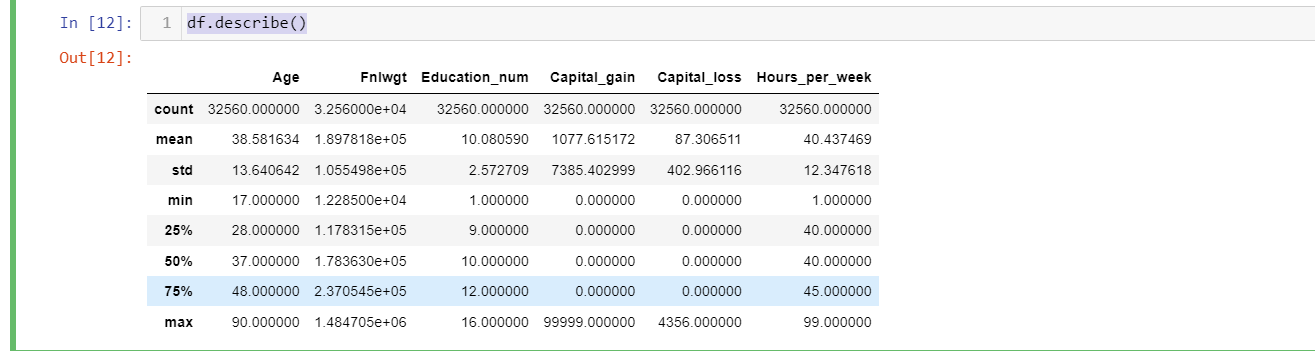








More information about the data can be gathered by using df.describe().

