IDS PROJECT

***CENSUS INCOME (ADULT) DATASET***

***GROUP MEMBERS***:-

Himanshi Sharma (21ucc052)

Vedant Gholap (21ucc112)

Arya Panchlinge (21ucc024)

# PROJECT OBJECTIVES:

Applying ML Classification algorithms on the data set and getting inferences from the data.

# SOURCE OF DATASET:

<https://archive.ics.uci.edu/dataset/2/adult>

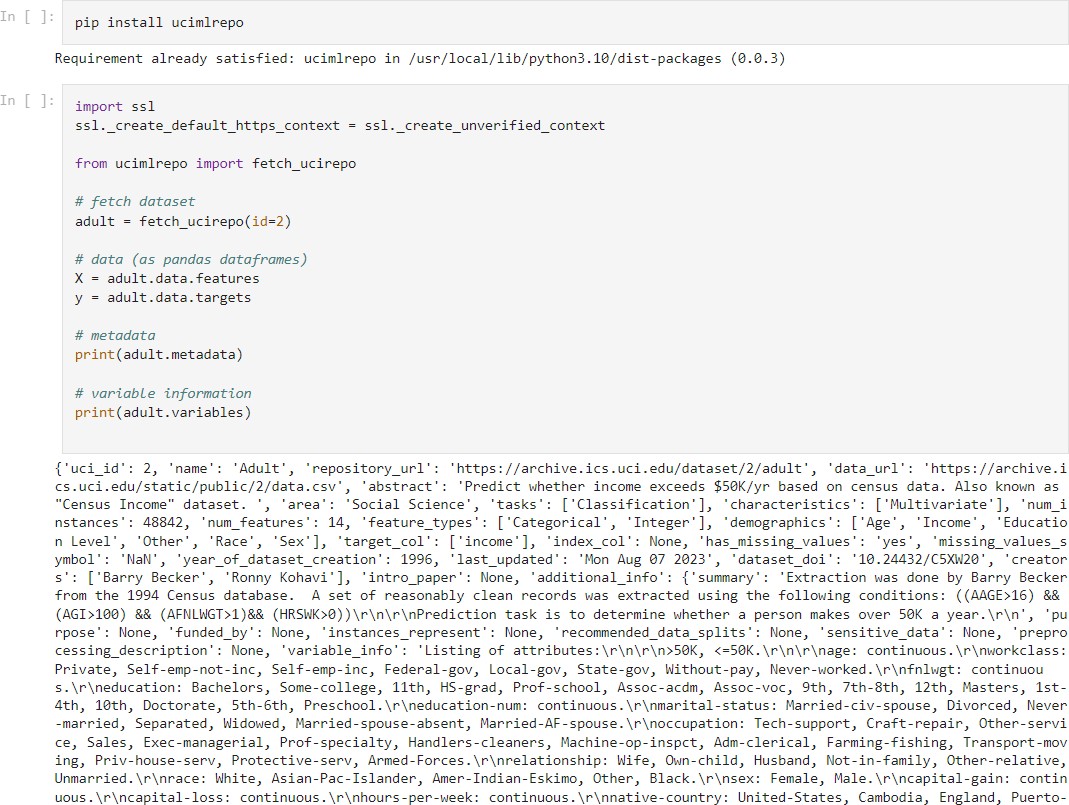
# ABOUT THE DATASET:

The dataset contains information from the 1994 census such as age, education, marital status, occupation, native country, income, etc. The task is to predict whether income exceeds $50K/year based on census data.

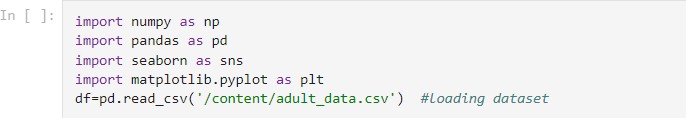
# DATA DESCRIPTION:

* Dataset Characteristics: Multivariate
* No. of attributes:15
  + Age
  + Workclass
  + Fnlwgt
  + Education
  + Education-num
  + Marital-status
  + Occupation
  + Relationship
  + Race
  + Sex
  + Capital-gain
  + Capital-loss
  + Hours-per-week
  + Native-country
  + Income
* Attribute Characteristics: Integer, Categorical, Binary
* No. of instances: 48842 (rows)
* Presence Of Missing Values: True

# IMPLEMENTATION:



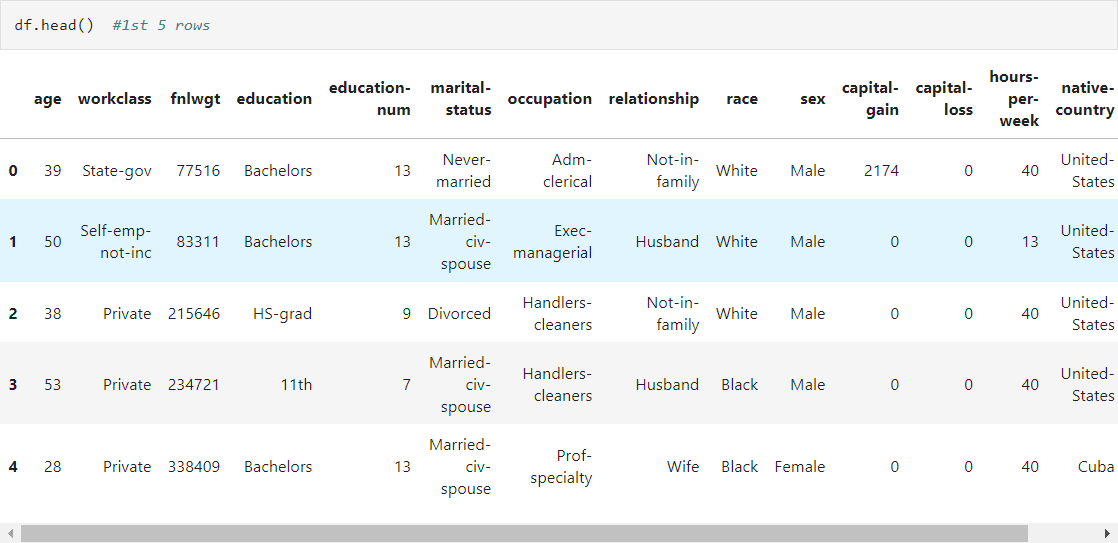
First, we imported data from the UCI dataset website and saved it in a CSV file.



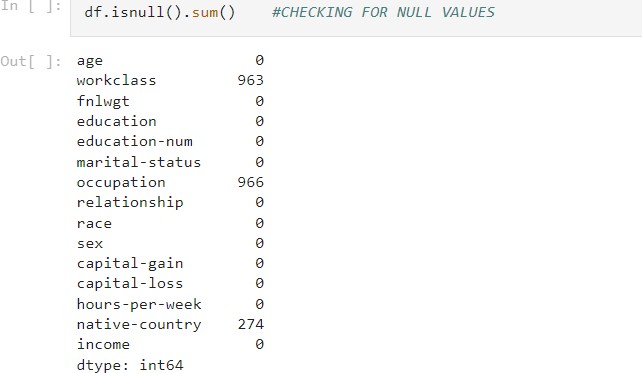
All important libraries such as Numpy, Pandas, etc are imported.

Numpy is used to perform a wide variety of mathematical operations on arrays. Pandas is useful for data manipulation and analysis.

