**HR Analytics Project- Understanding the Attrition in HR**

**Introduction:**

This article in on the title “HR Analytics Project- Understanding the Attrition in HR”, a project on machine learning which we will be doing in python. Human resource analytics (HR analytics) is an area in the field of analytics. It refers to the analyzing of the human resource department of an organization with an aim to improve the performance of the employees. We will be going through the entire steps that will be needed for the completion of the project and we will understand them thoroughly. The topics that we will be covering are:

1. Problem Definition
2. Data Analysis
3. EDA
4. Pre-processing Data
5. Building Machine Learning Models
6. Concluding Remarks

Let’s start with the problem definition or a short introduction on the project that I have chosen to elaborate and why it was made in the first place.

1. **Problem Definition**

HR Analytics in an area under the field of analytics which aims to provide insights into the processes of an organization by collecting the data and using it to analyze the performance of the organization in various fields and then finding ways to improve the performance.

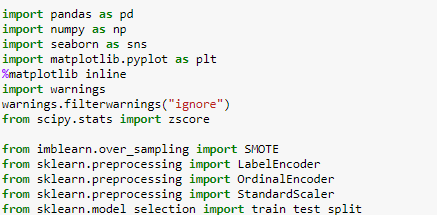
Attrition in HR generally refers to the gradual loss of employees overtime. This is problematic for the organization, as this increases the expense of the company with the new hiring, paper works, and training the new employee. It also affects the company in way that an experienced employee is replaced with a new hire, who will take time to get used to the company and gain experience and errors are more likely to occur if you constantly have new workers. Also, it is more concerning if the business is related to facing the customers, as customers feel more comfortable interacting with familiar faces.

Here, we need the help of machine learning to find a solution for this problem. We can use all the previous employee data of the organization along with the attrition data, i.e. If the worker had attrited or not. The data can then be feed into the machine and we can process the data and get an understanding on when and why workers choose to attrite and create a model that can predict using the new employee data that whether the person is likely to attrite or not.

Therefore, the major goal of this project is to identify the “Attrition” rate as a simple Yes or a No tag making this to be a classification problem!

1. **Data Analysis**

First, we are going to import all the necessary dependencies here that will be used in our project and obtain the rest as and when required. Before we begin with any process, we need to get the dataset in our Jupyter Notebook that can be achieved by a single step as shown below.

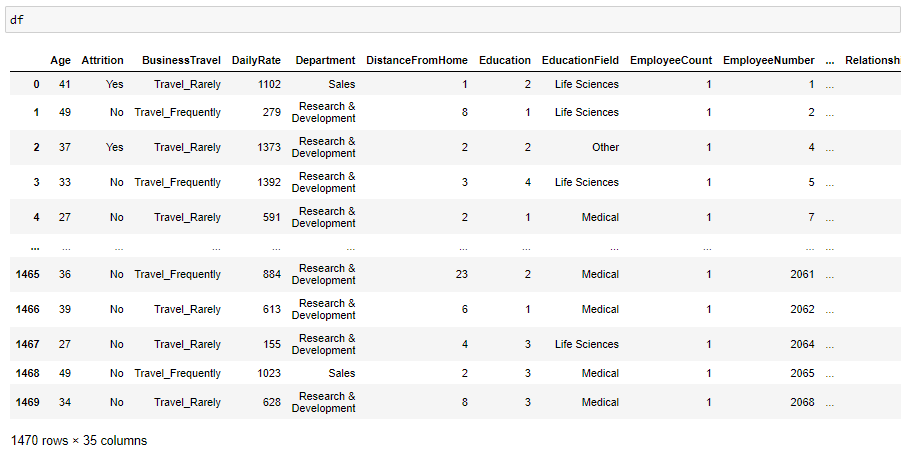


Also, I imported the entire dataset on the Jupyter Notebook with the help of pandas.



This gives us our entire dataset stored in the variable name “df” for our dataframe.

For data analysis part we can simply eye ball the contents for our dataset trying to make sense of some columns, it’s related values and anything that comes to your mind.



In the above line of code, we can see that the total number of rows present in our data is 1470 and the total number of columns are 35. Since it is a dataset with reasonably higher number of rows and columns the visualization gets truncated.



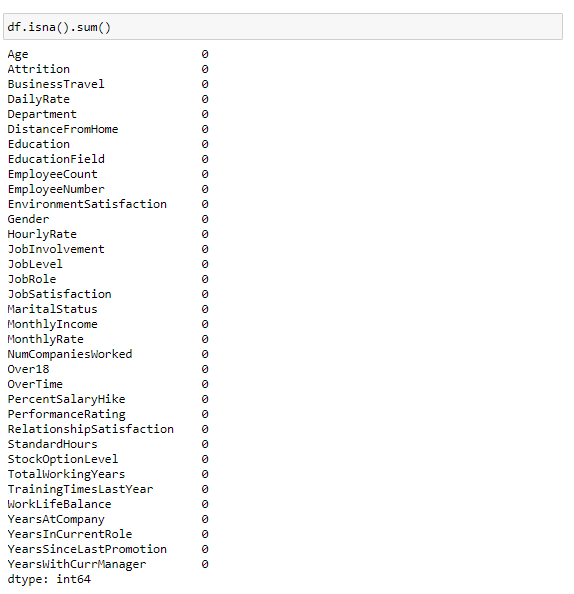
You can write this piece of code and then try to check the dataframe “df” again. It will show you the entire row and column information on your Jupyter Notebook directly.