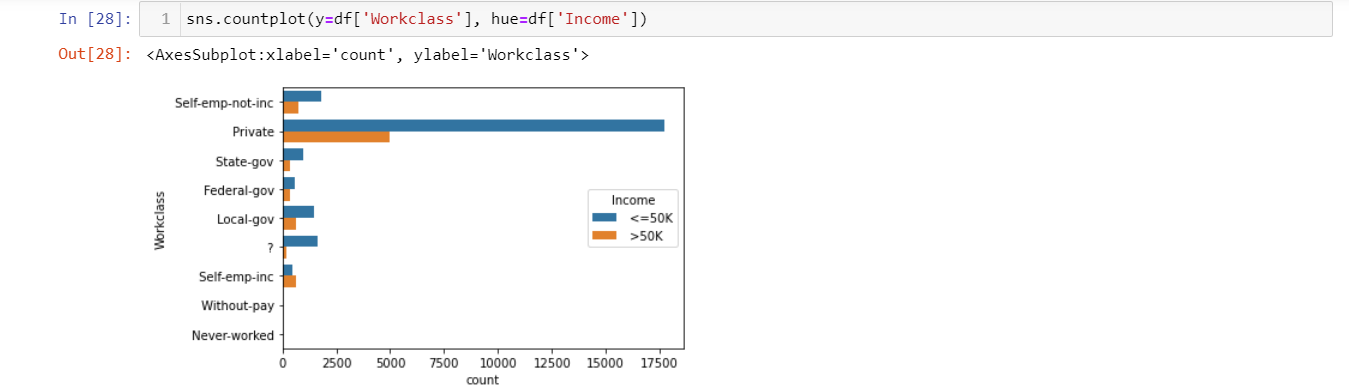


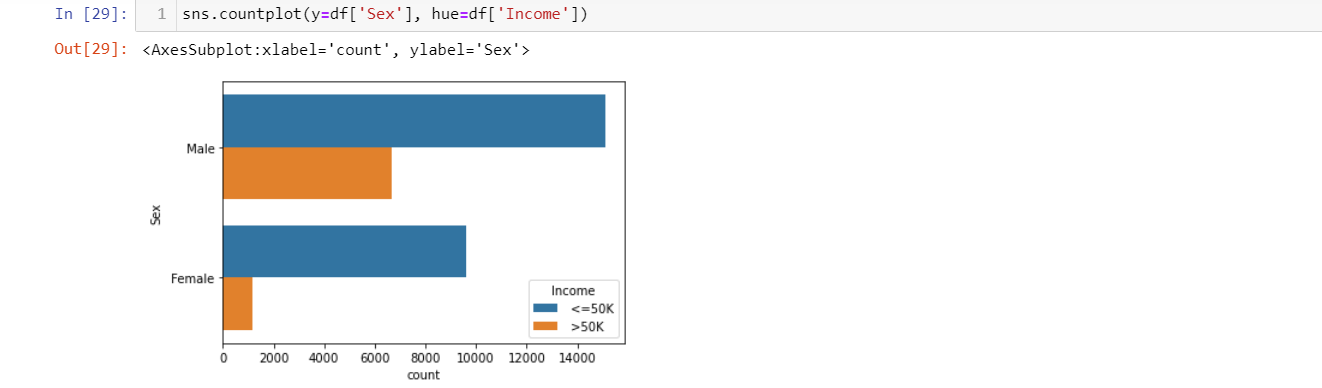
**Observations:**

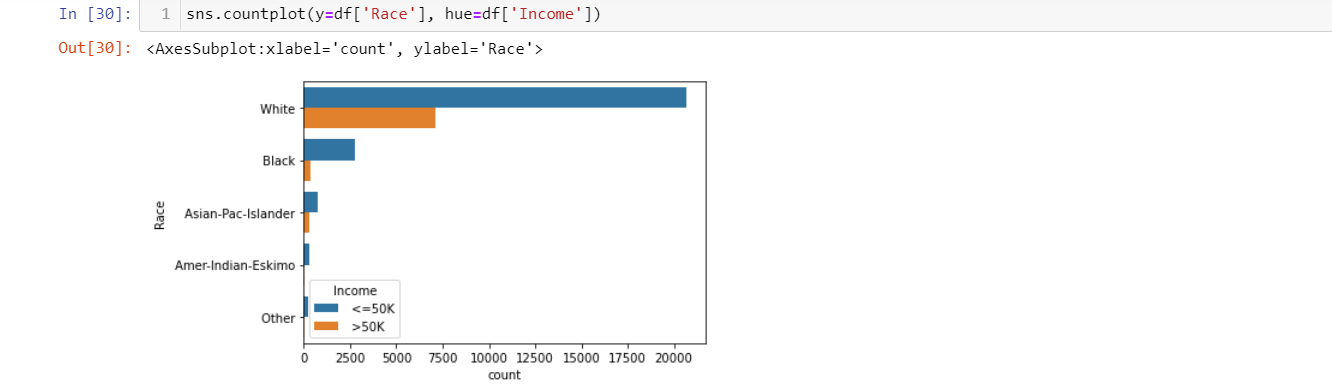
* None of the numerical attributes have missing values.
* The values are on different scales. Many machine learning models require the values to be on the same scale. We will use [StandardScaler](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html) from the sklearn library to scale the features.

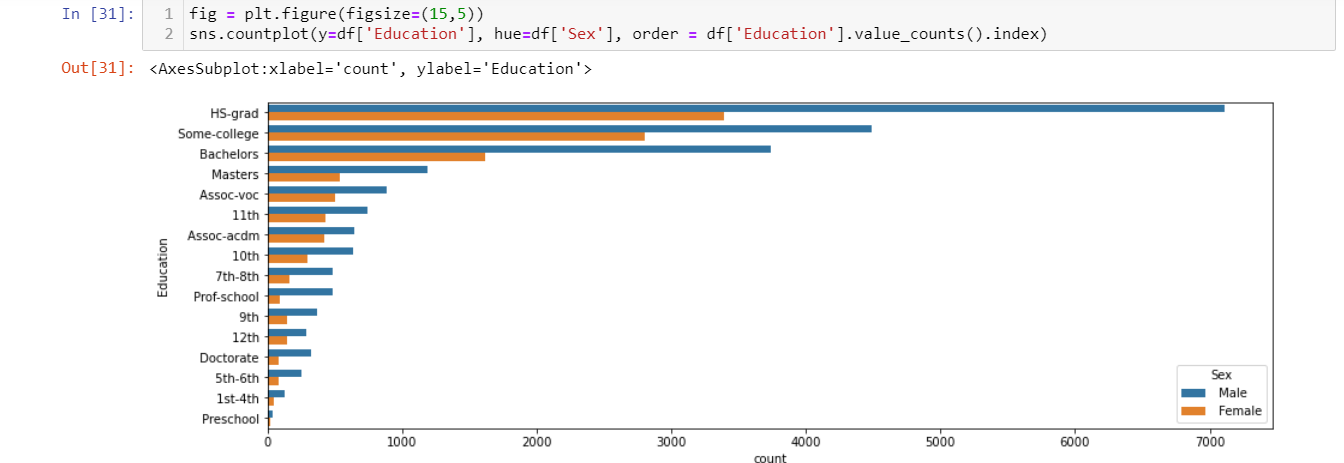
**Data Visualization:**

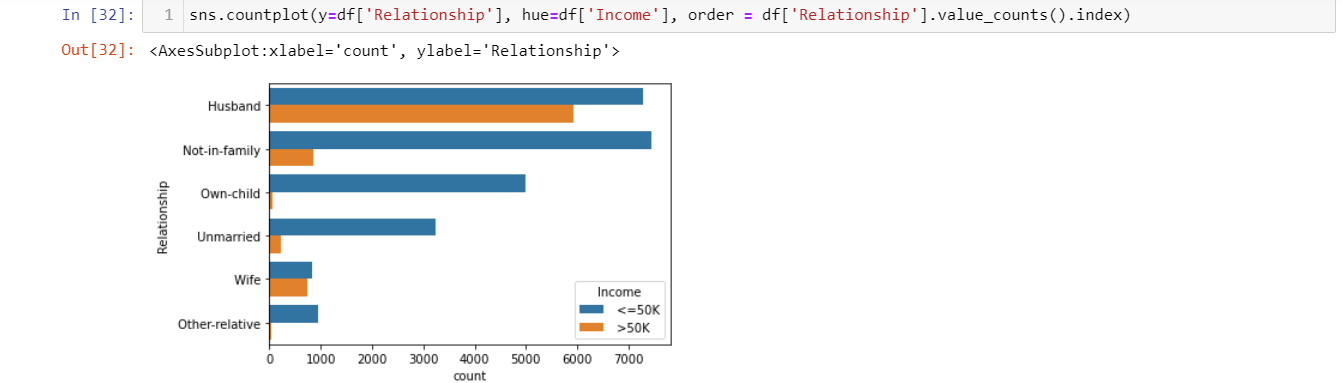
We will use count plot from the seaborn package.