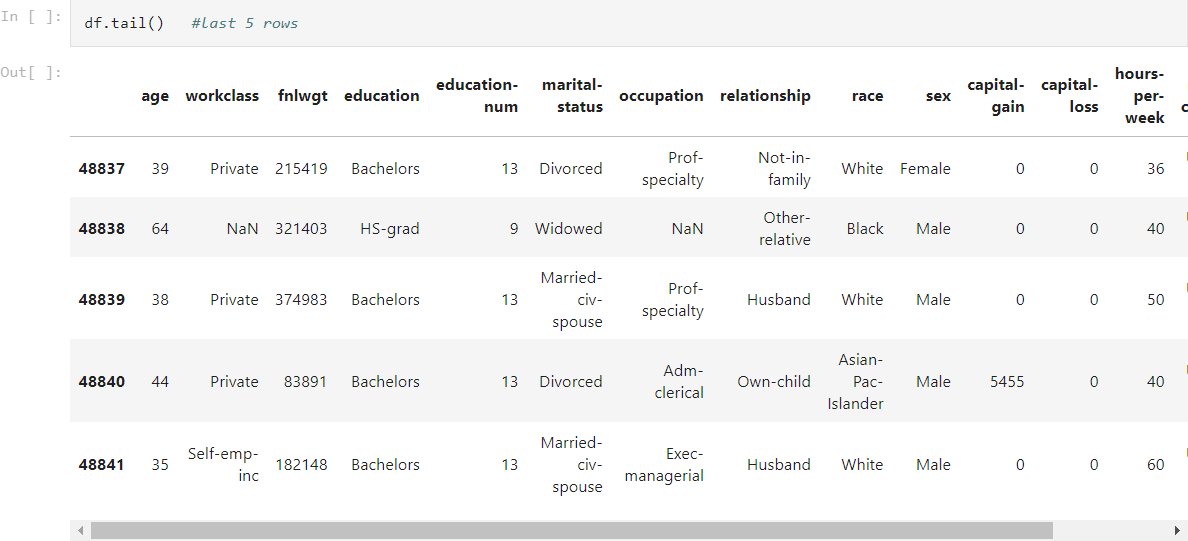
Seaborn is useful for making statistical plots/graphics in Python. Matplotlib is useful for making static and interactive visualizations. Through pd.read\_csv the data is loaded into df dataframe.



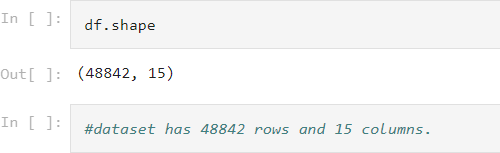
To see 1st 5 rows we used df.head() command.



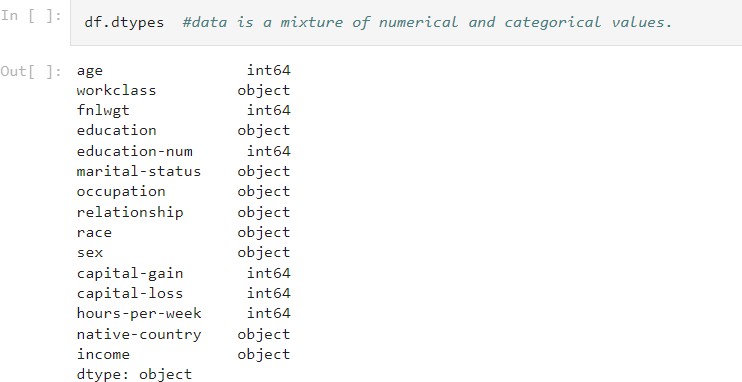
We used df.isnull().sum() to get the count of null values in all columns.



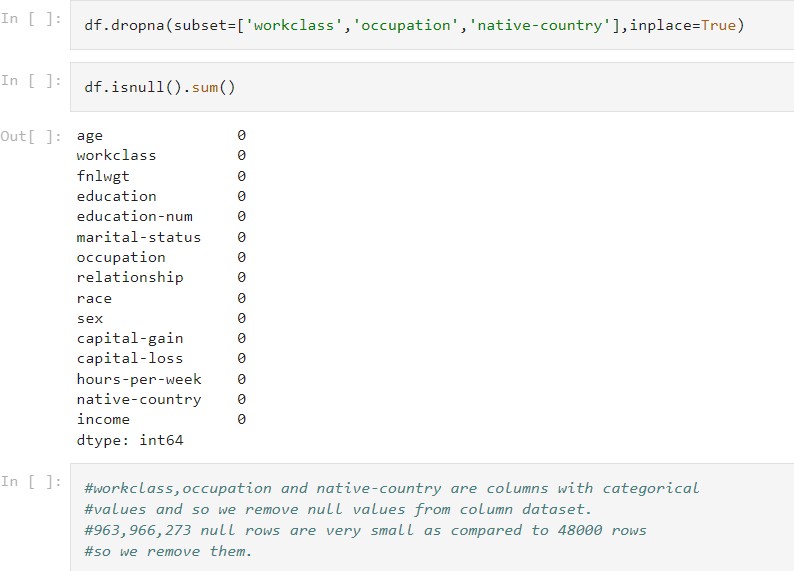
To see the last 5 rows we used df.tail() command.



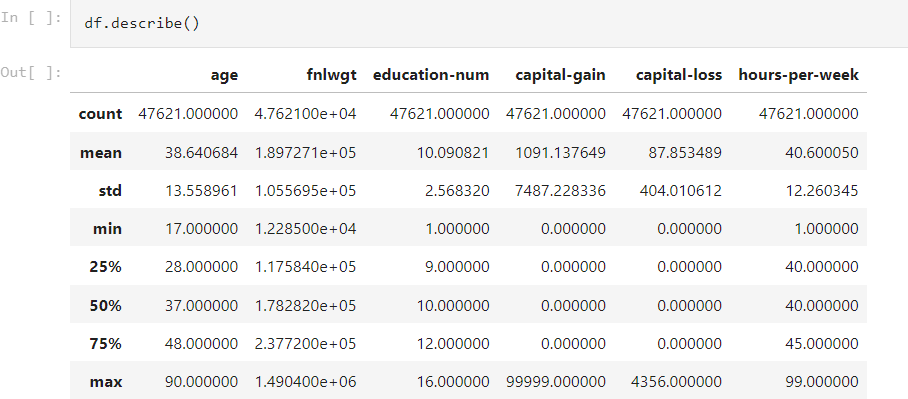
df.shape() return no. of rows and columns in the dataset.



df.dtypes is used to get the datatype of all columns.



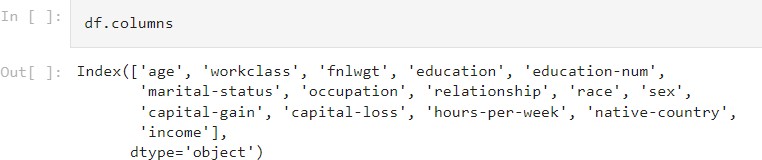
**Data Preprocessing:** Since the columns work-class, occupation and native- country had null values and all these columns contain categorical values, we drop those rows from the dataset.

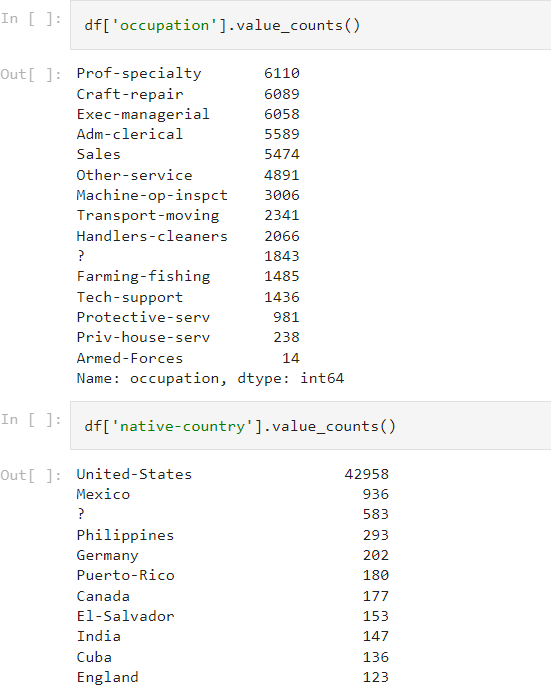
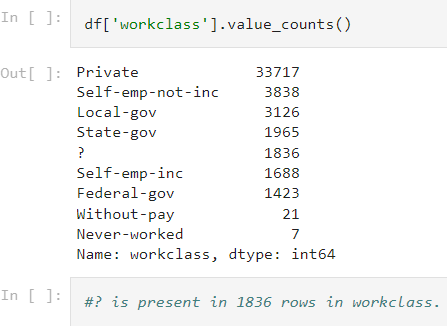


