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

I am going to write about a complete end-to-end analysis for HR Analytics which should serve as a guiding path for many Data Science aspirants. When I began my journey into studying Data Science, I saw many content that is available online and most of them scattered into chunks emphasizing on a deeper knowledge that a newbie will most likely not be able to comprehend or connect through. I agree that there are already many projects available into some deep web beneath multiple clicks and paths hidden like a core of an onion protected by multiple layers. But here I am just trying to do my bit in making things easier for a new comer to understand the basic architecture that’s required in the real world for creating a Data Science project.  
  
So, without taking much time, please allow me to explain the agenda for this blog post. In this article, I have pointed down all the techniques in the form of sub-topics that I will be explaining one by one. And those pointers are as follows:

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.

**Problem Definition**

Every year all the companies hire many employees. The companies invest time and money to train those employees, not just this but there are training programs within the companies for their existing employees as well. The aim of these programs is to increase the effectiveness of their employees. But where HR Analytics fit in this? and is it just about improving the performance of employees?

**What is HR Analytics?**

Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analytics does not just deal with gathering data on employee efficiency. Instead, **it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.**

**What is attrition in HR?**

Attrition in human resources refers to the gradual loss of employee’s overtime. In general, relatively high attrition is problematic for companies. HR professionals often assume a leadership role in designing company compensation programs, work culture, and motivation systems that help the organization retain top employees.

How does Attrition affect companies? and how does HR Analytics help in analyzing attrition? We will discuss the first question here and for the second question, we will write the code and try to understand the process step by step.

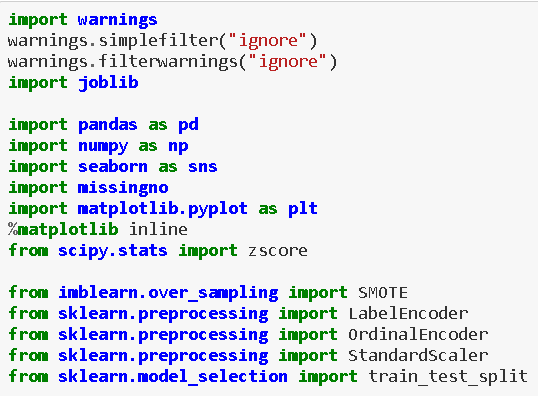
**Attrition affecting Companies**

A major problem in high employee attrition is its cost to an organization. Job postings, hiring processes, paperwork, and new hire training are some of the common expenses of losing employees and replacing them. Additionally, regular employee turnover prohibits your organization from increasing its collective knowledge base and experience over time. This is especially concerning if your business is customer-facing, as customers often prefer to interact with familiar people. Errors and issues are more likely if you constantly have new workers.

Hope you got the basics. Let’s move towards coding part and try finding out how HR Analytics help in understanding attrition.

**Data Preparation**

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.



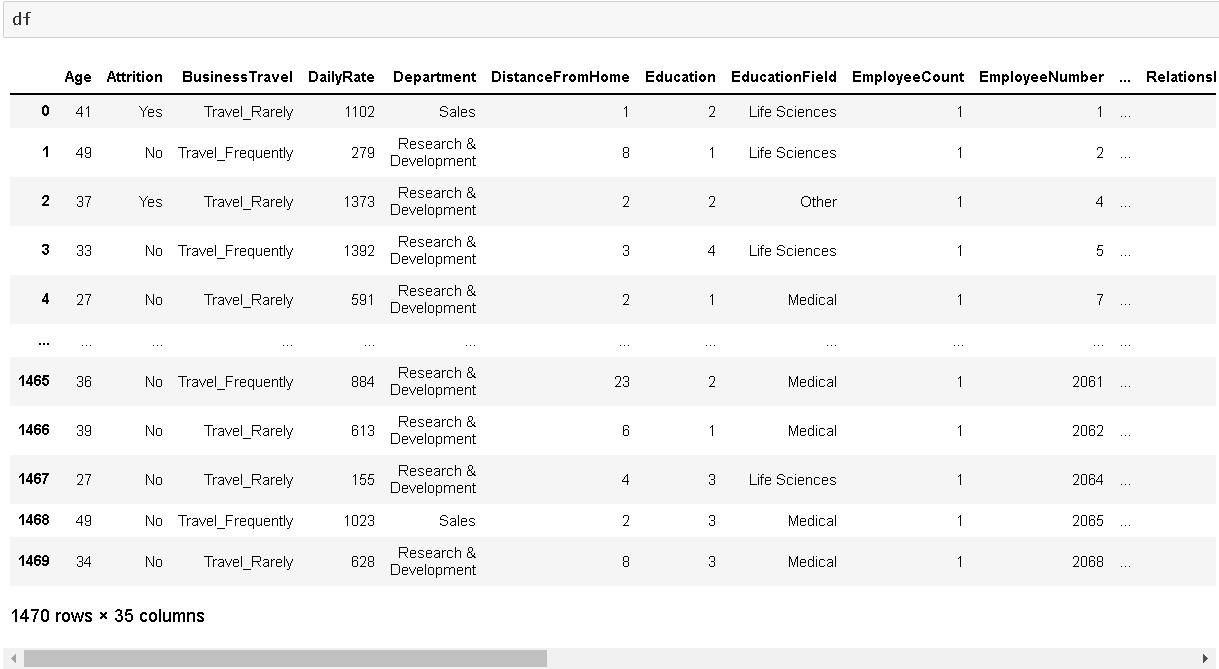
And now, we need to get the dataset in our Jupyter Notebook that can be achieved by a single step.



