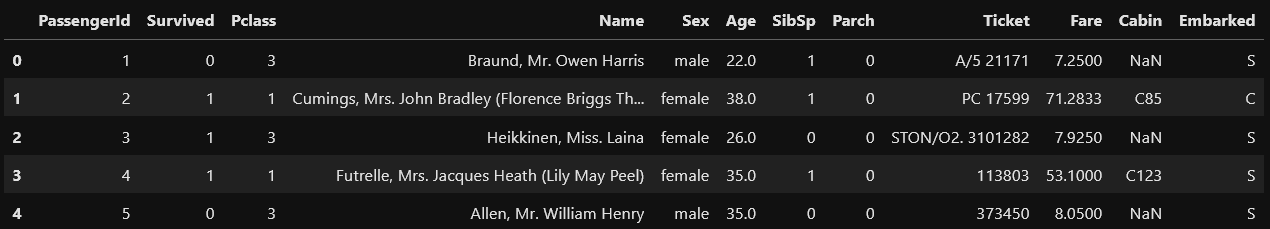
**Titanic Dataset**

**Problem Defination:**

In this project we are provided with the Titanic Dataset in which we have some columns providing the informations of the passengers travelling in the Titanic Ship and also the information of whether the peseenger survived the sink of titanic or not.



We can see in the image that there are several columns present in the dataset.

Using this dataset we have to predict the survival of the passenger whether the passenger survived or not in the accident.

**Data Analysis:**

We need to perform the data analysis on the provided data in the dataset

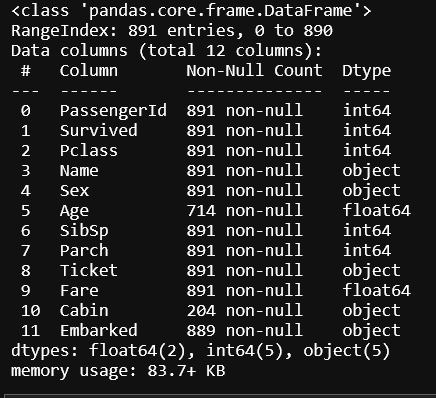
There are several types of analysis we need to do i.e.

* Checking the shape of the data using “df.shape”, as in this dataset the shape of the data is (891,12) which shows that the dataset has 891 rows and 12 columns.
* Checking the Statical summary of the data using “df.discribe()” which provides us the following details of thr dataset



We can see that it has the statistical information of the data i.e. Count, Mean, Std, Min, 25%, 50%, 75%, max.

* Checking the datatype of all the columns using ‘df.info()’



* checking if null values present in the dataset using ‘df.isnull().sum()’



As we can see that there are null values present in some columns.