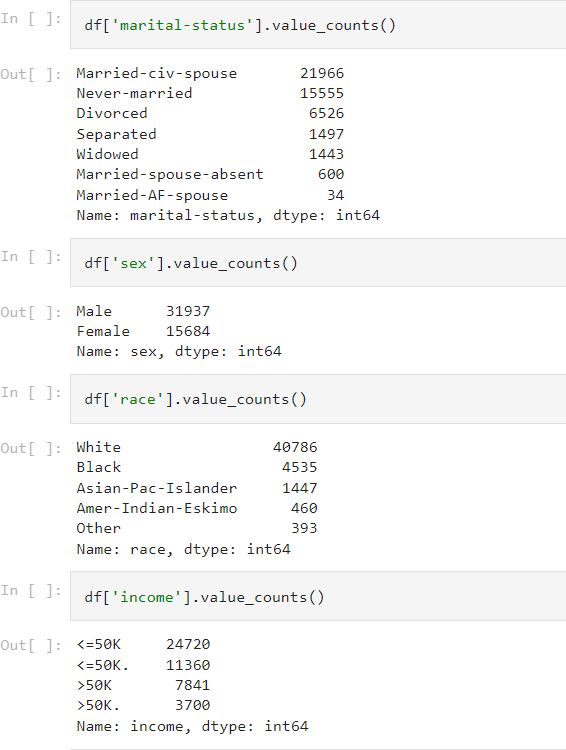
df.describe() gives count, mean, standard deviation, minimum, 1st

quartile(25%), median, 3rd quartile(75%) and maximum values in the numerical columns of dataset.

Also, while going through the dataset we found ‘?’ values in some columns of the dataset which need to be preprocessed.







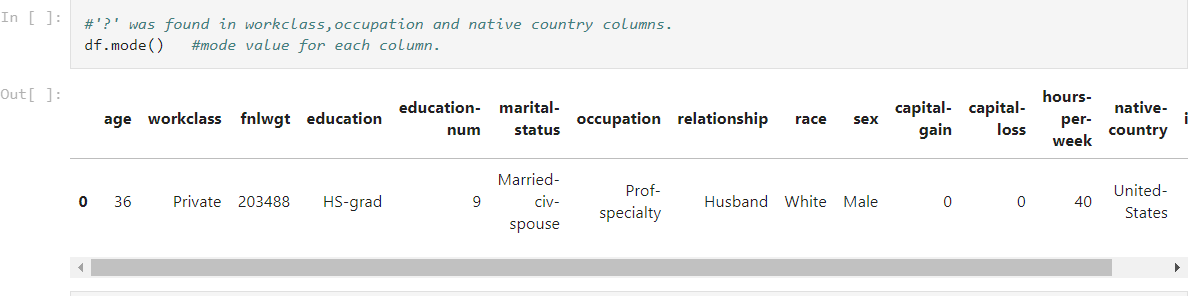
We can get all column header names by df. columns.

df[<column-name>].value\_counts() is used to get different values in columns along with their counts.

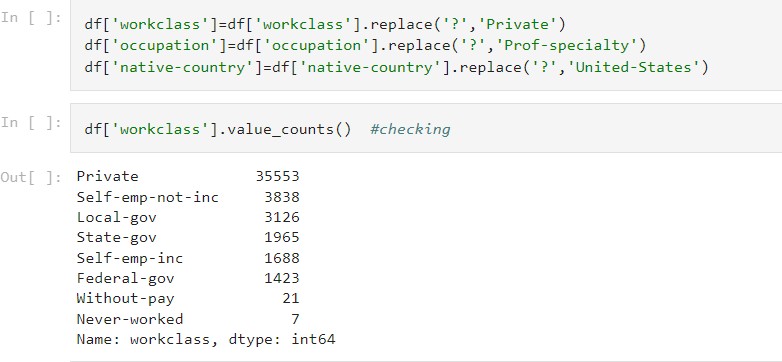
We checked for ‘?’ for this in all columns.

‘?’ is found in workclass, occupation, and native-country columns. Also, they are found in large numbers so their values need to be imputed by some suitable values. For categorical variables, mode is the best value for

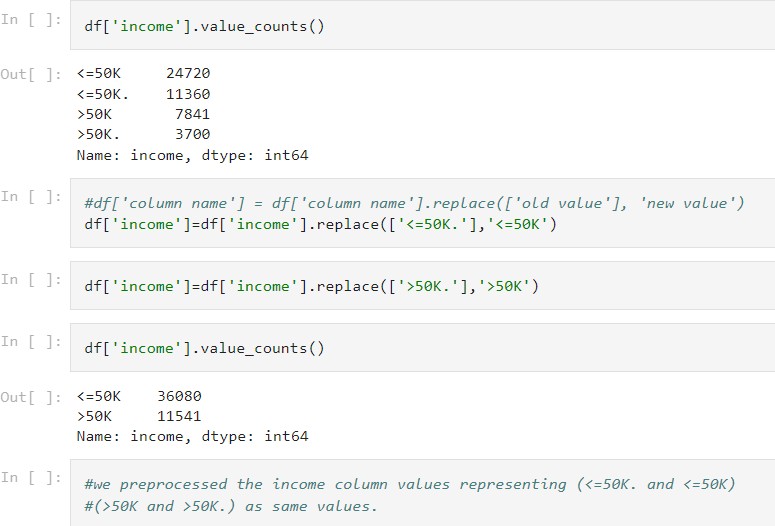
replacing/missing values.



We found the mode for all columns and replaced all ‘?’ by the corresponding column’s Mode values.

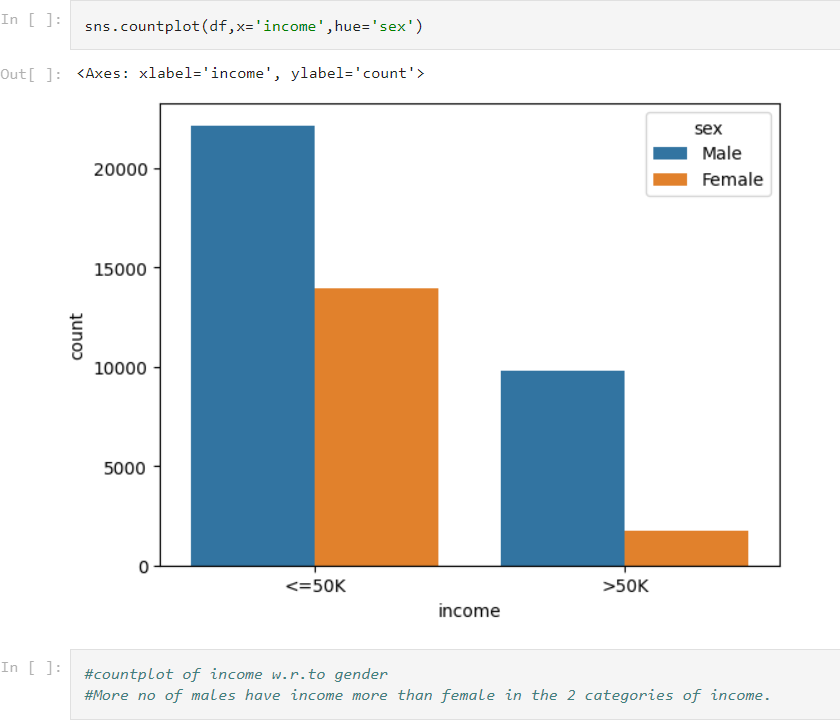


In the income column, we found 4 different options that needed to be corrected, So we used df[‘income’].replace() function to correct them.