This gives us our data set stored in the variable “df” for our dataframe.

**Data Analysis**

For data analysis, let’s take a look on our dataset.



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



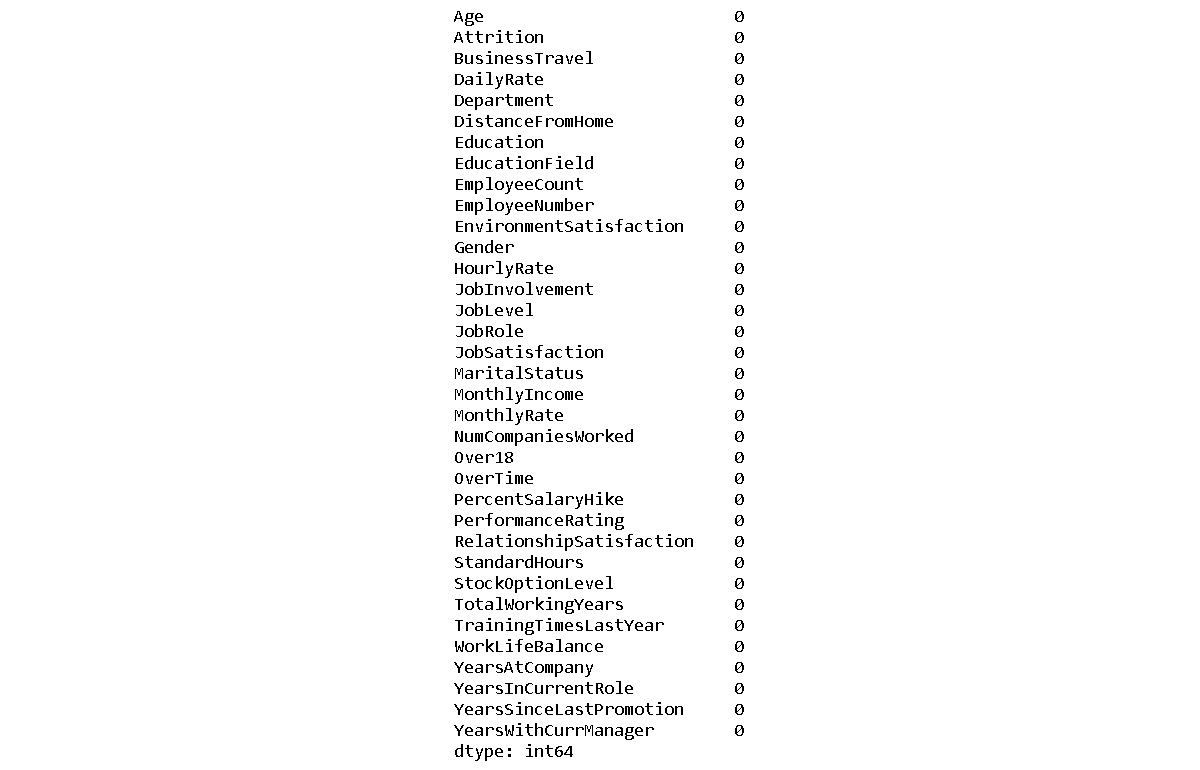
You can write this piece of code and then try to check the dataframe “df” again. It will show you the entire row and column information on your Jupyter Notebook directly.