1. **EDA**

EDA also known as Exploratory Data Analysis is considered the most important aspect in Data Science by many Data Scientist including me. After following a huge number of expert Data Scientist on various platforms I can confirm one thing that it boils down to a single important thing of conveying a story on how you were able to achieve each and every step via your code showing the provided problem statement, the observation, the challenges faced and what was done to tackle or rectify those issues.

Building a good model comes only when you understand clearly what you are doing and why you are doing it. Making sure that you have a clean data in proper processed format to feed into your model and get appropriate result. Because no amount of Machine Learning model usage and hyper parameter tuning is going to help if you have not invested time to sort out and fix your data that’s the only input you have at hand.

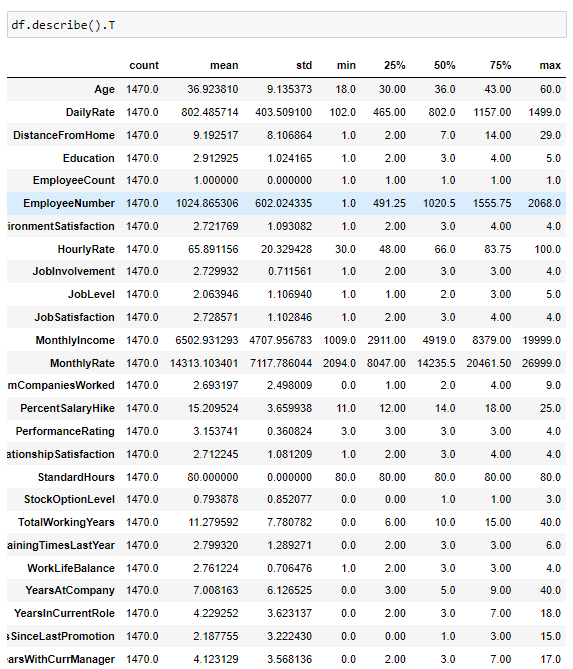
The first thing I am going to take a look at is the missing data information in our dataset by using the codes below.



This code gives us the missing values information in a tabular form that looks something like this.

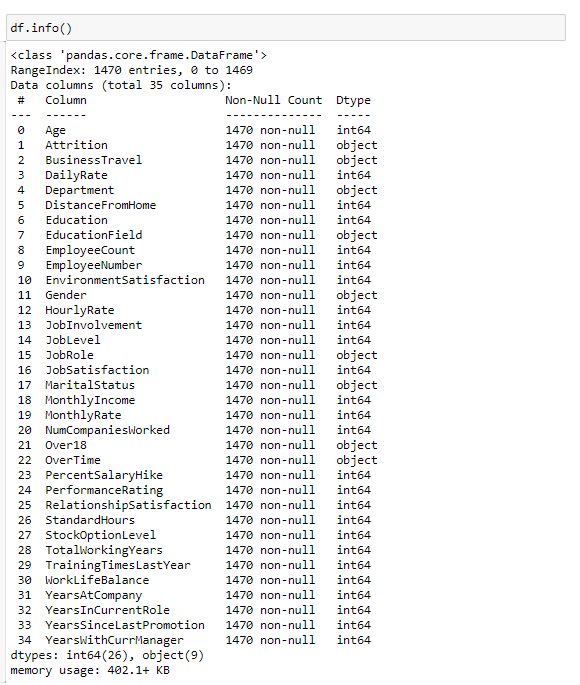
Now that we were able to confirm our dataset being free of any missing data, we will drop any duplicates that might be present using the code below.

Next, we move on to using the describe method to take a look at the count value, mean data, standard deviation information and the minimum, maximum, 25% quartile, 50% quartile and 75% quartile details. As the describe method works best for numeric data all the object (text) type data gets ignored.



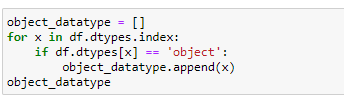
The output provided are in transpose format to accommodate all the columns from our dataset in tabular.

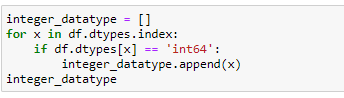
When we are able to draw insights from the describe method, we can take a look at the datatype information using the code below and that shall give us the list of all the columns marking them to be either integer, float or object datatype depending on the values present inside the columns.



This is the output that I get explaining the datatypes of all the columns present in our dataframe. We also get an opportunity to drop or remove any unwanted columns from the dataframe here.

One of the things that I like to do is separate the object datatype and numeric datatype values that allows for easier processing in further steps. The code to do that is a simple for loop usage.