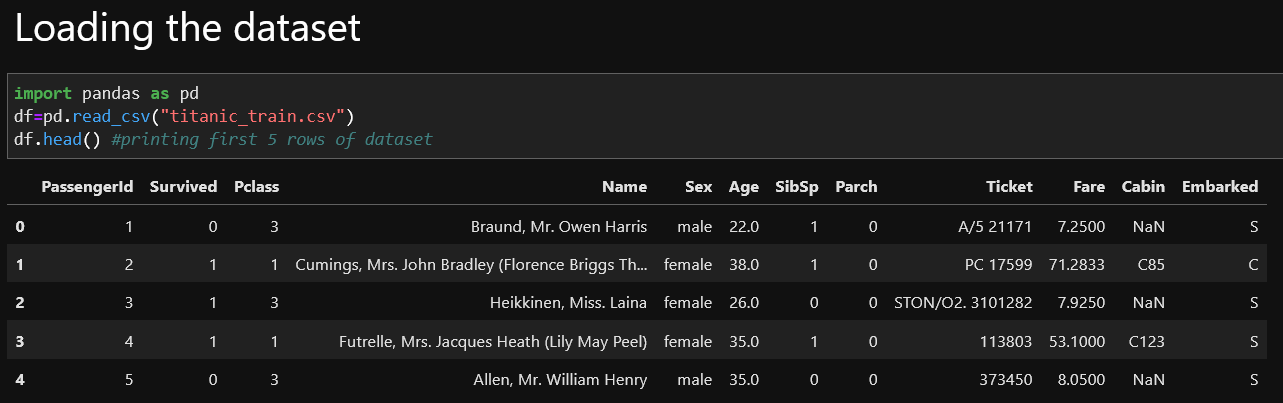
**EDA Concluding Remarks:**

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

To share my understanding of the concept and techniques I know, I’ll take an example of white variant of [**Titanic** data set](https://archive.ics.uci.edu/ml/datasets/wine+quality) which is available on UCI Machine Learning Repository and try to catch hold of as many insights from the data set using EDA.

To starts with, I imported necessary libraries (for this example pandas, numpy, matplotlib and seaborn) and loaded the data set.

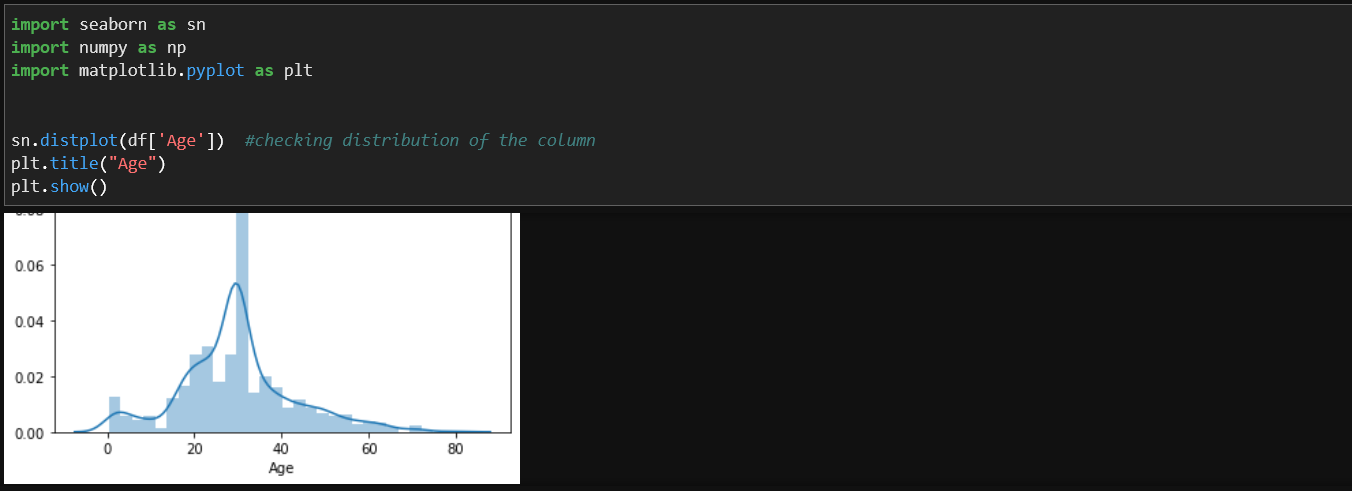


To take a closer look at the data took help of “.head()”function of pandas library which returns first five observations of the data set. Similarly “.tail()” returns last five observations of the data set.

Let’s now explore data with beautiful graphs. Python has a visualization library, [Seaborn](https://seaborn.pydata.org/) which build on top of matplotlib. It provides very attractive statistical graphs in order to perform both [Univariate](http://www.statisticshowto.com/univariate/) and [Multivariate analysis](http://www.camo.com/multivariate_analysis.html).

**Univariate**

Checking distribution of the ‘Age’ Column of the dataset



* The data of Age Column is a bit positively skewed.

Ploting the Countplot of ‘Survived’ Column of the dataset



* As seen in the graph the rate of survival is low, Around 350 passengers survived and more than 500 passengers couldn’t survive.