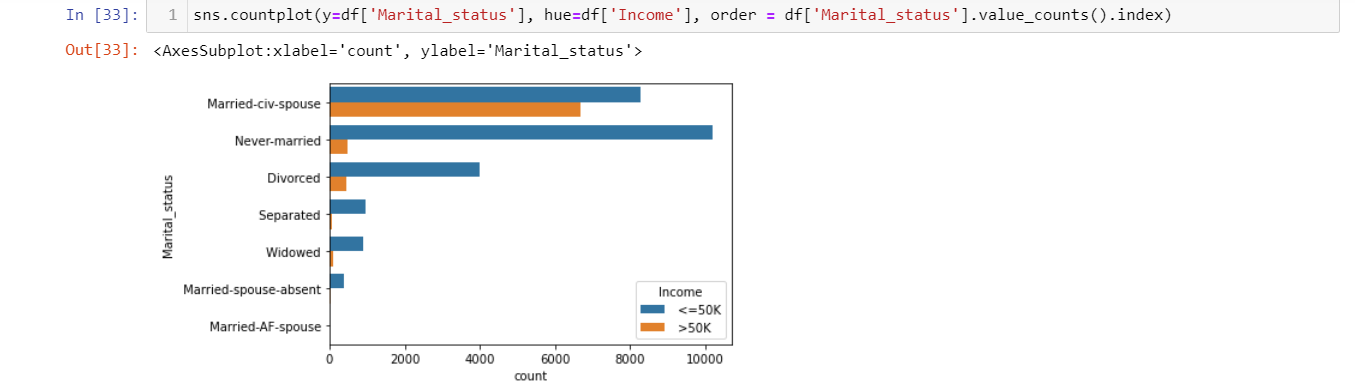


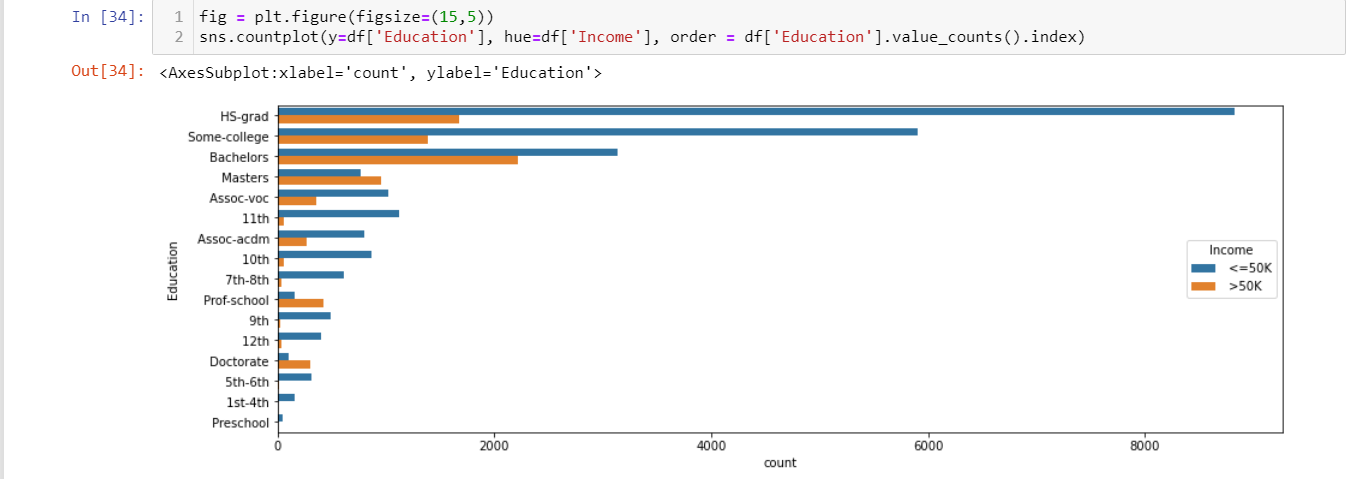












# Creating Pipelines:

# Ordinal Encoding-

To convert categorical text data into model-understandable numerical data, we use the Ordinal Encoder class. So, all we have to do, to Ordinal encode a column is import the Ordinal Encoder class from the sklearn library, fit and transform the column of the data, and then replace the existing text data with the new encoded data.