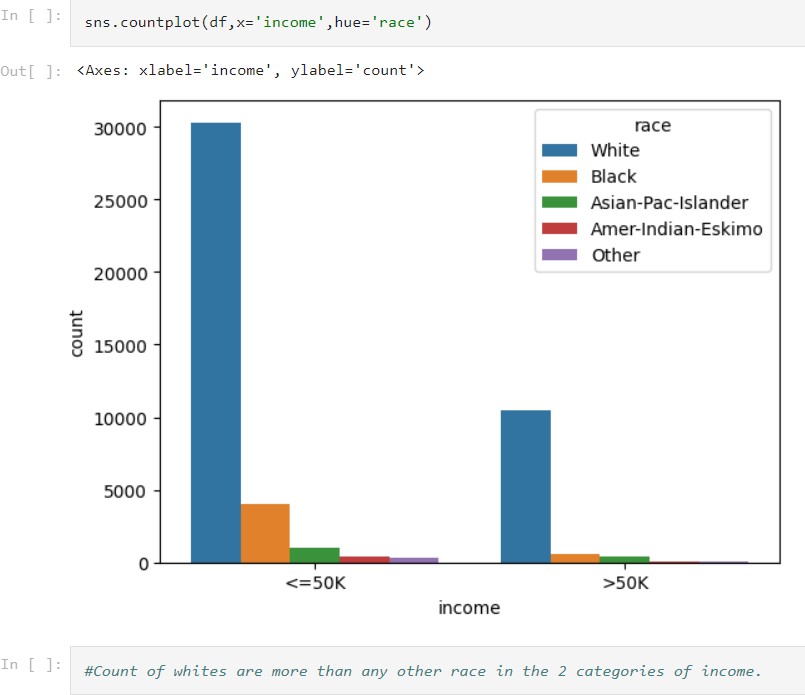


Above we replaced <=50k. by <=50k and >50k. with >50k Our data preprocessing step is completed.

# Now we perform Data Analysis And make many visualizations.

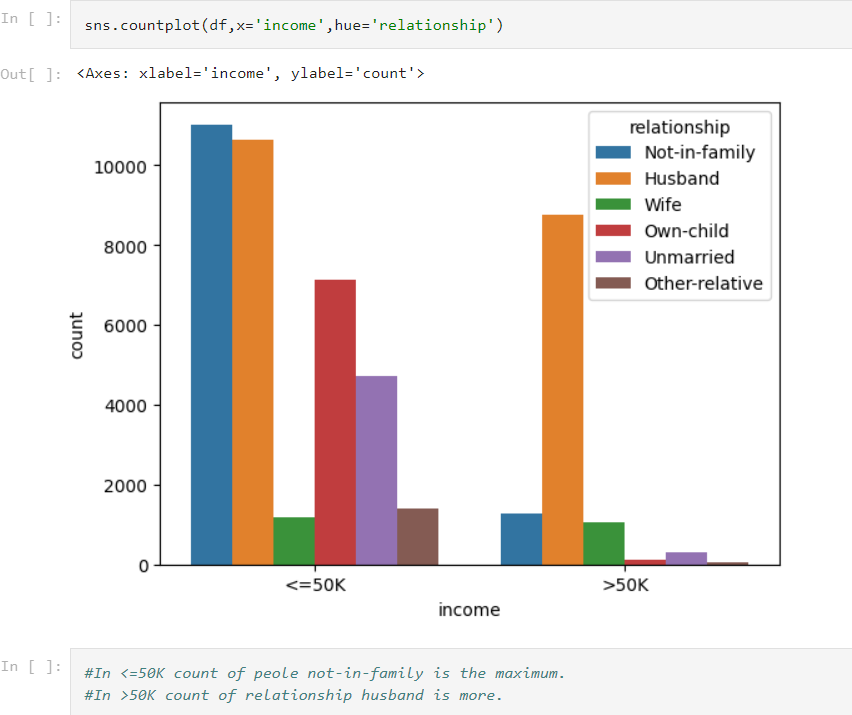


The countplot shows the count of males & females with respect to different income levels. Males have more income as compared to females.



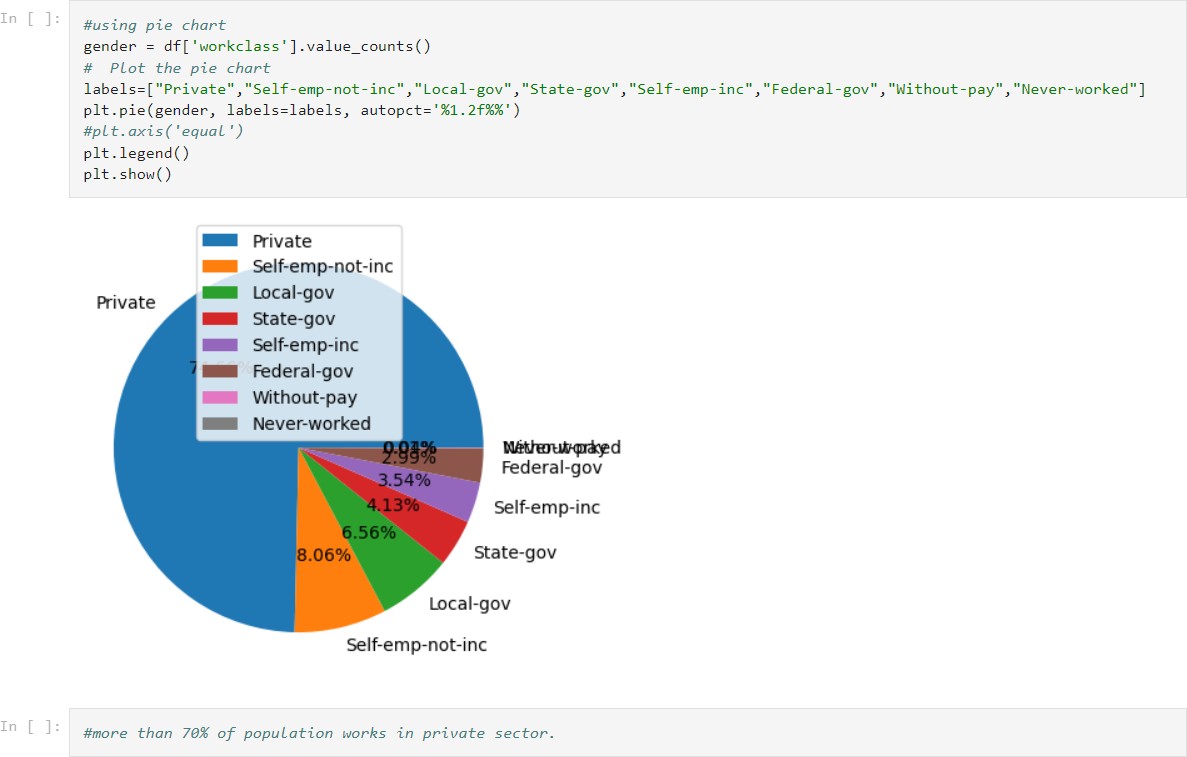
The countplot shows the count of people of different race vs categories of income.

While raced people have the highest count in 2 categories of income.



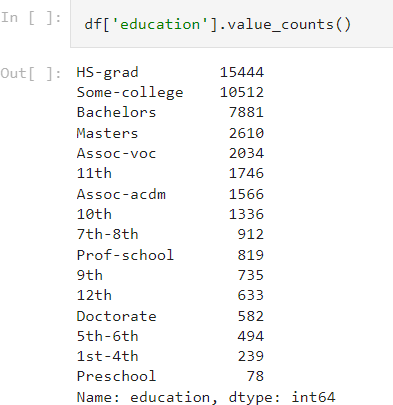
The count plot shows the count of people in different relationships vs income.

In <=50k level Not in family has the highest frequency and for >50k level of income Husband has the maximum frequency.

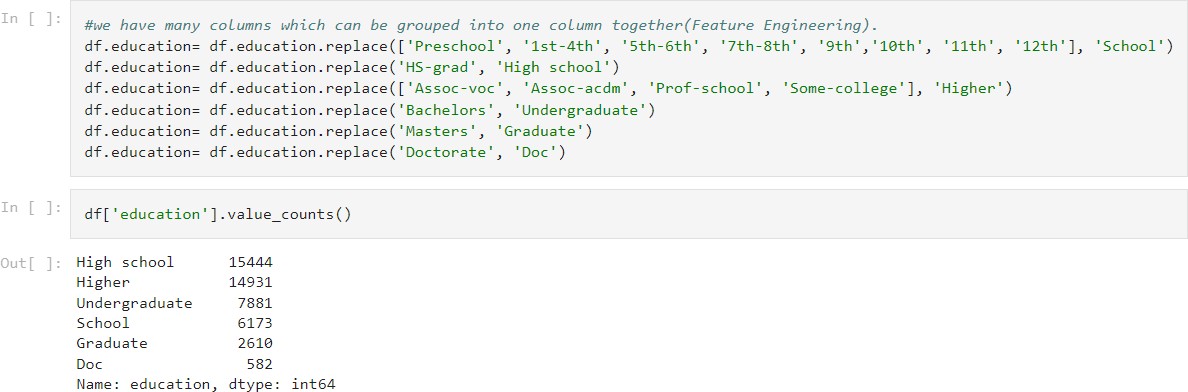


The pie chart shows the distribution of the population in the work class column. We find that more than 70% of the population works in the private