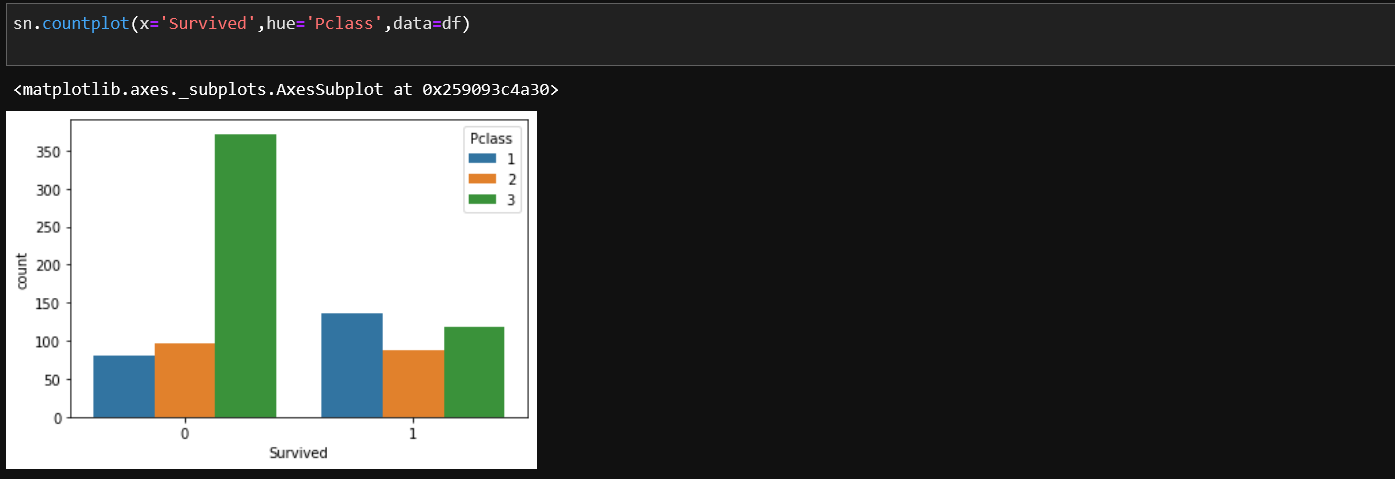
**Bivariate**

Checking the surviaval of the passengers according to their Sex



* The Blue bar represents the Male and the orange bar represents the Female as mentioned in the graph
* According to the graph the majority of the passengers survived are Females and majority of the passengers couldn’t Survive are Males.

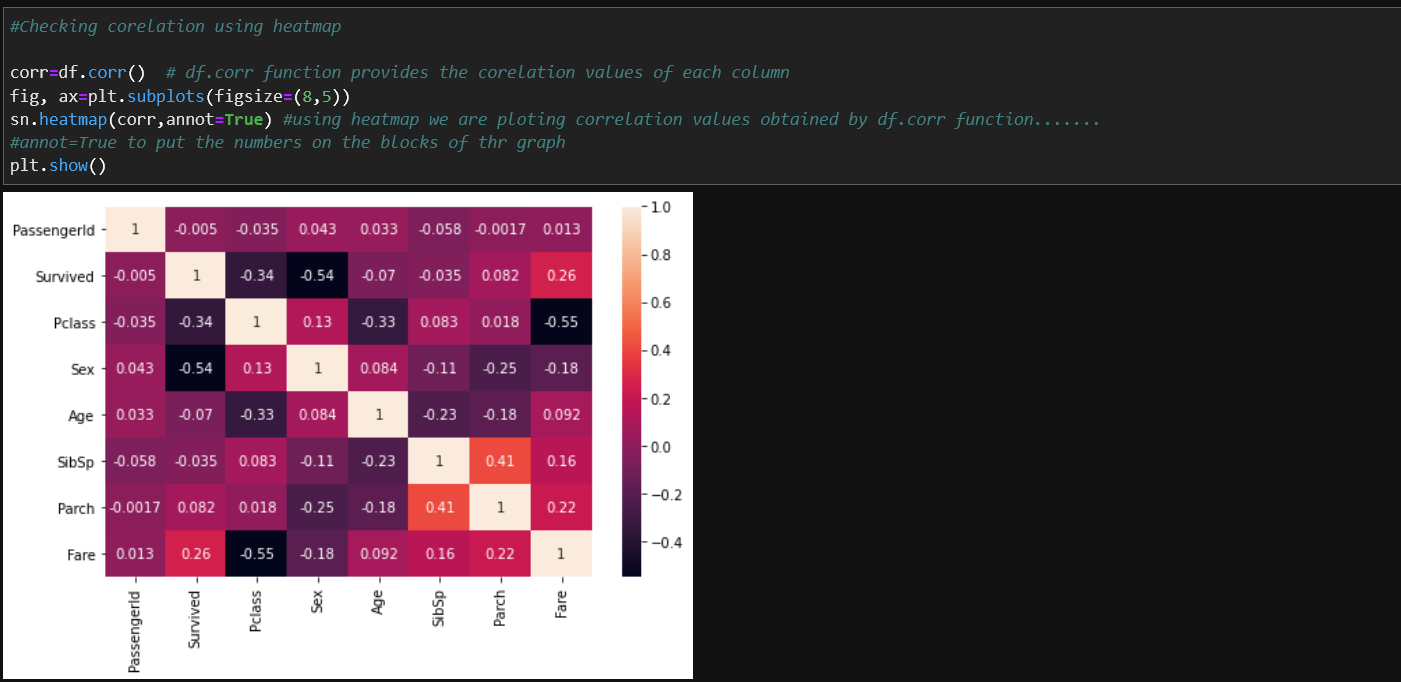
Checking the surviaval of the passengers according to the class they are travelling in.