# 

# We have converted all categorical data into integer and float datatype.

# Using Z-score method to remove Outliers-

# 

# Using Z-score we have removed outliers in the dataset, and we can see total data loss after removing outliers is 12.19%.

# Skewness-

# 

# As we can see skewness range in the data set is beyond +0.50 to -0.50 so, we must use transformation methods to get the data within the skewness range.

# Divide the data set into test and train:

Now that all our data is numerical after Ordinal encoding, so we split the data into test and train and drop the Income column from the test set because we must predict the Income with our test data set.

# 

# Power transformation:

# 

# Using power transform method, we have skewed the data and got it within the skewness range.

# Scaling:

# StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance. This method removes the median and scales the data in the range between 1st quartile and 3rd quartile.

# 

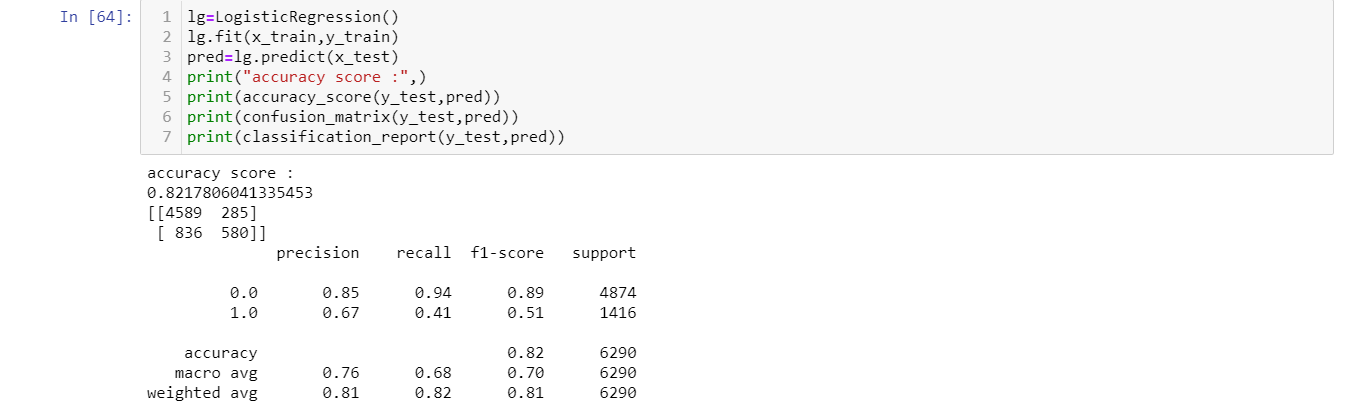
# Train-Test spit:

# 

**Building Machine Learning Models:**

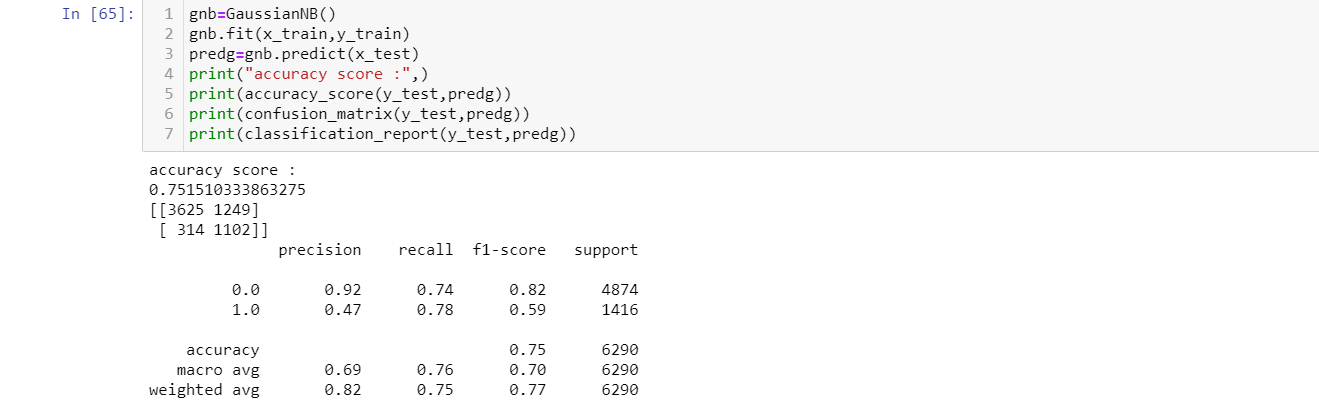
The goal in this step is to develop a benchmark model that serves us as a baseline, upon which we will measure the performance of a better and more tuned algorithm. We are using different Regression & Classification Techniques and comparing them to see which algorithm is giving better performance than other and at the end we will combine all of them using Stacking and see how our model is predicting.