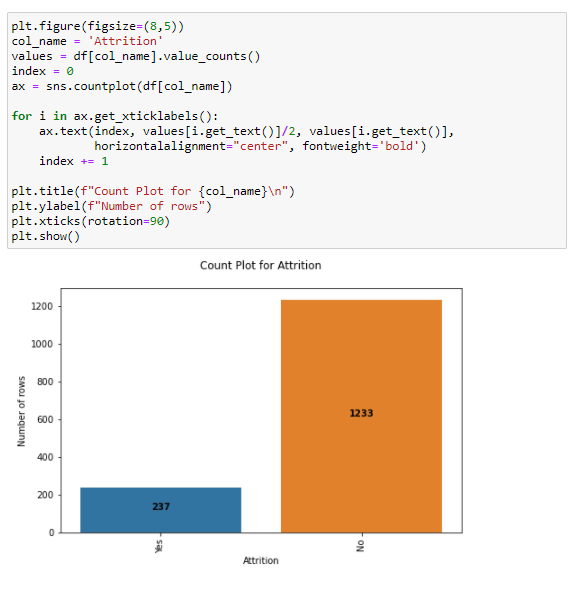
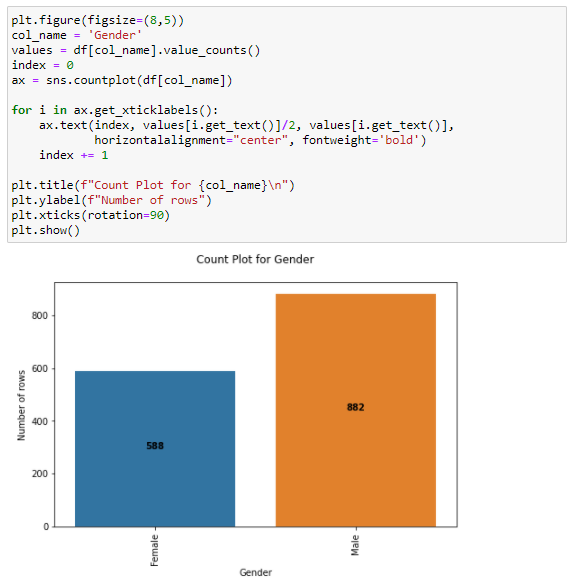


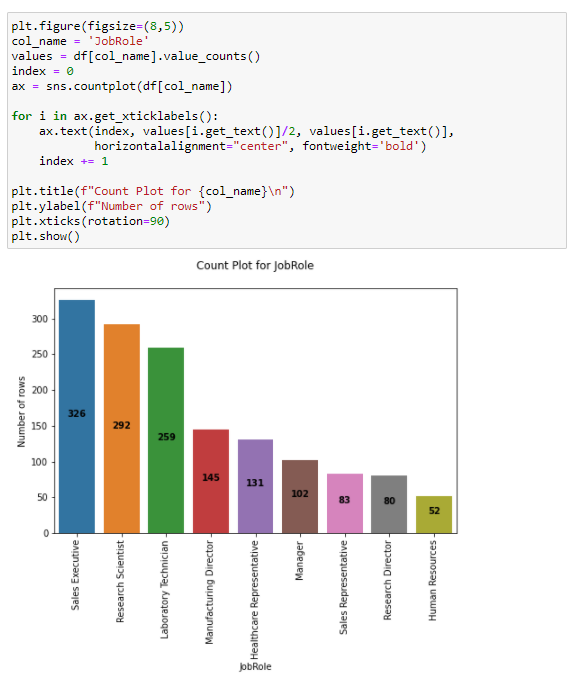
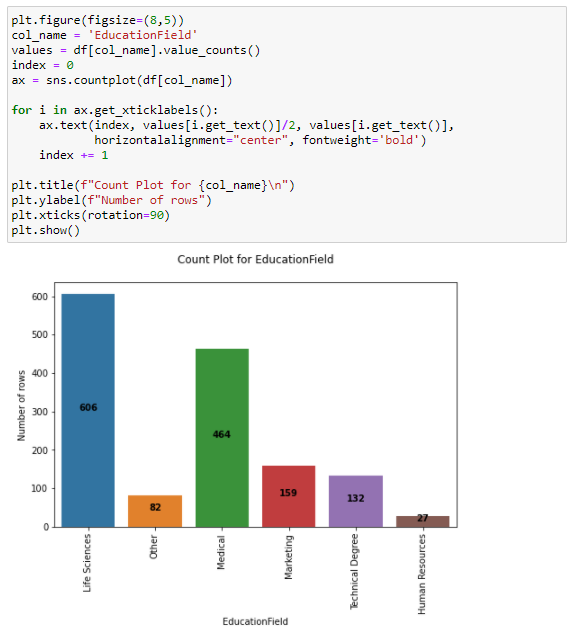
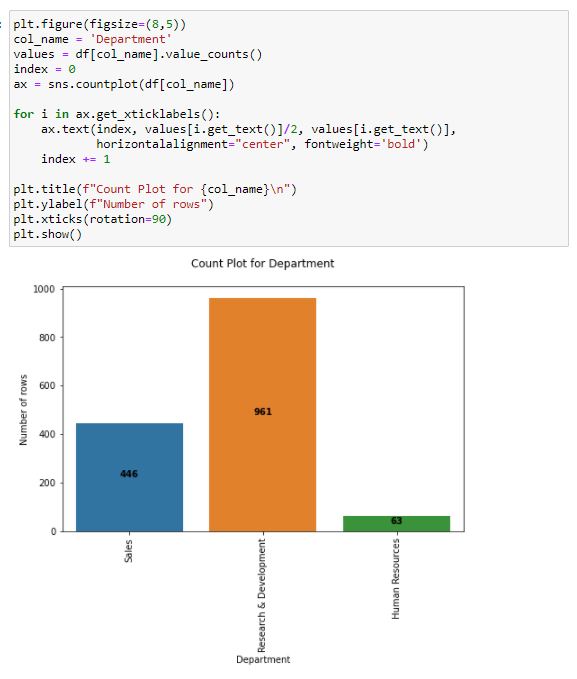
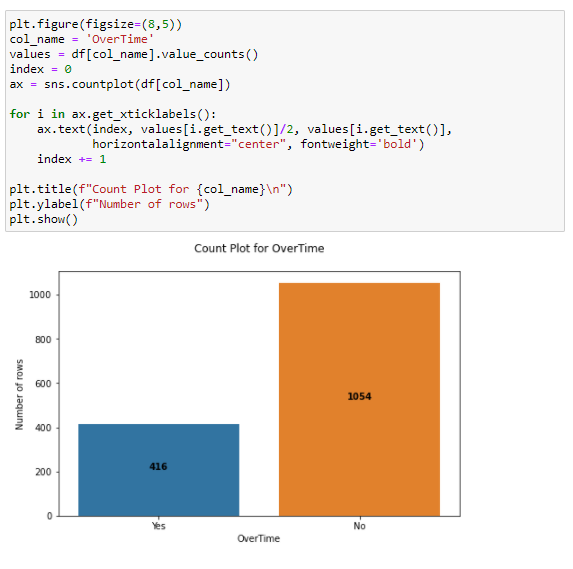
This allows us to store the column names in a list format within the variables namely object\_datatype and integer\_datatype.