* The Blue, Orange and Green bars represents Pclass 1,2 and 3 respectively as mentioned in the graph
* Aaccording to the graph shown we observe that majority of the passengers couldn’t survive are Pclass 3 passengers.
* And the majority of the passengers survived are Pclass 1 passengers.

**Multivariate**

Checking the correlation between all the columns of the dataset using heatmap.



* As we can see that the columns Survived and Sex is negatively correlated with each other
* Survived and Fare is slightly correlated.

**Boxplot**

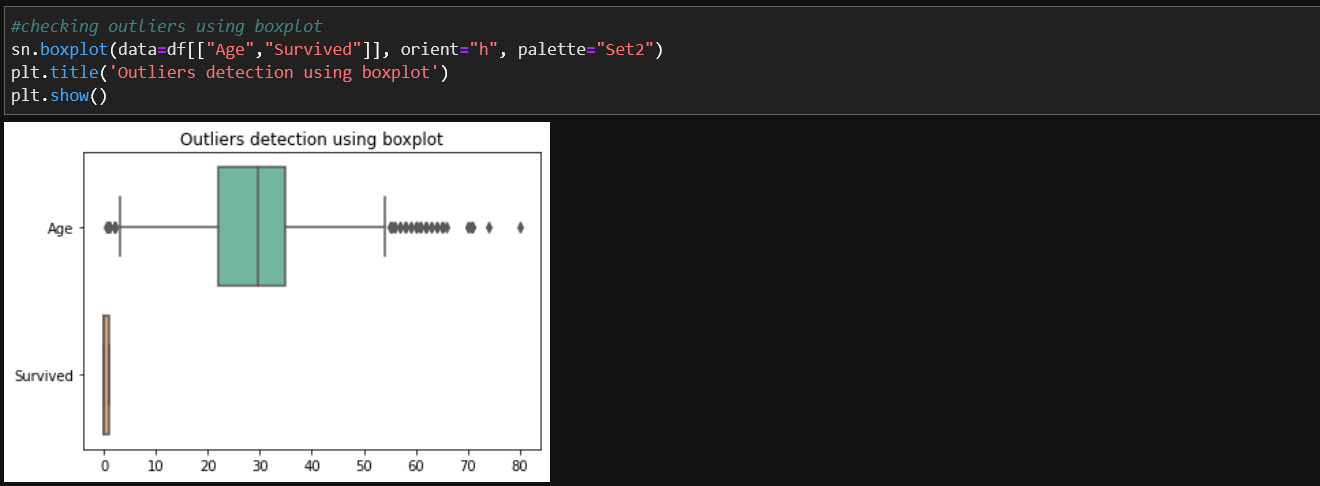
A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution.