1. **Logistic Regression:**



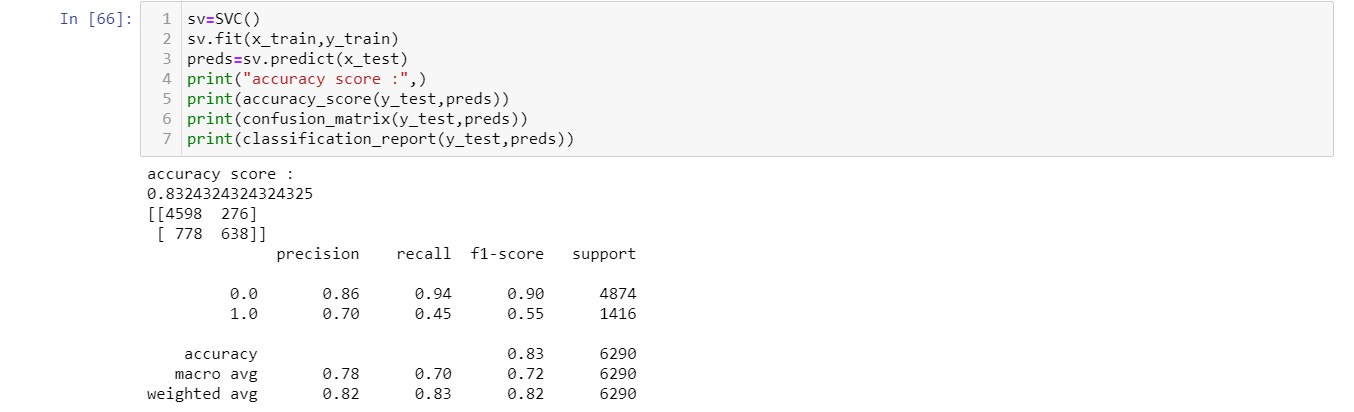
From above we come to know that Logistic Regression score is 82.17%.

1. **GaussianNB:**



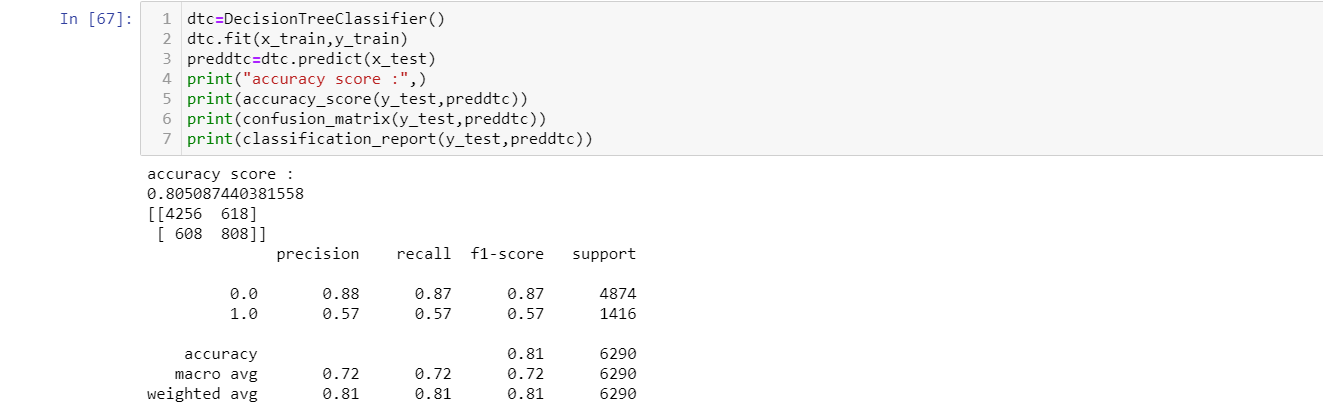
From above we come to know that GaussianNB score is 75.15%.

1. **Support Vector Classifier:**



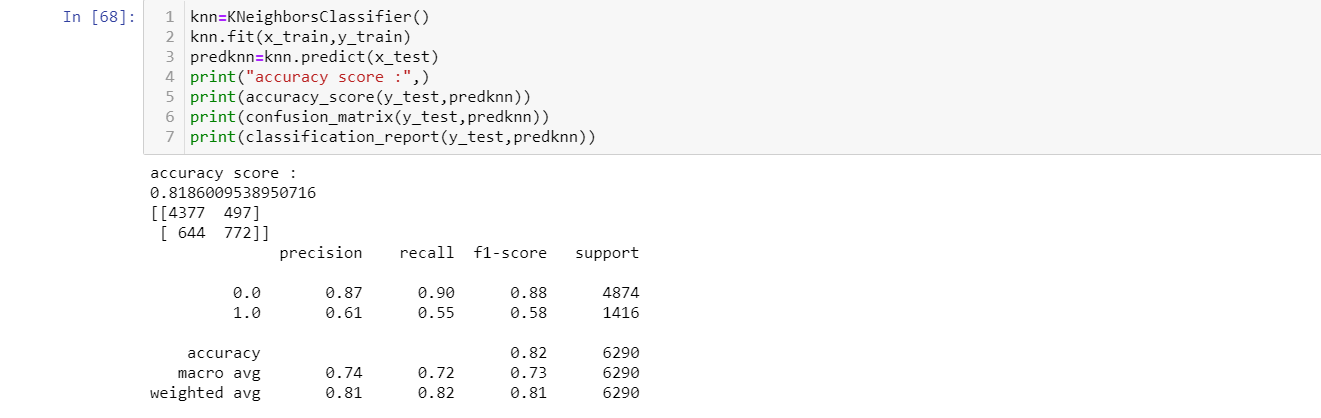
From above we come to know that Support Vector Classifier score is 83.24%.