After I have bifurcated the datatype column names in two separate lists, we will take a look at the overall unique values for all the columns. 

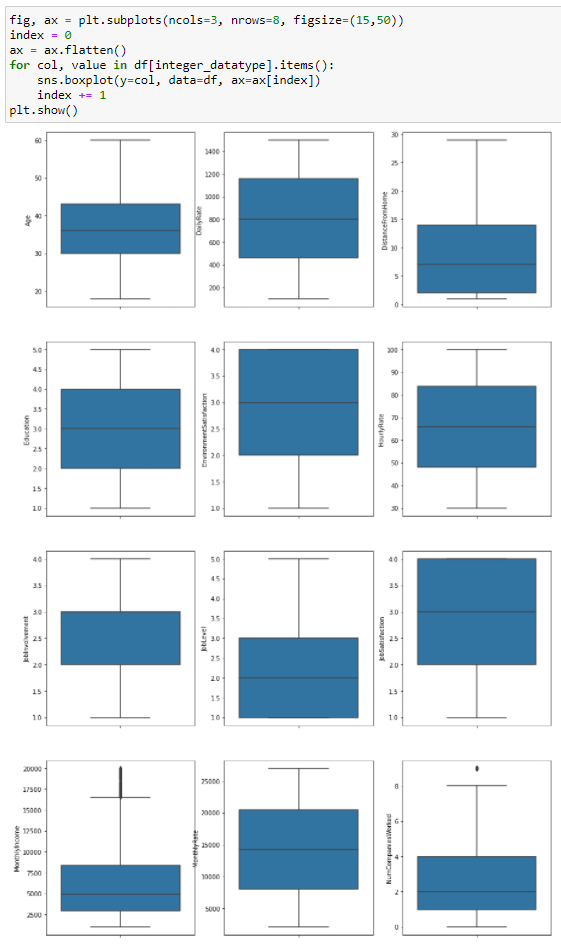
Let me go ahead and list down all the visualization codes and their output for your reference.