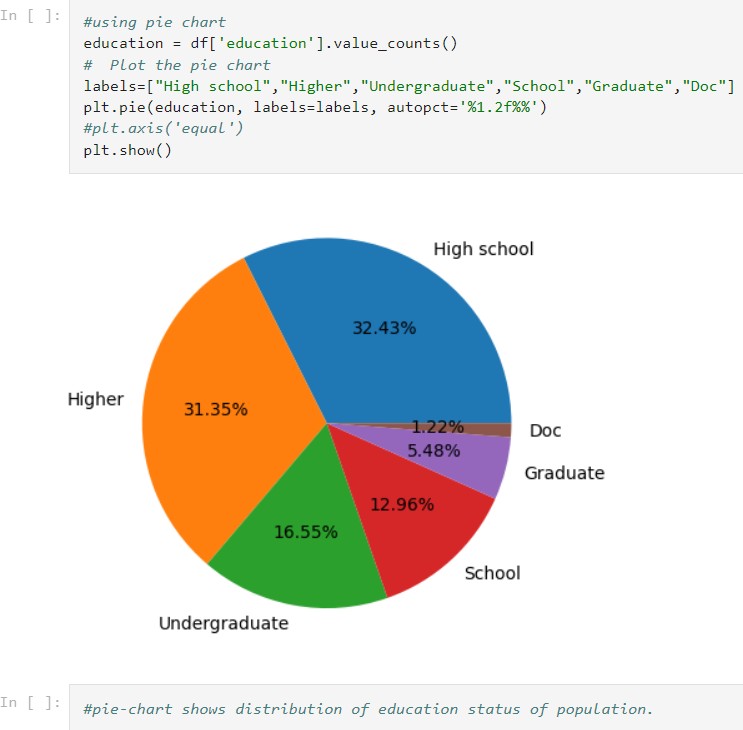
sector. Nearly 13% of people work in local-gov, state-gov or federal-gov sectors.



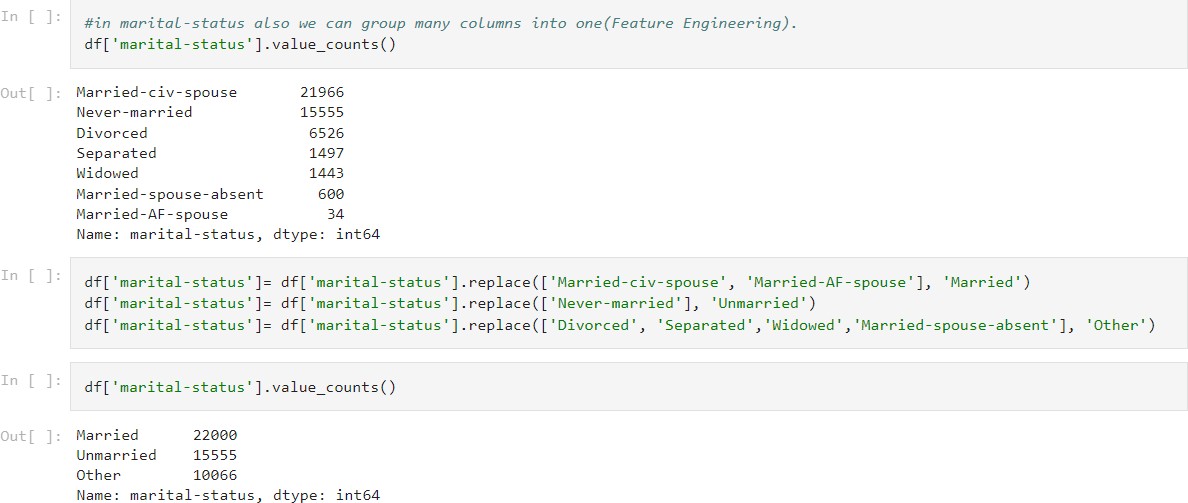
Now we want to see distribution in the education column but it has many values so we group many values into one group only so that distribution and analysis are easily understandable.



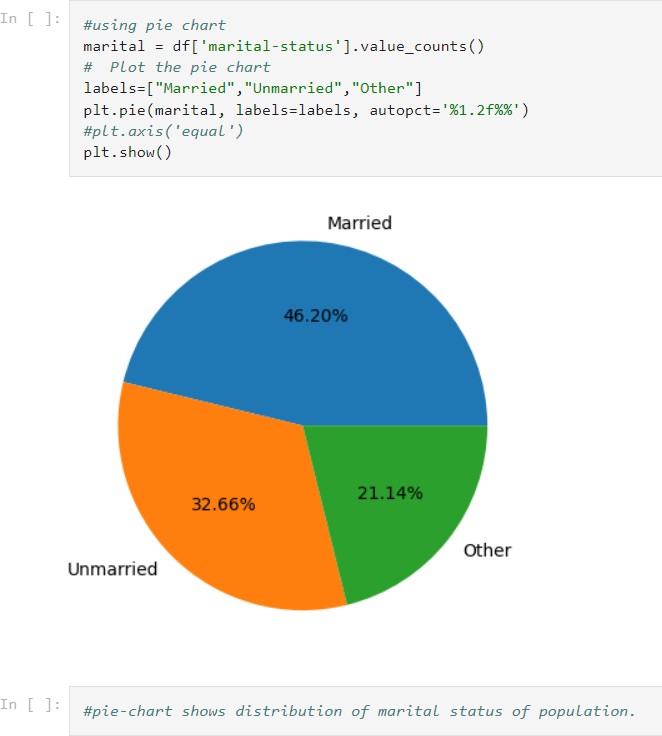
We combined/replaced pre-school,1st-4th,5th-6th,7th-8th,9th-10th,11th,12th into one variable of ‘school’. HS-grad is replaced with ‘High school’. Other columns are also combined or their names are changed/replaced.



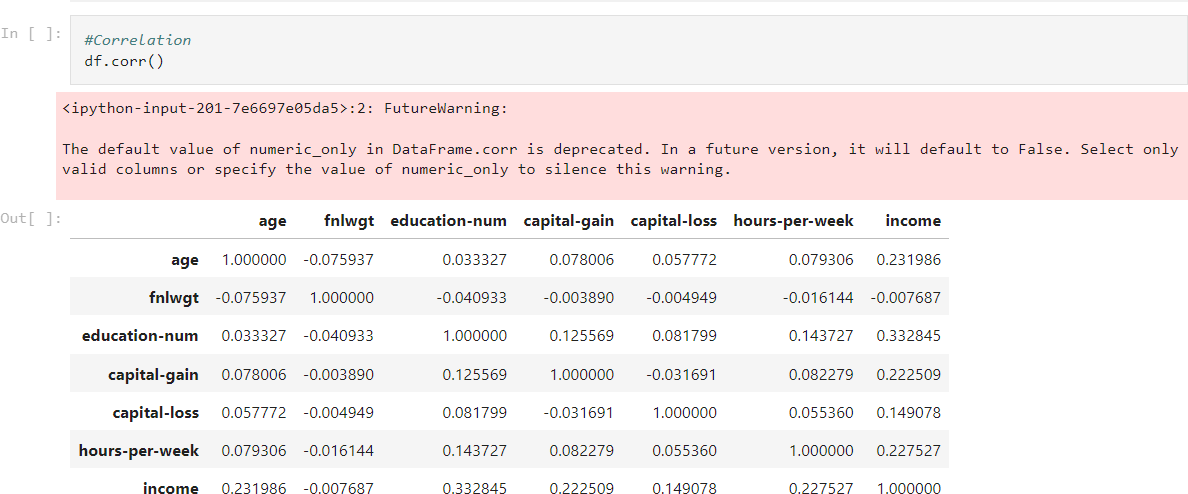
The pie chart shows the distribution of education levels in the population with 32% of the population in high school, 17% studying undergraduate, 13% of the population in school, and so on.