**EDA**

For EDA also known as Exploratory Data Analysis first thing I am going to take a look at is the missing data information in our dataset by using the codes below

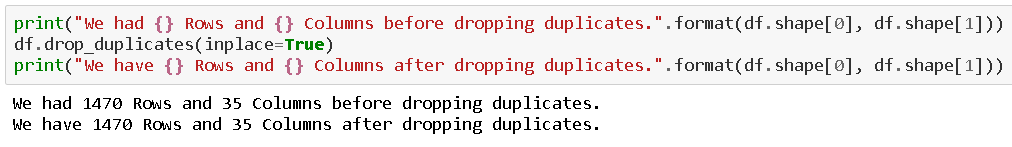
  


The above code gives us the missing values information column vise in tabular format.



Luckily we don’t have any missing value in our data set and from the above screenshot it looks that we don’t need to format the data.

Now we will remove any duplicate that might be present by using below code

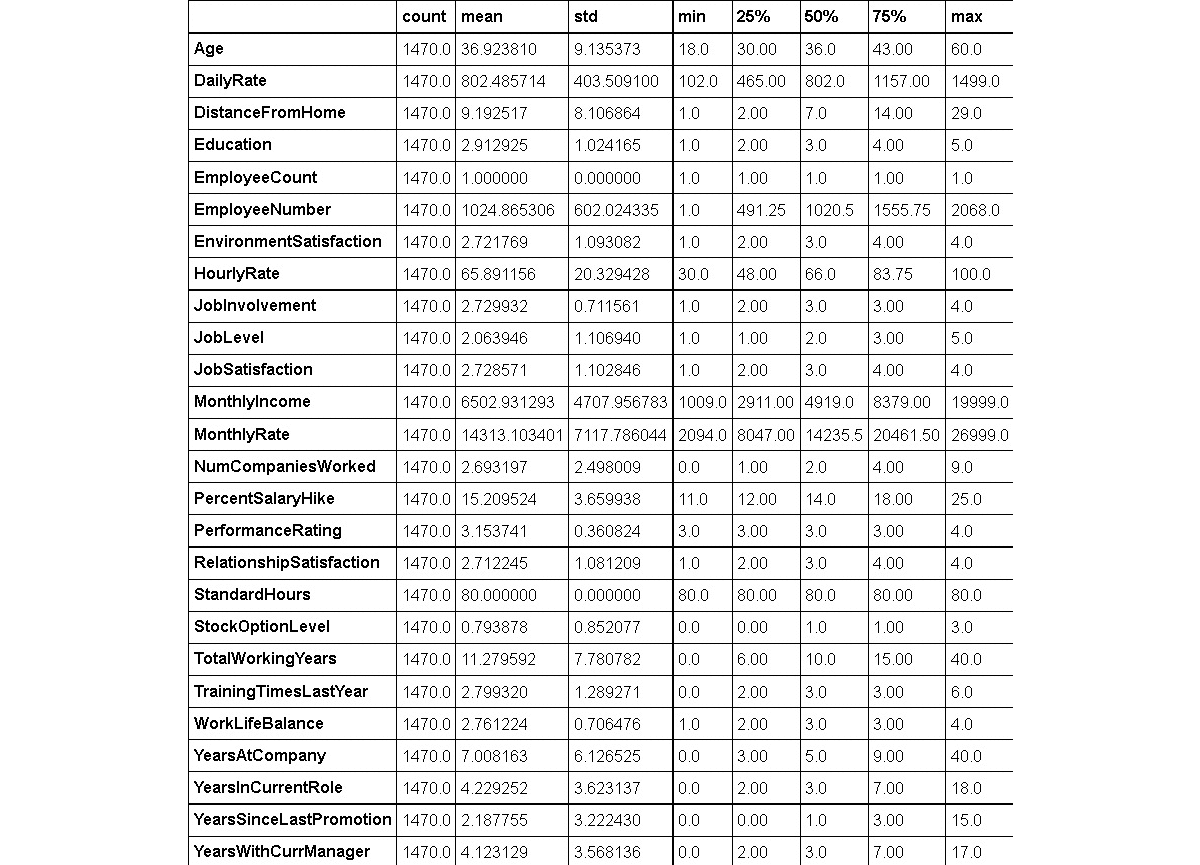


From the above code we conclude that there is no duplicate data present in our data set.

Now we use describe method to get the basic information like 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. Take a look at the below code and you will get an idea on how to use it.