1. **Decision Tree Classifier:**



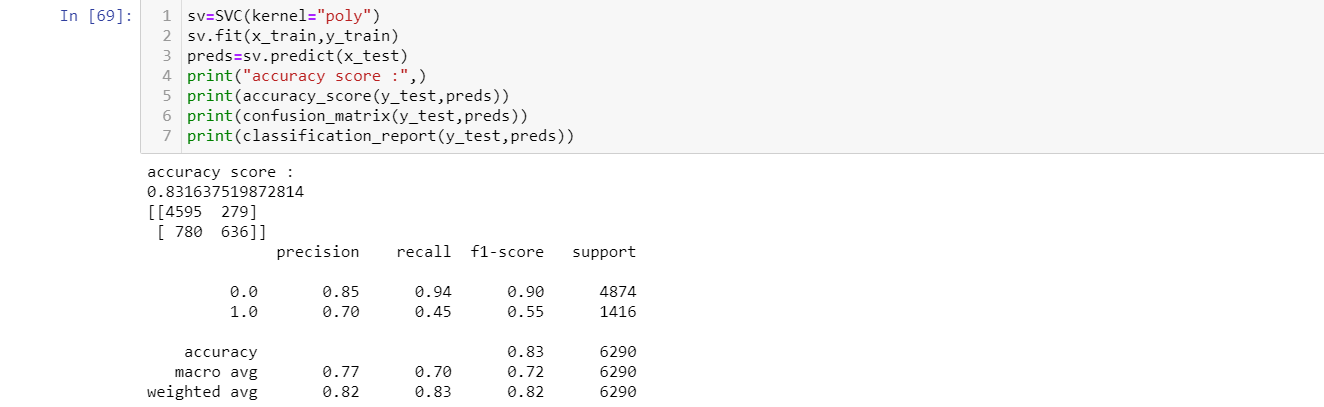
From above we come to know that Decision Tree Classifier score is 80.50%.