The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the five number summary:

* Minimum
* First quartile
* Median
* Third quartile
* Maximum.

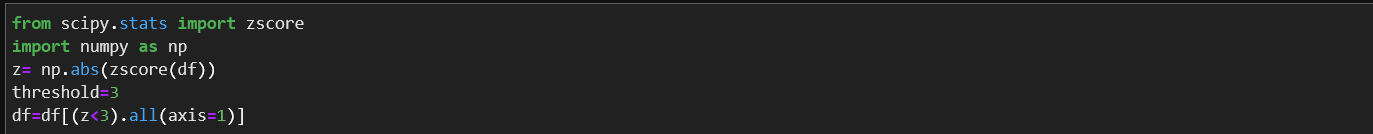
In the simplest box plot the central rectangle spans the first quartile to the third quartile (the interquartile range or IQR).

A segment inside the rectangle shows the median and “whiskers” above and below the box show the locations of the minimum and maximum.



* Boxplot also helps in detencting the outliers in the dataset
* High number of outliers present in the ‘Age’ Colummn.

Removing the oulliers



**Pre-Processing Pipeline:**

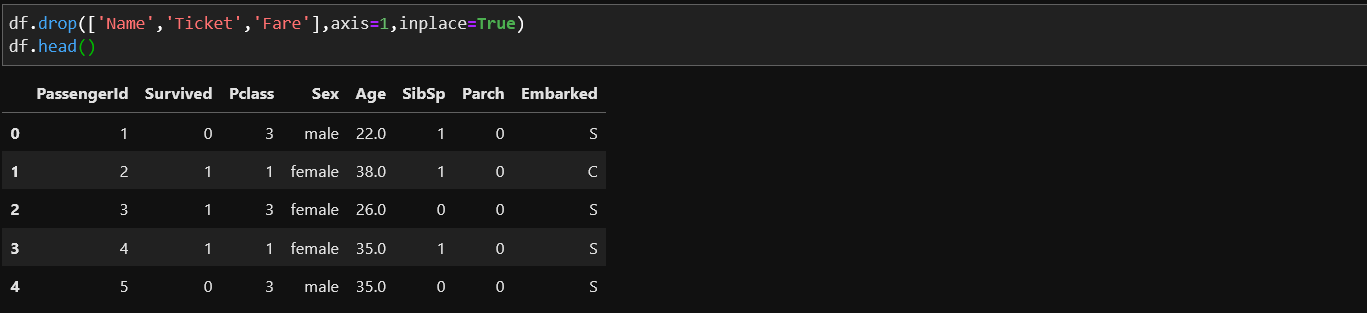
Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, the greater is the reliability of the produced results. Incomplete, noisy, and inconsistent data are the inherent nature of real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise, and resolving inconsistencies.

* Incomplete data can occur due to many reasons. Appropriate data may not be persisted due to a misunderstanding, or because of instrument defects and misfunctions.
* Noisy data can occur for a number of reasons (having incorrect feature values). The instruments used for the data collection might be faulty. Data entry may contain human or instrument errors. Data transmission errors might occur as well.