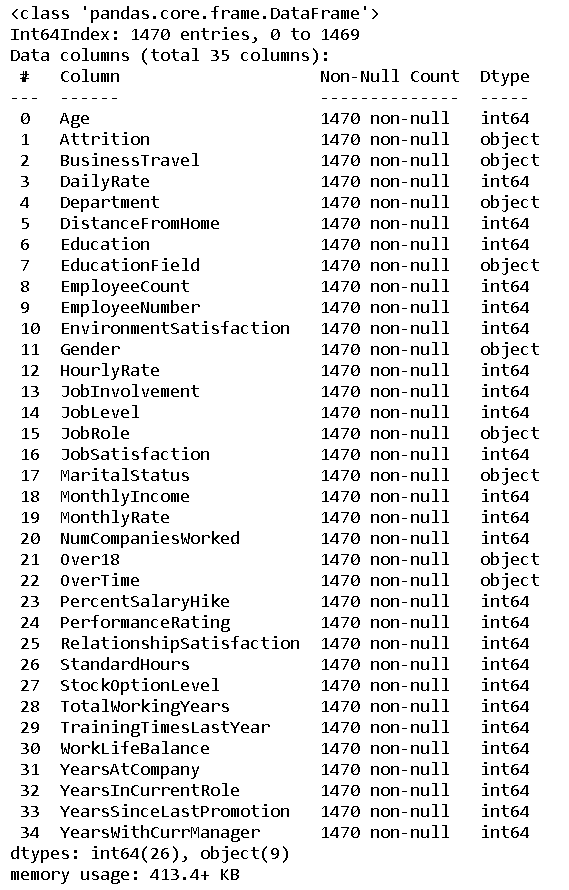


Once you have used the code the output provided are in transpose format to accommodate all the columns from our dataset in tabular as well as visual format.

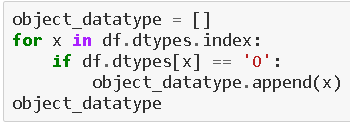


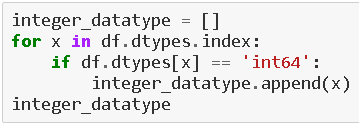
Now we are able to draw insights from describe method, now we can take a look on data type information by using below code.

  
  
 Output :



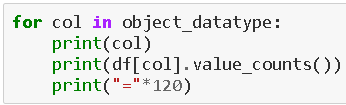
This is the output that I get explaining the data types of all the columns present in our dataframe. We also get an opportunity to drop or remove any unwanted columns from the dataframe here.  
  
One of the things that I like to do is separate the object data type and numeric data type values that allows for easier processing in further steps. The code to do that is a simple for loop usage.

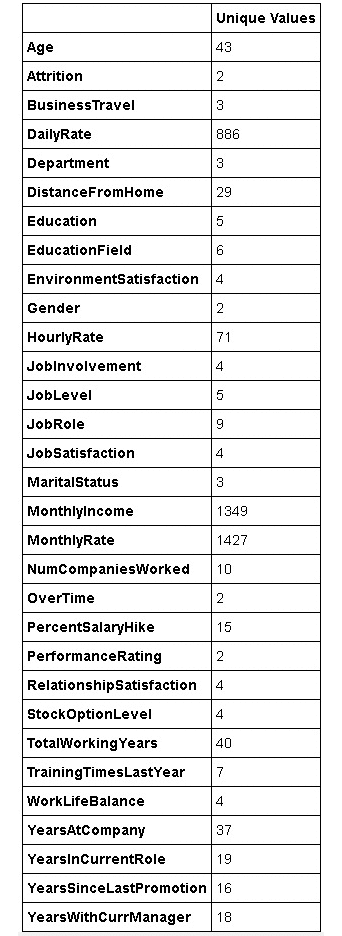




This allows us to store the column names in a list format within the variables namely object\_datatype and integer\_datatype.  
  
After I have bifurcated the datatype column names in two separate lists, we will take a look at the overall unique values for all the columns and then the data numbers for only object datatype columns using the below codes.