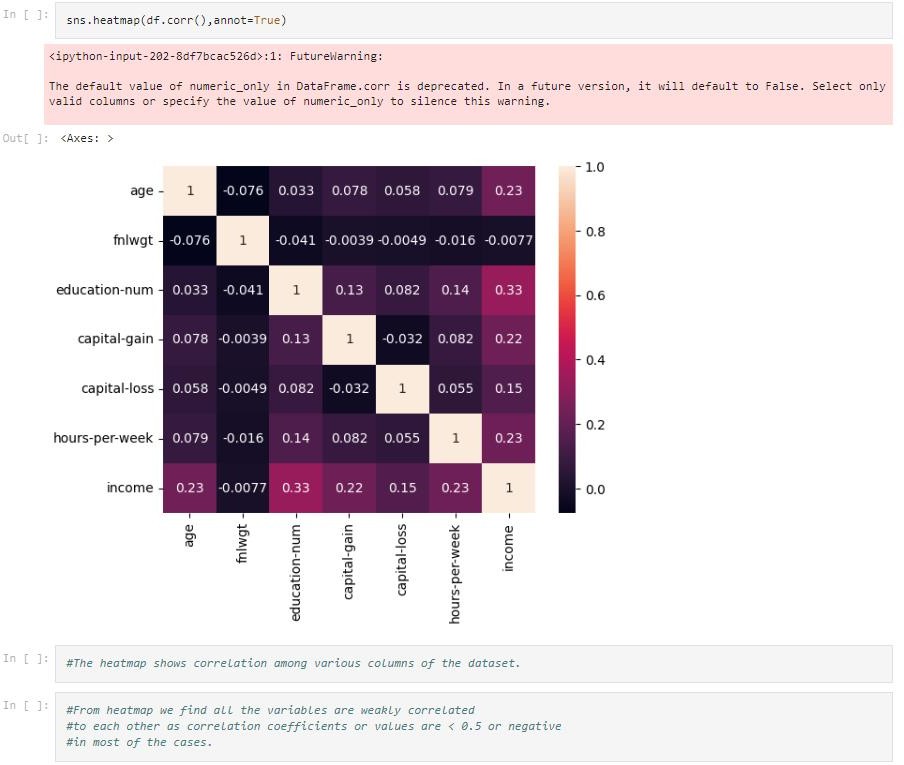
We grouped various values of the marital-status column into 1 column as many values pointed to the same column values. We made divorced, separated, windowed, and married-spouse-absent into ‘Other’ for easy distribution and understanding of the data.



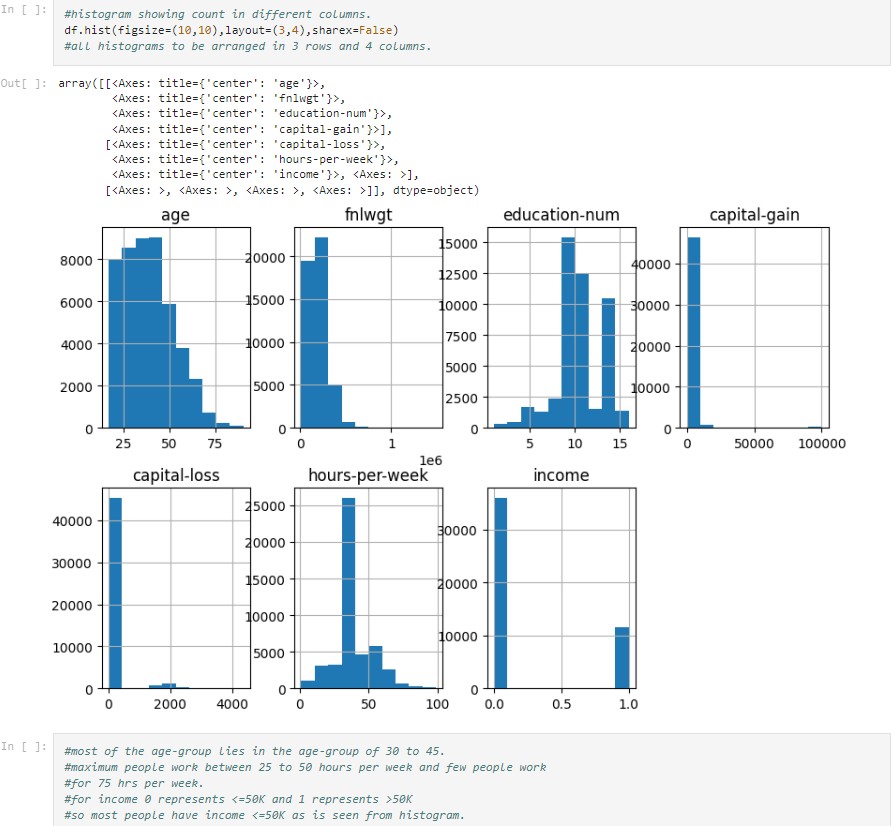
The pie chart shows the distribution of marital status in the population with 46% of the population being married, 33% unmarried, and 21% in the ‘Other’ category.



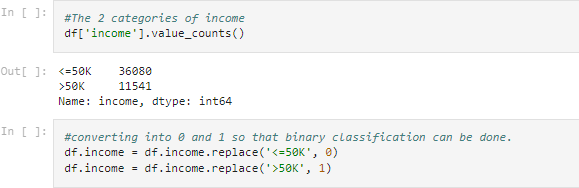
df. corr() is used to get correlation values among different columns or attributes of the dataset. This can be plotted into a heatmap by using the Seaborn library.



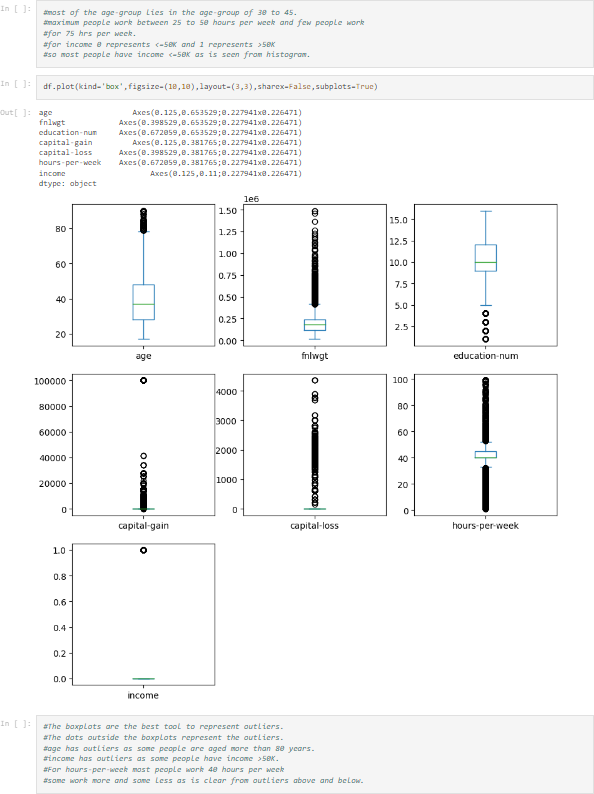
The heatmap shows the correlation among different columns and we find the correlation coefficients are < 0.5 or negative in some cases, so we conclude that the variables/columns are weakly correlated.