There are many stages involved in data preprocessing,

1. Data Cleaning
2. Data Integration
3. Data Transformation
4. Data Reduction

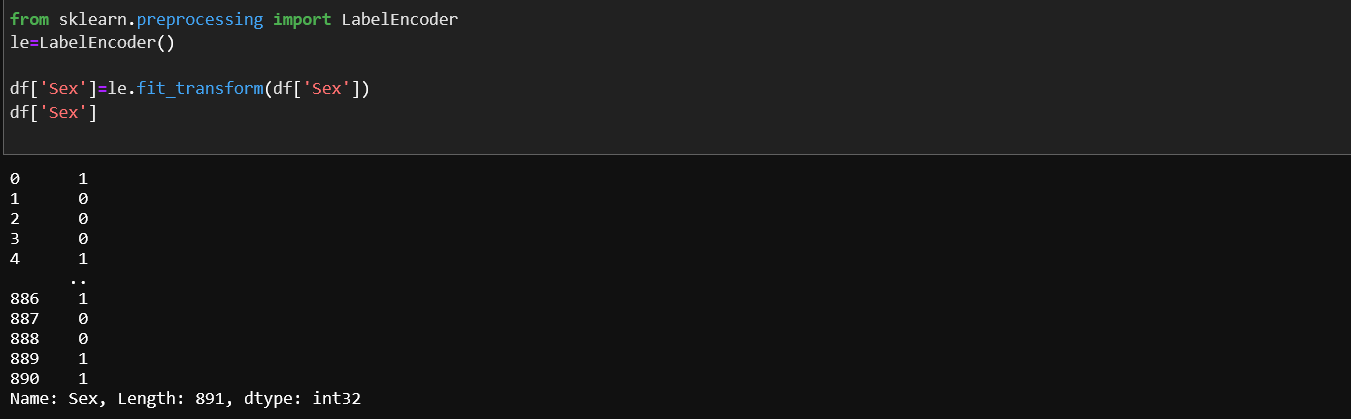
Cleaning the data

****

* Cleaning the unwanted data

# Drop the columns which has no relation with the target variable and which have more than 70% of missing values.

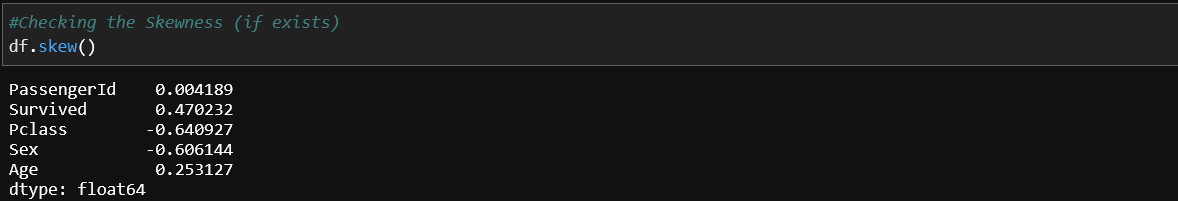
Transforming the data



Data transformation is done using different methods but here we are using **LableEncoder**

* Importing required libraries for the LabelEncoder from sklearn.preprocessing.
* Transforming the ‘Sex’ column from object datatype to int.

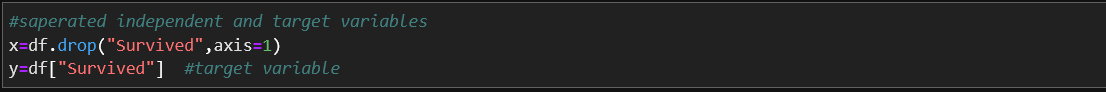
Checking the skewness of the data by using ‘df.skew()’.



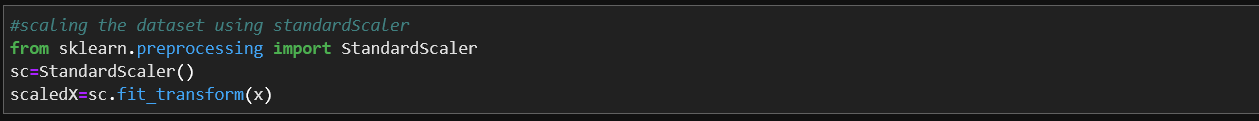
The skewness of the data can be acceptable in the range of -0.5 to 0.5.

* As observed there is some skewness present in the dataset in column ‘Pclass’ and ‘Sex’ but it can be ignored.

Saperating the independent and the target variables.