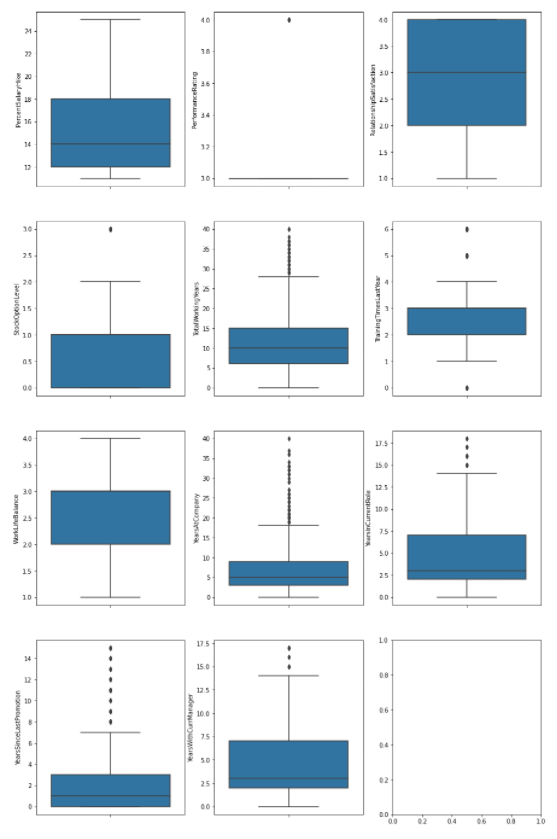
1. Count Plot



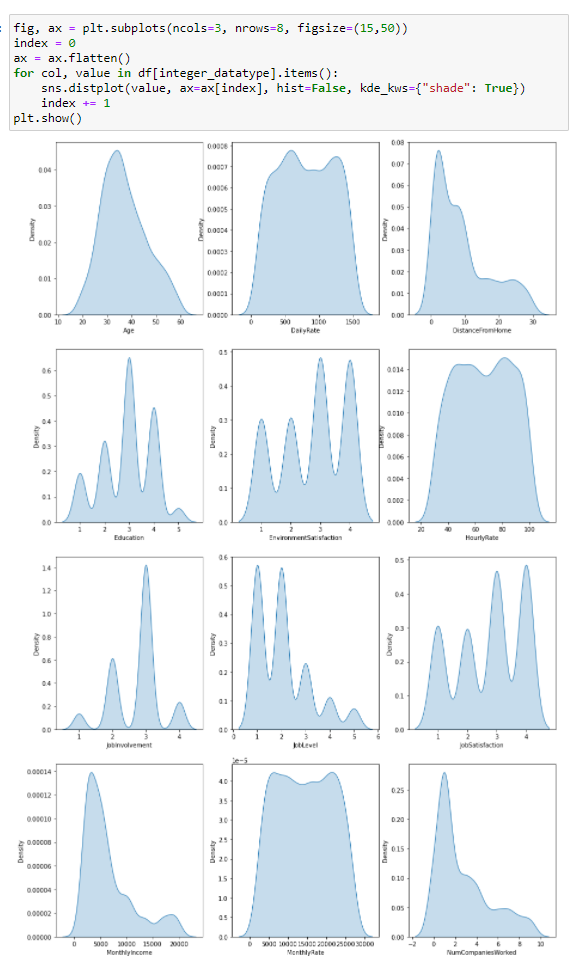


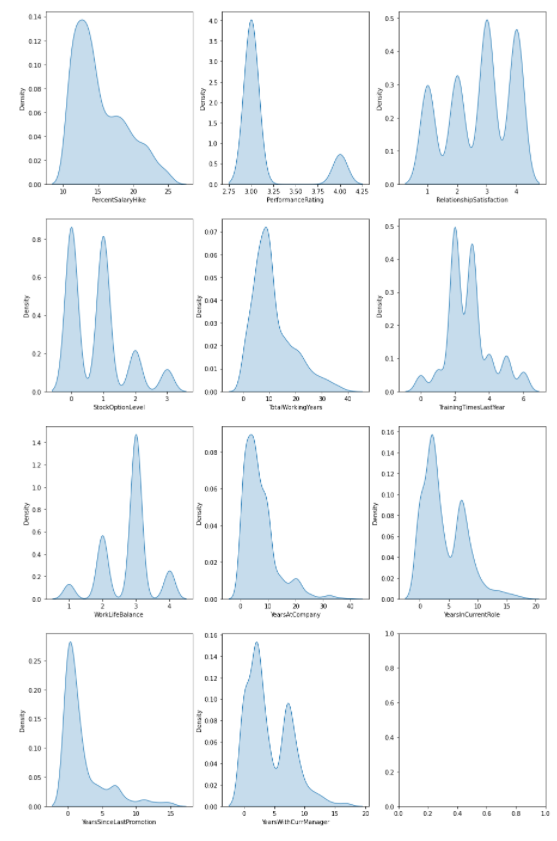
1. Boxplot





1. Distribution plot



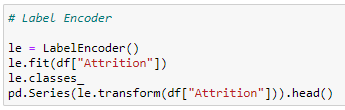


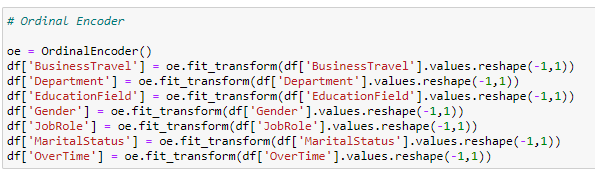
You can see that with the help of above codes and getting the outputs I was able to take a look at all the column values/counts, the boxen plots gave me a view on the presence of outliers and the distribution plots showed me the skewness information that will needed to be treated. These are like the challenges that will need to be dealt with before I even think of building my Classification Machine Learning models.

1. **Pre-processing Data**

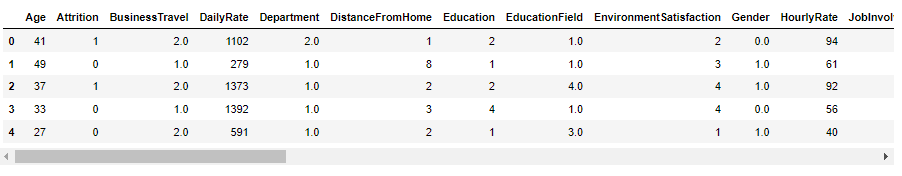
In the pre-processing step I am going to tackle all the miss fits and fix them one by one starting with the problem that out dataset has object datatype values where as our Machine Learning models can only understand numeric values. I am making use of the encoding methods to convert all the object datatype values. For our label I am using Label Encoder while for other categorical feature columns I am using the Ordinal Encoder. Instead of the Ordinal Encoder I could have used The One Hot Encoder but as I mentioned it is all about preference with trial and errors. The One Hot Encoder method increases the number columns while application of Ordinal Encoder on data values offering an order deems a better option to me. I have also seen many people apply Label Encoder on feature columns as well and it does not make any sense to me since the name itself says Label Encoder how much of a specification is required to understand that it is only for our label(s) columns. Hope my readers won’t make the same mistake!