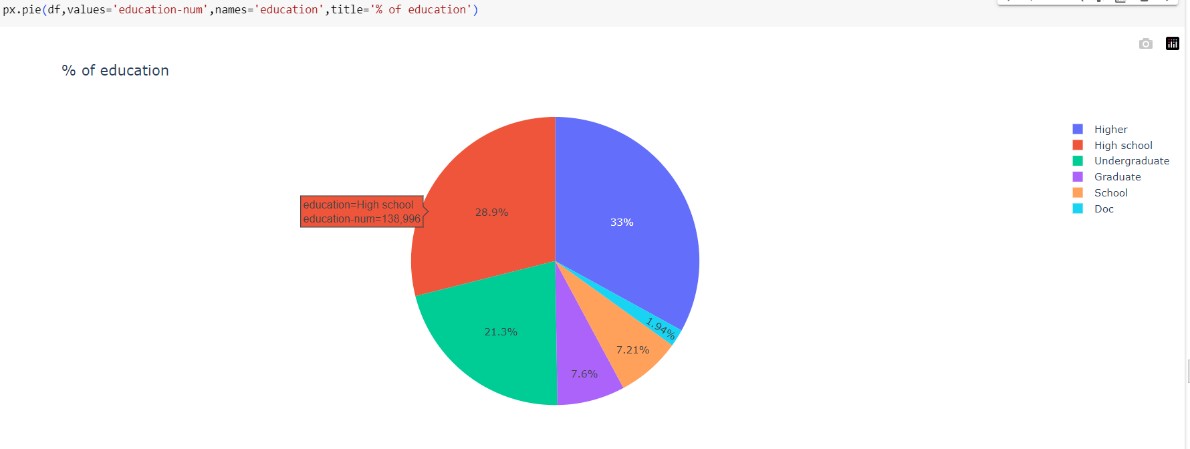
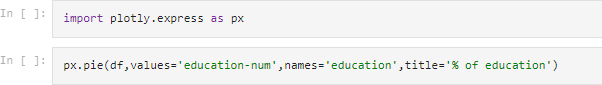
df.hist() plots histogram and here we plotted to count distribution for different columns. From the graphs, we can conclude that the majority population lies in the age group of 30 to 45. Maximum people work 25 to 50 hours per week but few people also work for 75 hours per week. For income 0 represents <=50k and 1 represents>50k income level.



We changed it into 0s and 1s as it would be useful for binary classification of the 2 income levels(<=50k and >50k). So, by the graphs majority of people have income <= 50k.



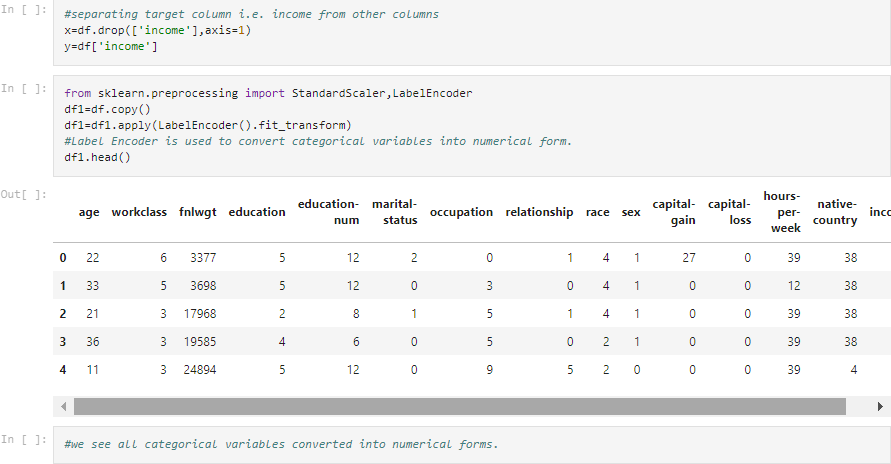
df.plot(kind=’box’,..) plots boxplots showing the distribution of different columns of the dataset. Based on the boxplots above we can conclude that the Age column has outliers as some people are aged > 80 years. The income column also has outliers as some people have income >50k while the majority population has income <50k. Also in hours-per-week, we find most people work 40 hours per week but some people work much more and some very much less as is seen from outliers.



Plotly is used to create beautiful interactive web-based visualizations.

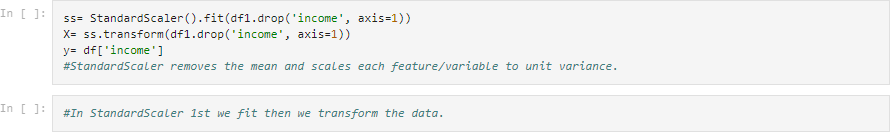
Based on the education-num column the ‘Higher’,’ High school’ and ‘Undergraduate’ have higher shares.

Now we move to **Machine Learning Classification**.

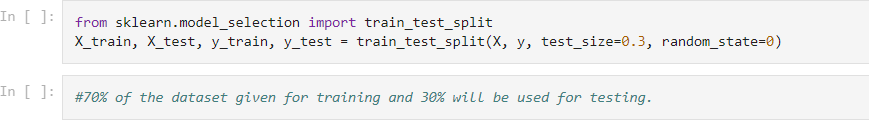


First, we separated the target column i.e. income from the dataset and stored it in ‘y’. We use a Label Encoder as the data has categorical values that cannot be understood by the ML algorithm so to convert the categorical values to

numerical variables we use the label encoder.



Standard Scaler removes the mean and fits each variable to unit variance. For a standard scaler we first fit and then transform the data.

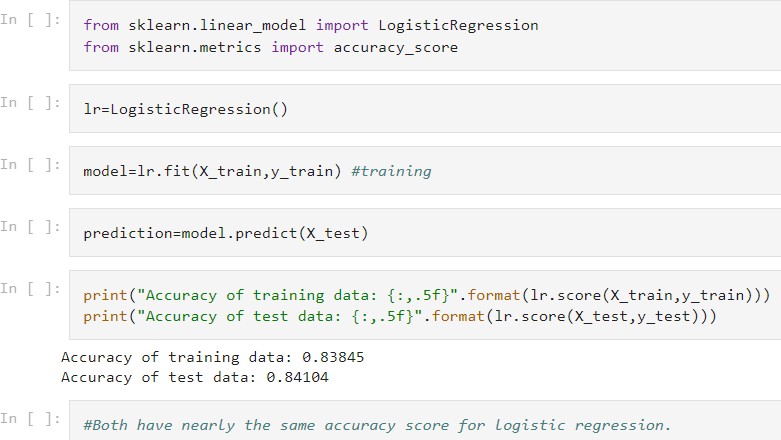


Using sci-kit-learn’s train\_test\_split we split 70% of data into the training set and 30% into the test set. The y contains the target column and x has all other columns and random\_state = 0 fixes the distribution of split and we can give any no. to the random state.

The target ‘income’ column has 2 values 0 and 1 denoting income <= 50k and income>=50k which is a case of binary classification. For this case, we can use various classification models such as Logistic Regression and Ensemble

methods such as Random Forests.

# First, we proceed by Logistic Regression.

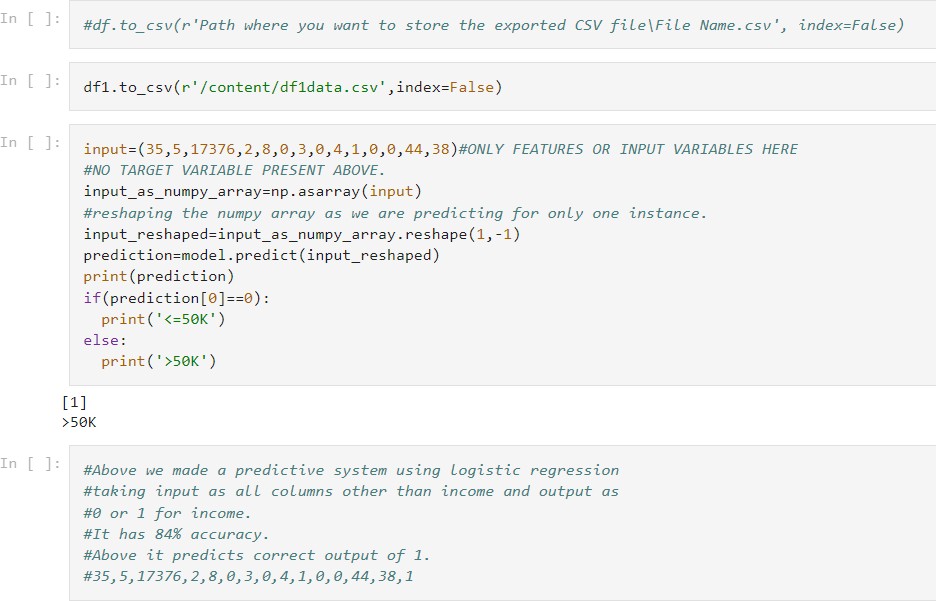


From sklearn.linear\_model we imported the LogisticRegression and from sklearn.metrics we imported accuracy\_score.

The evaluation metric for classification is the ***Accuracy score*** which is (no. of correct predictions/total no. of input data points)\*100%.

We called the lr=LogisticRegression() function. Then we fit the training dataset into that model and later predicted the results from the test dataset.