Scaling the data using Standard scalar method



* Importing StandardScalar using sklearn.preprocessing

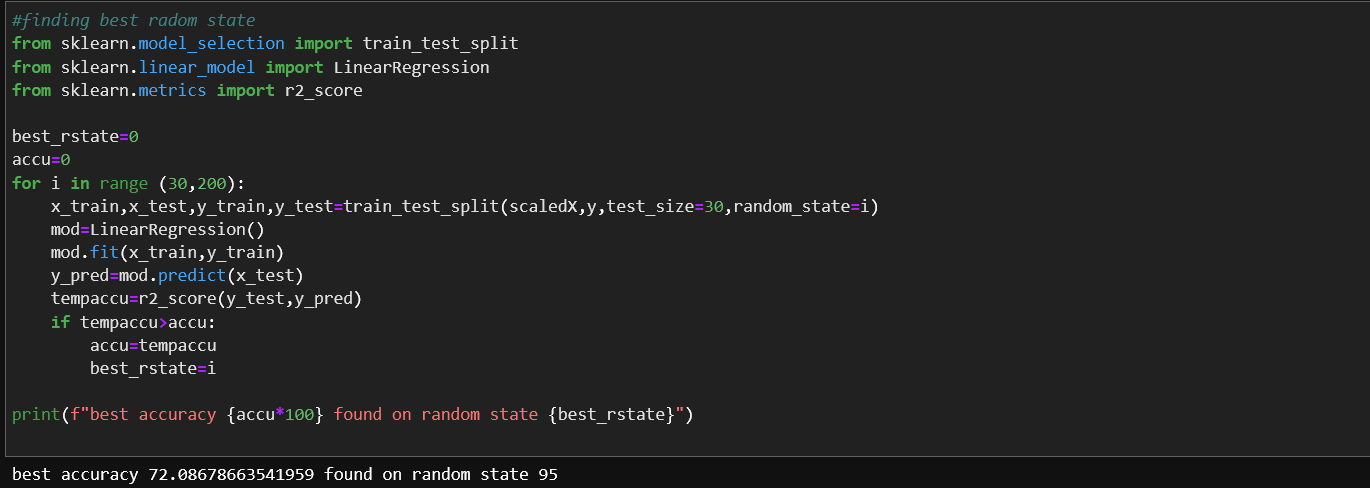
Data Spliting

## Train-Test Split

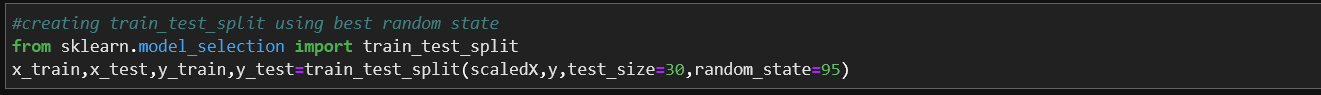
In the development of machine learning models, it is desirable that the trained model perform well on new, unseen data. In order to simulate the new, unseen data, the available data is subjected to data splitting whereby it is split to 2 portions (sometimes referred to as the train-test split). Particularly, the first portion is the larger data subset that is used as the training set (such as accounting for 80% of the original data) and the second is normally a smaller subset and used as the testing set (the remaining 20% of the data). It should be noted that such data split is performed once.

Next, the training set is used to build a predictive model and such trained model is then applied on the testing set (i.e. serving as the new, unseen data) to make predictions. Selection of the best model is made on the basis of the model’s performance on the testing set and in efforts to obtain the best possible model, hyperparameter optimization may also be performed.

Importing train\_test\_split from sklearn.model\_selection.



As observed we found that the best accuracy calculated is 72.08% and random state is 95, using this random state we will perform train test split.



**Building Machine Learning Models:**

Now, comes the fun part where we finally get to use the meticulously prepared data for model building. Depending on the data type (qualitative or quantitative) of the target variable (commonly referred to as the **Y** variable) we are either going to be building a classification (if **Y** is qualitative) or regression (if **Y** is quantitative) model.