Code:

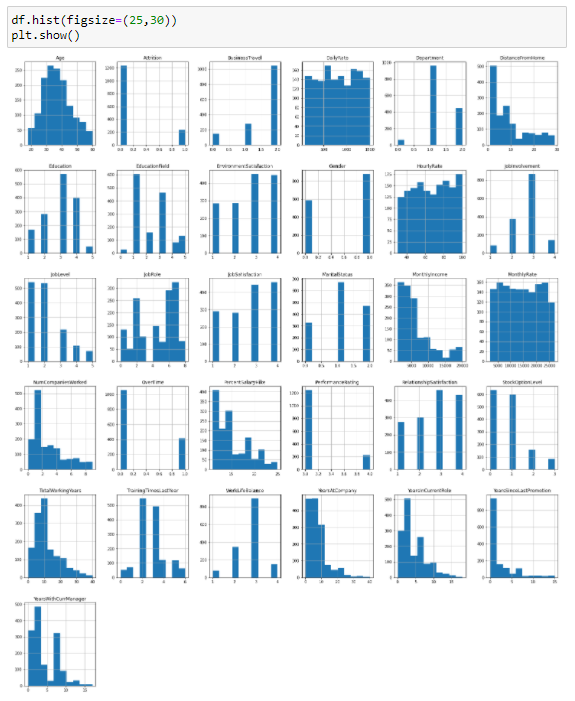




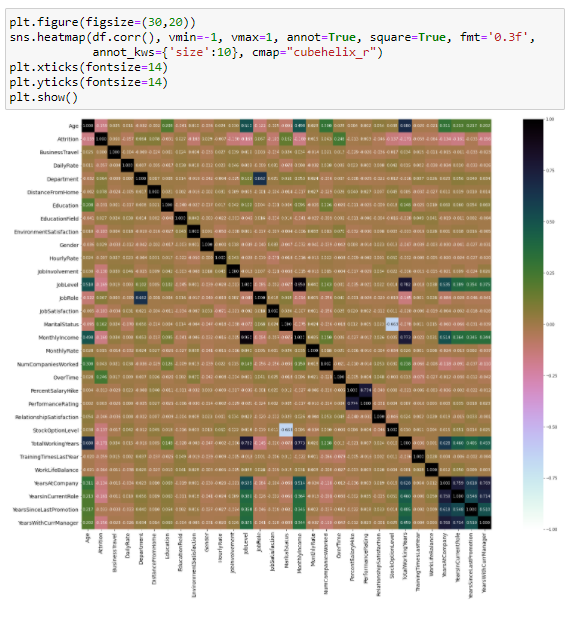
Output:



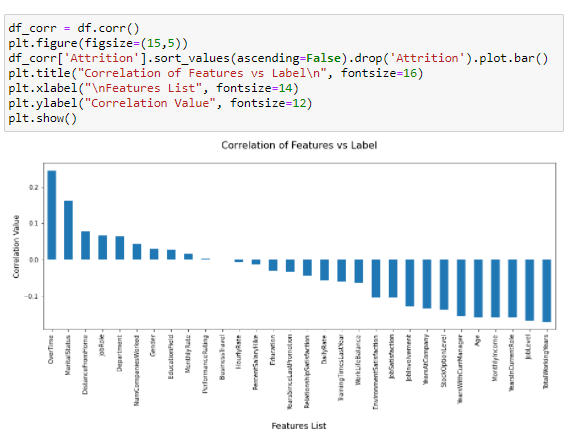
After I have encoded all the columns in our dataset, I am using a Histogram to view the data distribution. Since Histograms only consider numeric data, it should be able to identify all the information from our encoded dataframe.



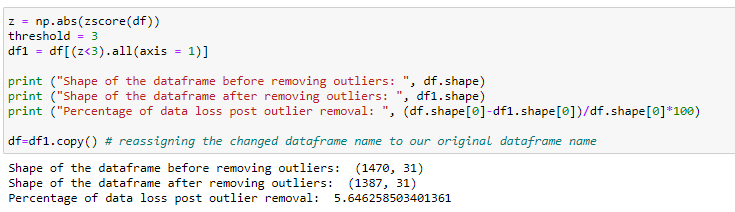
I now feel the need to check for correlation details in our dataset through a Heatmap. For those who still feel a confusion on correlation details let me break it down in two simple points that there are Positive correlation - A correlation of +1 indicates a perfect positive correlation, meaning that both variables move in the same direction together and Negative correlation - A correlation of –1 indicates a perfect negative correlation, meaning that as one variable goes up, the other goes down. The code to see this information is displayed below.



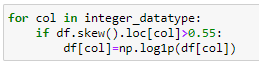
Trust me even on the Jupyter Notebook it looks all tiny due to the high number of columns. So, in such scenarios I majorly look at the colours to understand if there is any multicollinearity issue among the feature columns and if there is still any column that I can drop. But to clearly view the correlation between our label and feature columns I use a Bar Plot comparison and you can find it’s code here.



In the above Bar Plot we are able to clearly define the feature columns that are positively correlated with our label and the feature columns that are negatively correlated with our label. Now coming back to the outlier and skewness concern in our dataset I will be using the Z score and Log transformation methods.



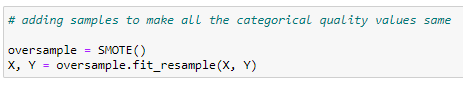
As for the usage of Z score, I was able to lose only about 5% of data but when I used the IQR method it took away I believe 30% of the data. And as Data Scientist retaining valuable data always takes priority and fixing it rather than simply deleting it unless it is the last resort. After this I am using the Log transformation to deal with the skewness since the acceptable range lies between +/-0.5 value for each column.