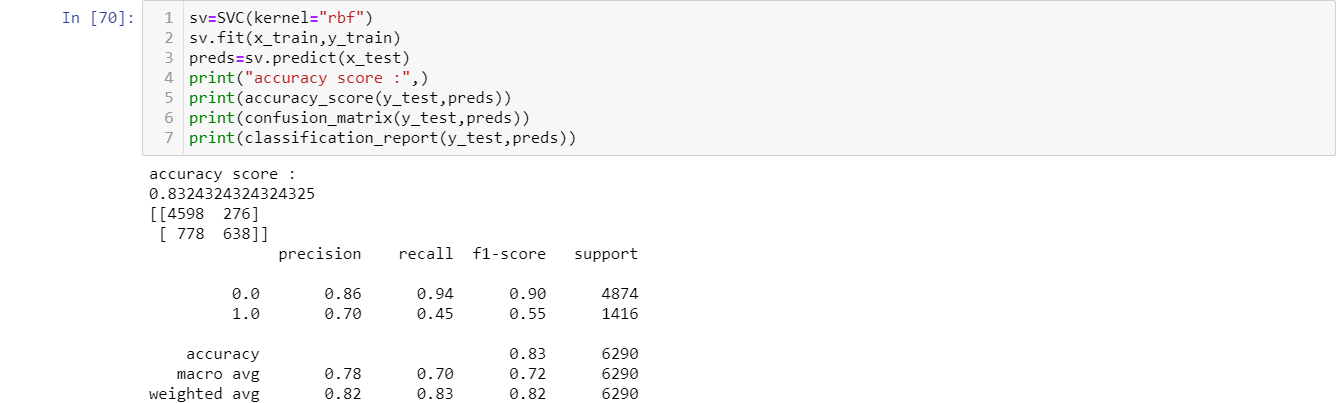
1. **KNeighbors Classifier:**



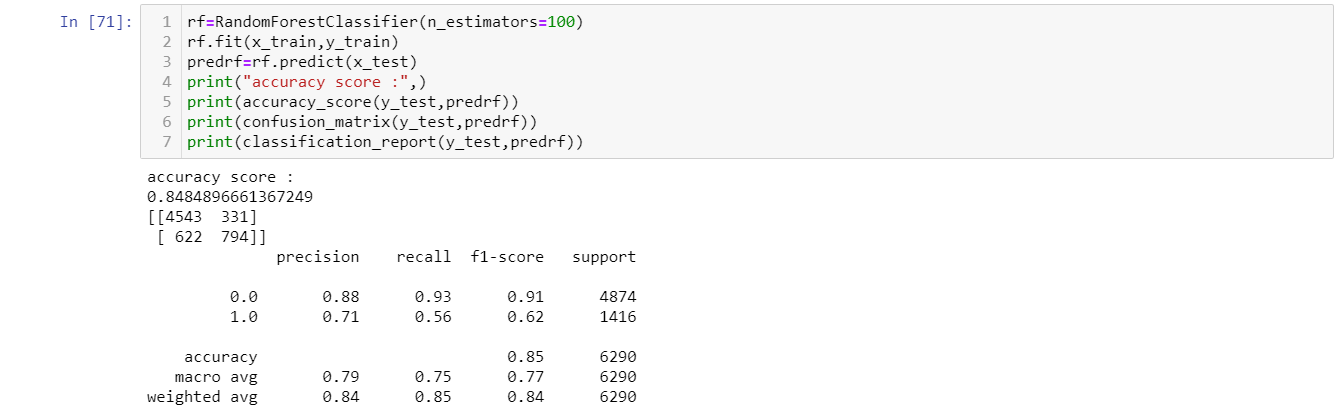
From above we come to know that KNeighbors Classifier score is 81.86%.