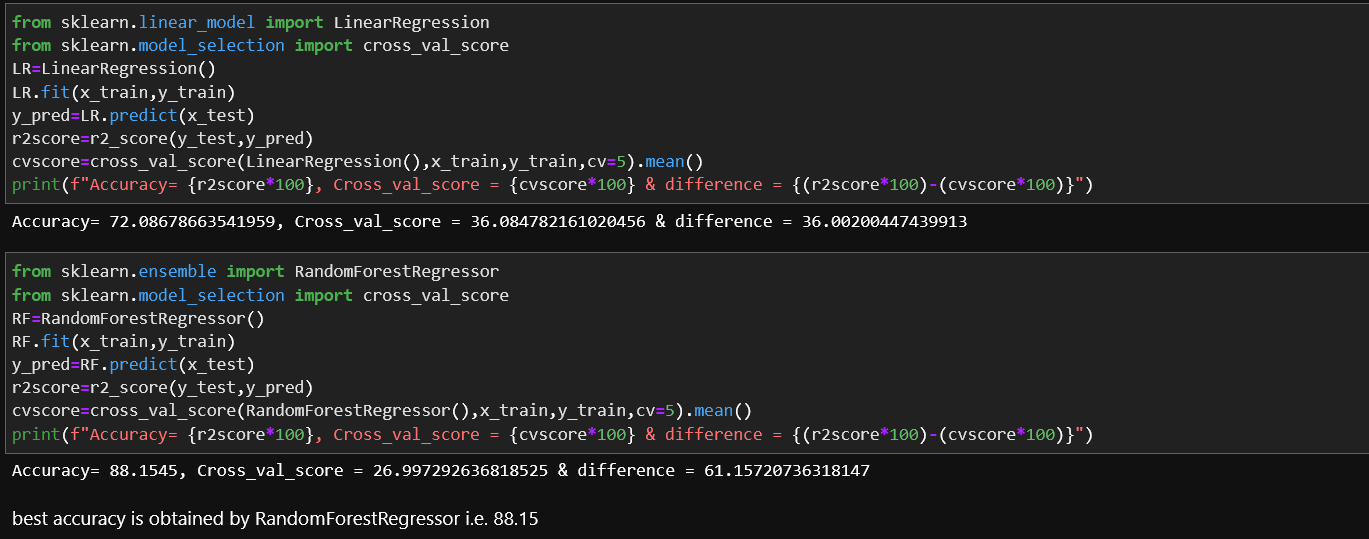
## Learning Algorithms

Machine learning algorithms could be broadly categorised to one of three types:

1. Supervised learning — is a machine learning task that establishes the mathematical relationship between input **X** and output **Y** variables. Such **X**, **Y** pair constitutes the labeled data that are used for model building in an effort to learn how to predict the output from the input.
2. Unsupervised learning — is a machine learning task that makes use of only the input **X** variables. Such **X** variables are unlabeled data that the learning algorithm uses in modeling the inherent structure of the data.
3. Reinforcement learning — is a machine learning task that decides on the next course of action and it does this by learning through trial and error in an effort to maximize the reward.

Here we are using the regression Models in model building process because it is a regression problem

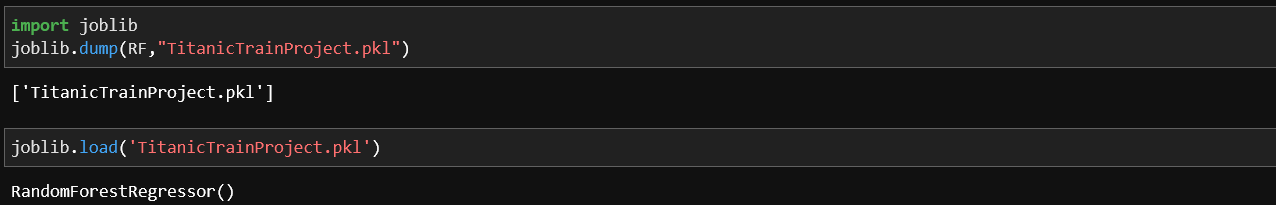
Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc.



* From the output we got it is clear that from linear Regression and RandomForestRegressor, the RandomForestRegresson is performing better an proving the accuracy of 88.15% which is a good percentage of accuracy.

Considering the RandomForestRegressor as the best model for the prediction of the data now we’ll serialize the data and load the project with .pkl extension

Importing joblib



Here we saved the predictions in a .pkl file bu using joblib.dump.

**Concluding Remarks:**

After all the procedures from loading the dataset into the jupyter notebook to finding the best accuracy score we have completed our project with a good accuracy score of 88.15%

we have got the 88.15% of accuracy with our models which is a good accuracy percentage.