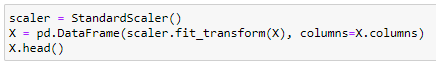
After dealing with the data concerns, I will then split our columns into feature and label. I am storing the feature columns in X and the target label column in the Y variable.



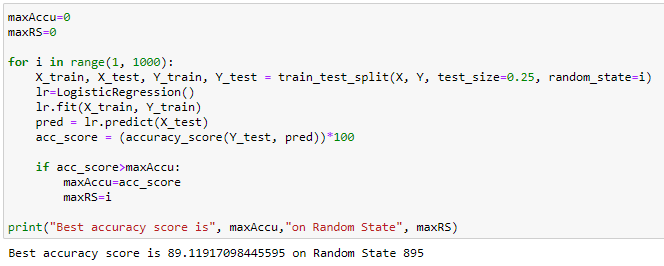
But there was an imbalance between the label classes. If you would notice the value displayed in the count plot earlier, there was a huge difference between the “Yes” and “No” data. Therefore, I will have to resolve it as the imbalance can make our machine learning model biased towards the “No” value.



Then I will also scale the feature columns that is stored in the X variable to avoid any kind of biasness over column values. Some integers cover thousands place and some cover hundreds or tens place then it can make the machine learning model assume the column with thousands place has a higher importance when in real that won’t be true due to difference in unit range.



I would like to share a simple piece of code that allows us to choose a fitting random state for the machine learning models.



Then I will use the train test split to bifurcate our entire data set into training data and testing data. Here I am using 75% data for training purpose and 25% data for testing purpose. Some people provide training and test data separately as well and hence it completely depends on you how you want to use this step.



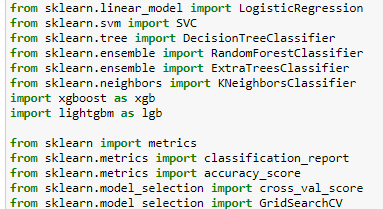
Now at this critical step before building my machine learning model I take a look at the importance of my feature columns. This gives me an insight on how the feature columns are involved and what kind of weightage they have in predicting my target label.

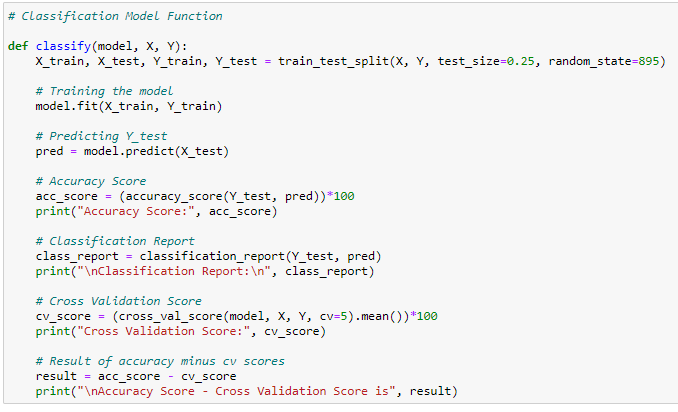
Once we have invested enough time in doing EDA and Pre-processing our data comes the step for which all the previous hard work was performed. That is to finally start building our Machine Learning model for classification purpose.

1. **Building Machine Learning Models**

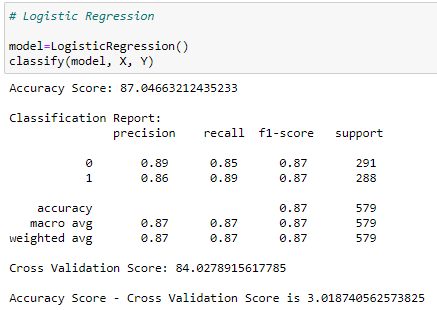
In order to build a classification method, I have imported the necessary libraries and created a function that contains all our machine learning model creation and its evaluation metrics steps. This makes our job easier since later on we just need to feed the model’s name and get the result without repeating/rewriting the same code again and again.

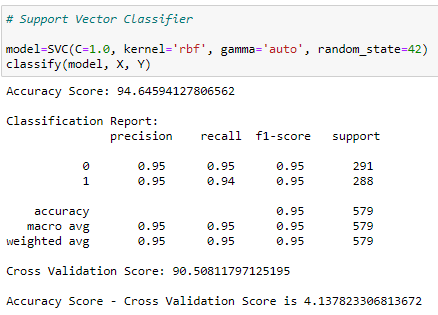
Code:

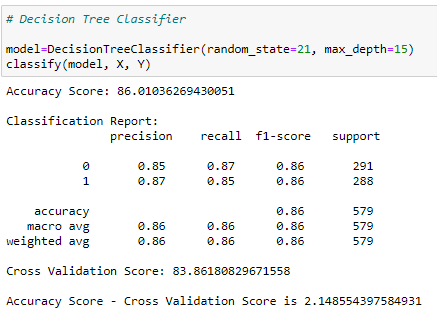


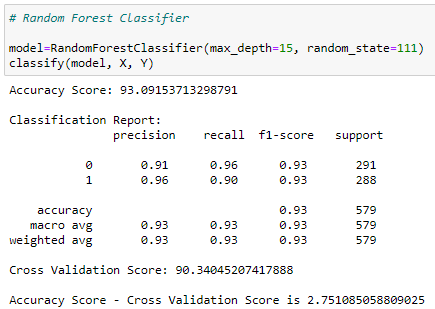


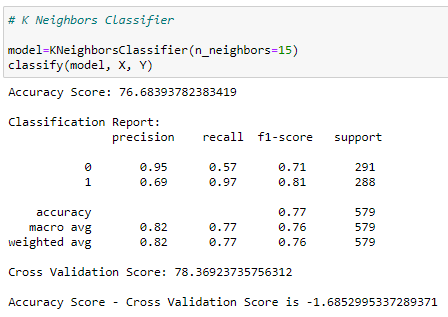
Output:

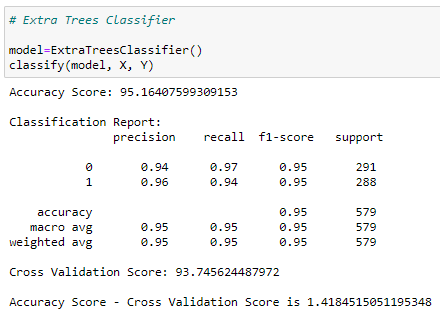


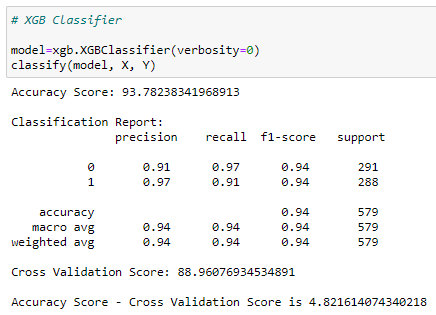


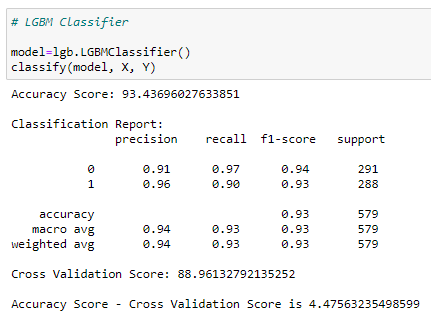






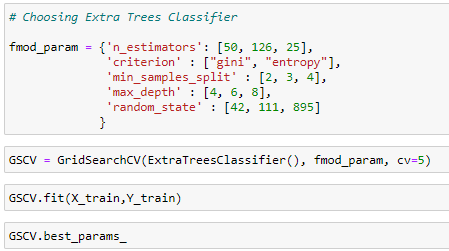




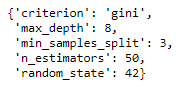


It is always advisable to build more than 5 machine learning models so that you can choose from the best performing model and then apply hyper parameter tuning to make it perform even better. I am going to use the Extra Trees Classifier as my choice of classification model as I see it is doing better than the other models I used.

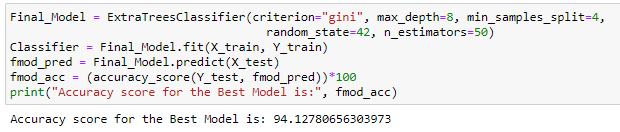
Code:

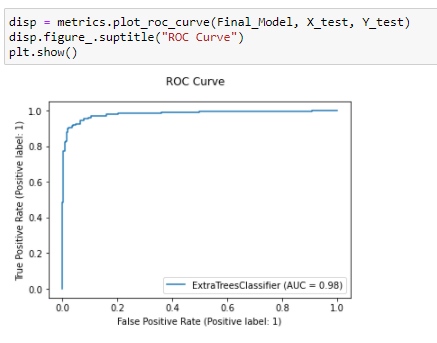


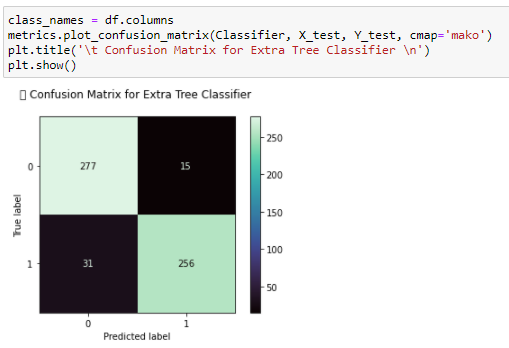
Output:



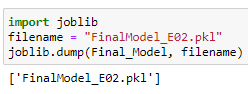
After applying the above steps to get the best parameters list, I simply have to plug it into my final model and receive the output of it. I have created an ROC curve plot and Confusion matrix for the final model.







Once you have gone through all the previous steps and you are satisfied with outcome you can then save the final model using either joblib or pickle. I have used the joblib method to save and then load my model from the same saved filename.



1. **Concluding Remarks**

With this project, we got the idea about what type of data we can work with in building a model and what type data we should avoid. We also found how balancing the target values in a classification problem play a crucial part.

By analyzing the dataset provided in this project, we found that the attrition has a high positive correlation with overtime, as overtime increases chances of attrition also increases. We also found that attrition has a negative correlation with the job level and monthly income, which meant that with a higher job level and a monthly income, the chance of attrition goes down.

We found how tuning the right parameters increase the performance of the model, and saw role of ROC curve in analyzing the performance and finalizing a model in a classification problem.