Using lr.score() we got the accuracy score for the training and test dataset. The Accuracy was found to be nearly the same i.e. 84% .



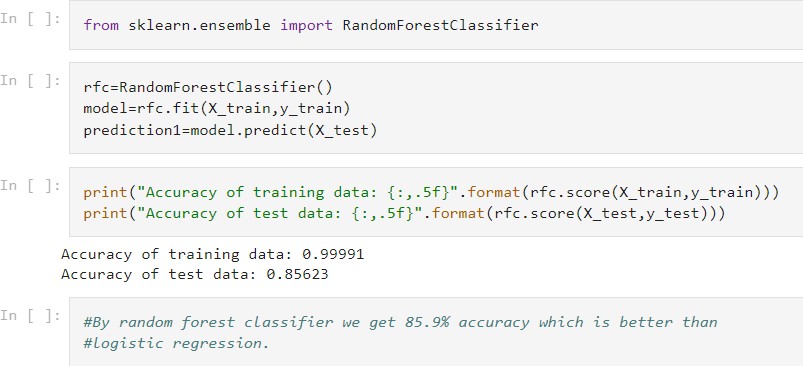
Above we stored the label-encoded and standard-scaled dataset in a CSV file and took one row from it then we predicted its income based on the model we built and we found the correct prediction that matches the dataset values.

# Second, we use Random Forest Classifier.

Ensemble learning is a supervised learning technique used in ML to improve overall performance by combining the predictions from multiple models.

Random Forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get more accurate predictions. merging for regression is the average of all predictions of different decision

trees in the random forest while merging for classification is the majority of the prediction made by different decision trees in the random forest.



From sklearn.ensemble we imported RandomForestClassifier. We called RandomForestClassifier() in rfc.

Then we train rfc on training data and then prediction is made on the test data.

The accuracy score is found by rfc.score() for training and test sets and we find 86% accuracy for test set.

The Random Forest Classifier Model performs slightly better than the Logistic Regression Model.

The confusion matrix is a table with 2 rows and 2 columns that reports the no. of true positives, true negatives, false positives, and false negatives.

The classification report displays precision, recall, F1-score, and support scores.

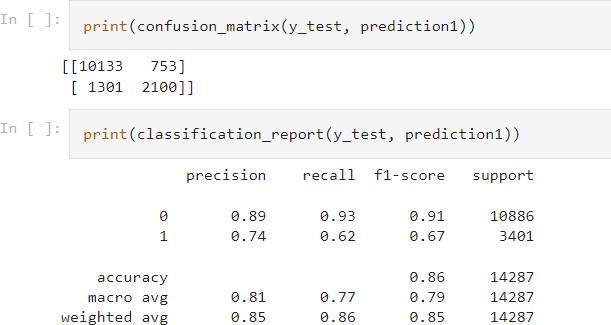
precision=accuracy of positive predictions=(true positives)/(true positives + false positives)

recall=fraction of positives that were correctly identified=(true positives)/(true positives + false negatives)

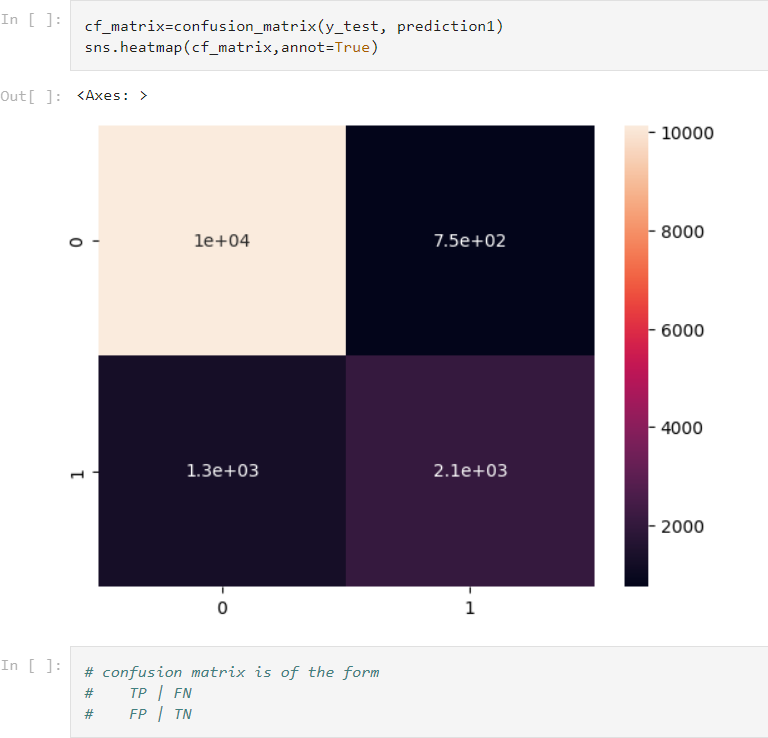
F1 score=weighted harmonic mean of precision and recall=(2\*precision\*recall)/(precision+recall)

support=no. of actual occurrences of the class in the specified dataset.

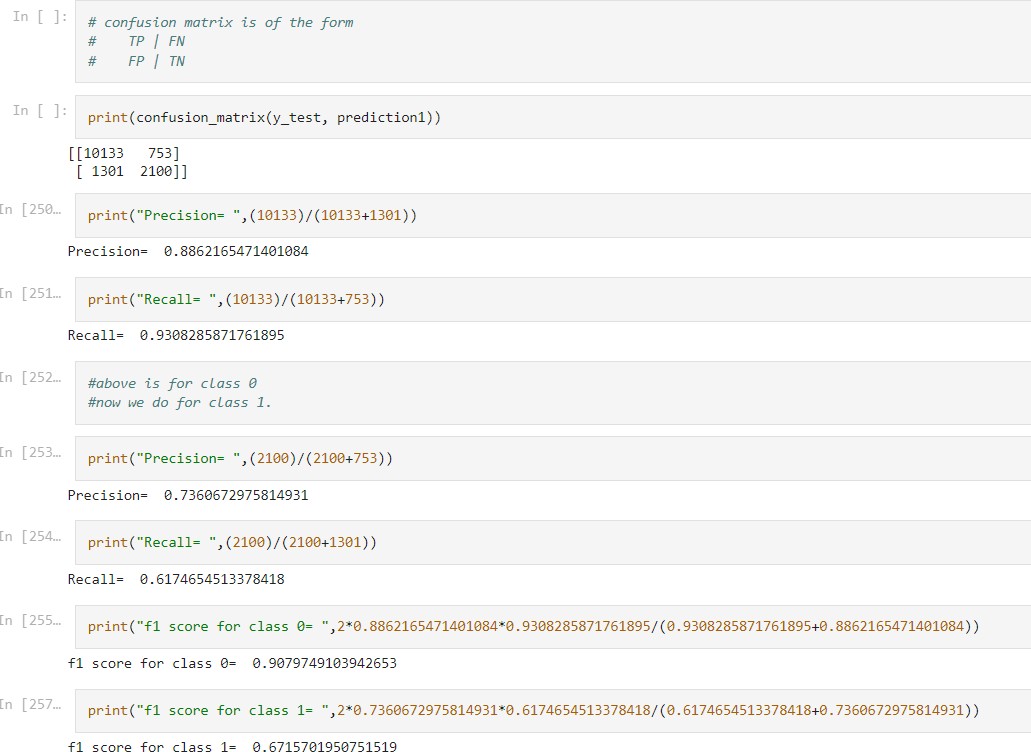
from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report



We printed the confusion matrix and classification report on the test dataset.



Above we printed a heatmap for the confusion matrix by the Seaborn Library.



We calculated the precision, recall, and f1 score for the 2 classes of income and found them to be exactly the same as those in the classification report.

# CONCLUSIONS:

The ‘Census Income’ or ‘Adult’ dataset is a classification-related dataset.

We performed and made many visualizations on it and analyzed and got valuable insights from the dataset.

We used ML classification algorithms such as Logistic Regression (as the target column i.e. income column consists of binary values) and Random Forest Classifier for better accuracy.

We got the test data accuracy score of **84% by Logistic Regression and 86% by Random Forest Classifier**.

# LINK TO THE PROJECT REPOSITORY :

# <https://github.com/himanshi-154/IDSproject>