1. **Support vector classifier with different kernel:**



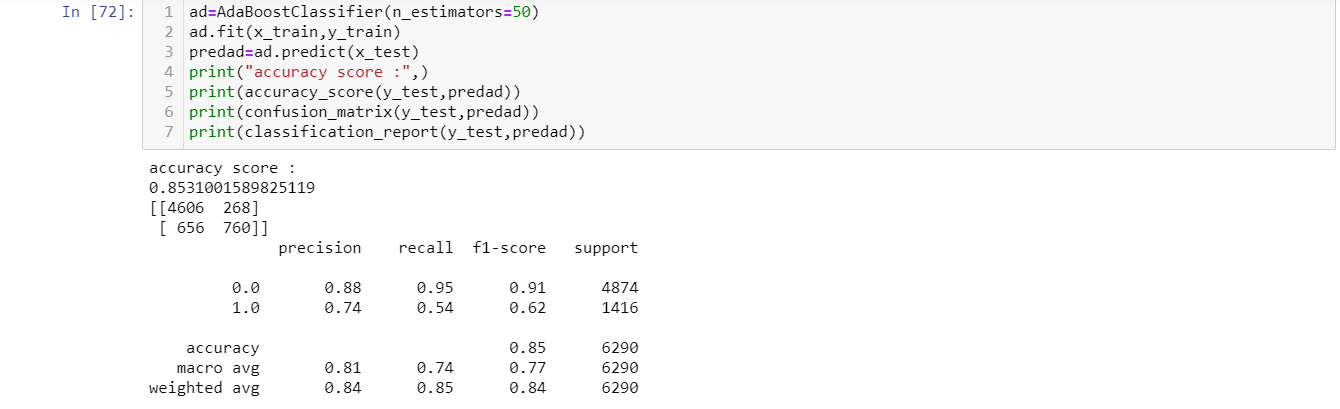


1. **Ensemble Methods:**
   1. **Random Forest Classifier:**



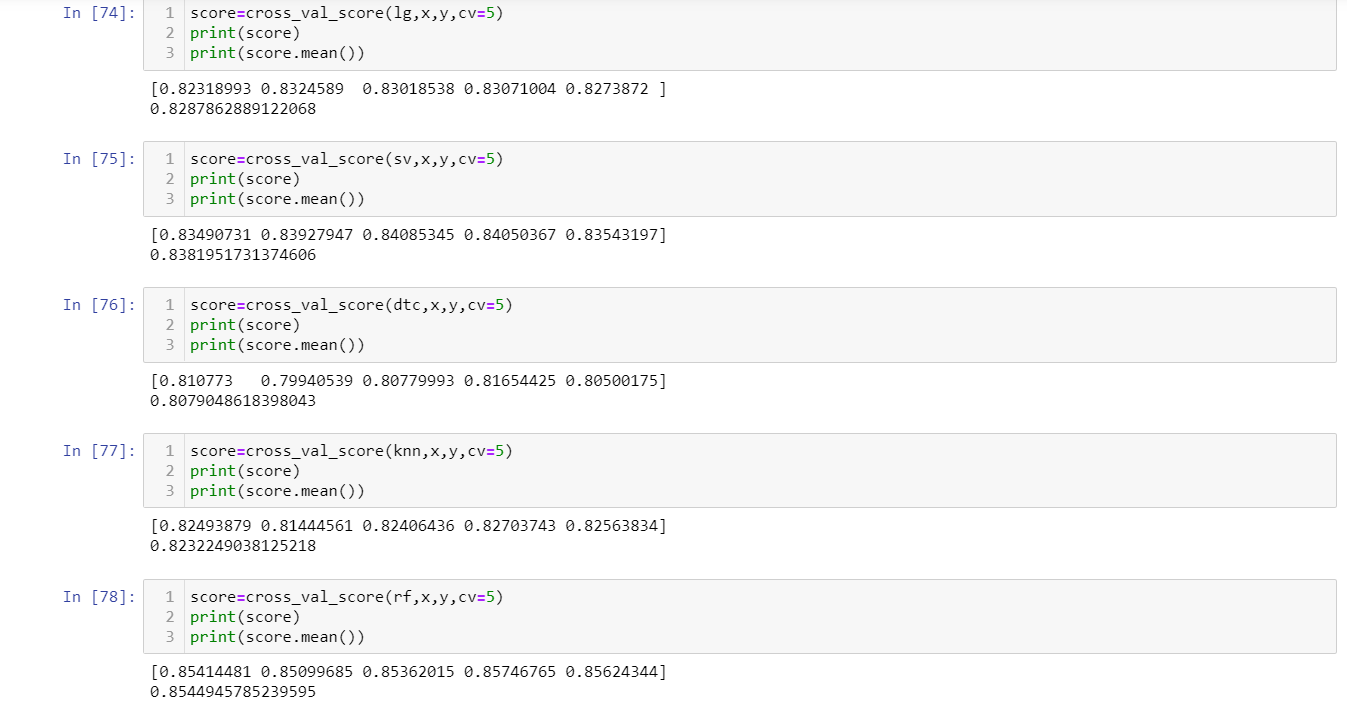
From above we come to know that Random Forest Classifier score is 84.84%.

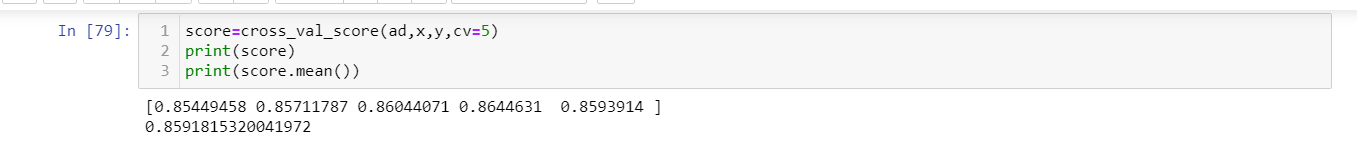
* 1. **AdaBoost Classifier:**



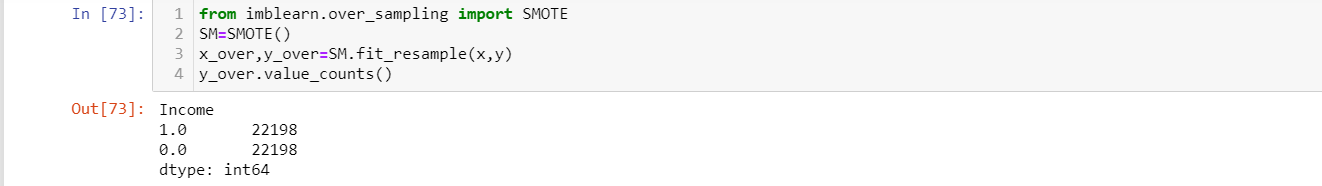
From above we come to know that AdaBoost Classifier score is 85.31%.

**Cross Validation of Build Models:**





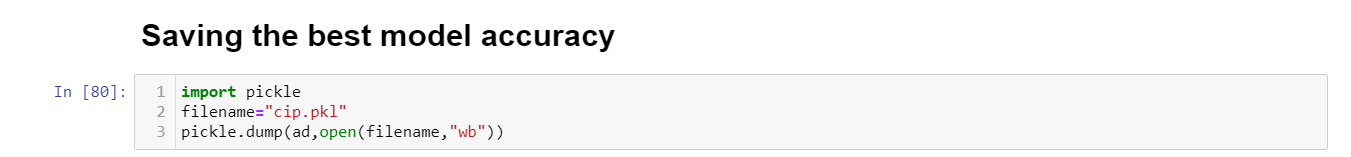
**Balancing the target variable using SMOTE method:**



As we see Target variable is equally balanced now.

**Saving the best model accuracy:**

As we can find from above Build model AdaBoost Classifier is performing well and has good accuracy score with cross validation. So, we are going to save AdaBoost Classifier as the best model.



**Conclusion:**

In this type of problem Feature Engineering is the most crucial think. You can see how we have handled the categorical and numerical data and how we build different ML model on the same dataset. We also check the accuracy score of each model so that we can understand how it should perform in our test dataset. At last, you can also further improve the Model by Tunning different parameters which are being used in the model.

**Loading the model from the pickle:**