   
With the above code output we get an entire list of column names with unique data covered in the dataset rows providing a numerical data and then a description of those values for categorical object data type columns

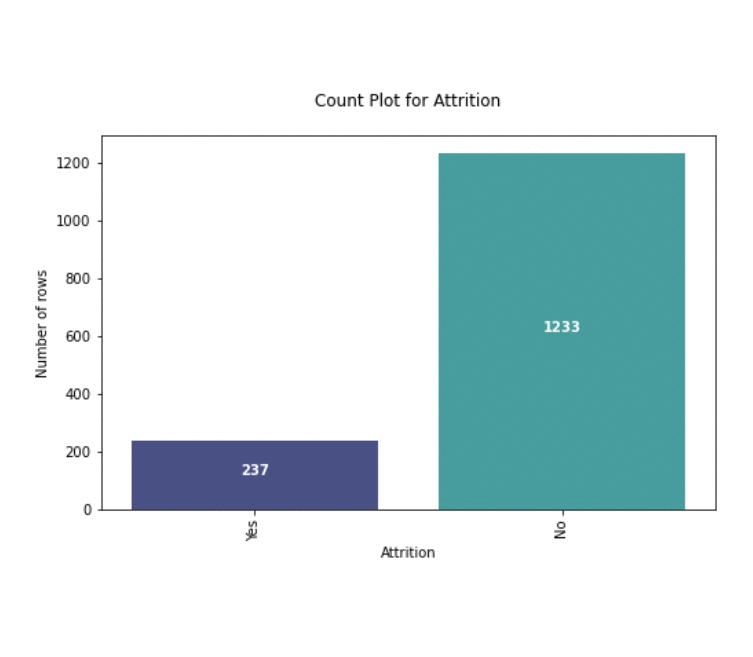
Considering the separation of object data I then take a visual on how many rows or count of rows these values cover in our data set. Usage of various visualization techniques allows me to optimize and analyze the columns further. It gives me an idea as to where data pre-processing will be needed and where removal of those data will benefit. Honestly all this can only be acquired from practicing on different projects and as everyone says the more you work the more you acquire knowledge in that field working like a 6th sense in such project creation. I am giving this example here but it does not mean that these are the only steps when it comes to creating a project. The architecture or the backbone of the project will remain the same however depending on what data you are working on the usage of techniques with all differ.

Now let’s move ahead and list down all the visualization codes and their output for our reference

Code:

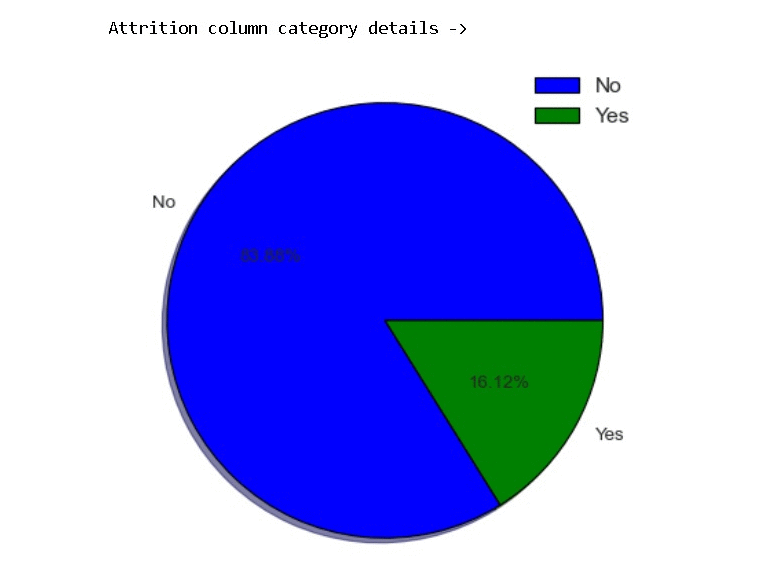


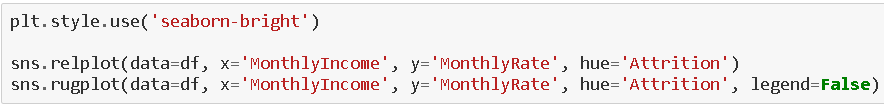
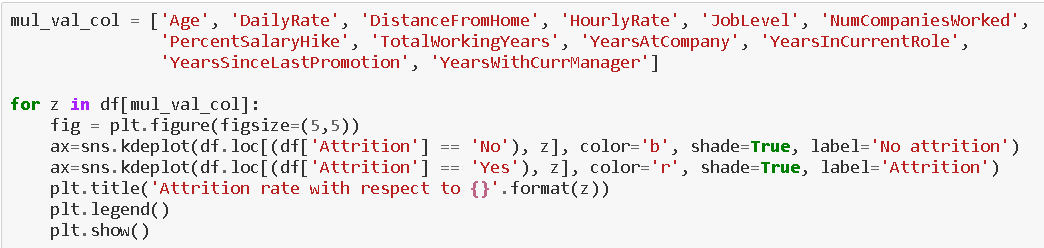
Output:

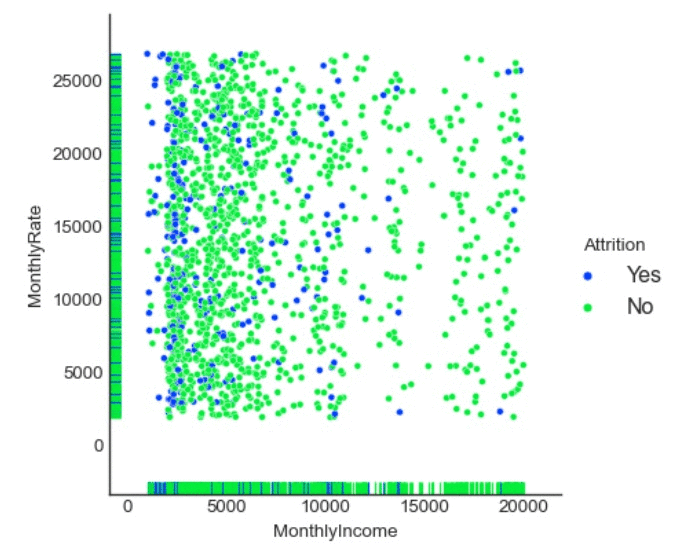


Code:

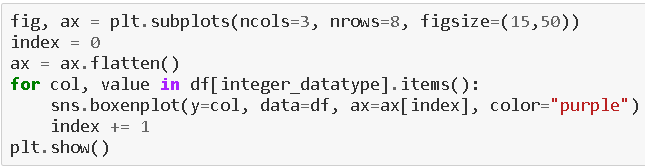
Output:



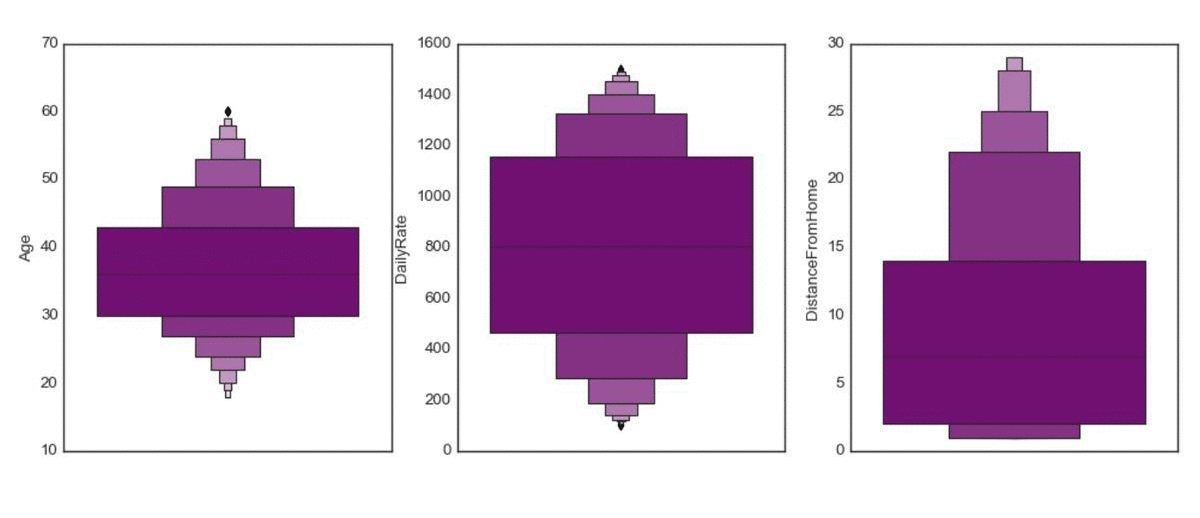
Codes:  
  


Output:  


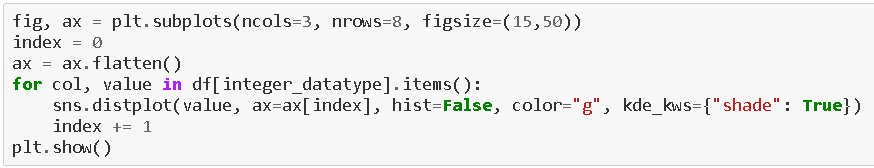
Code:



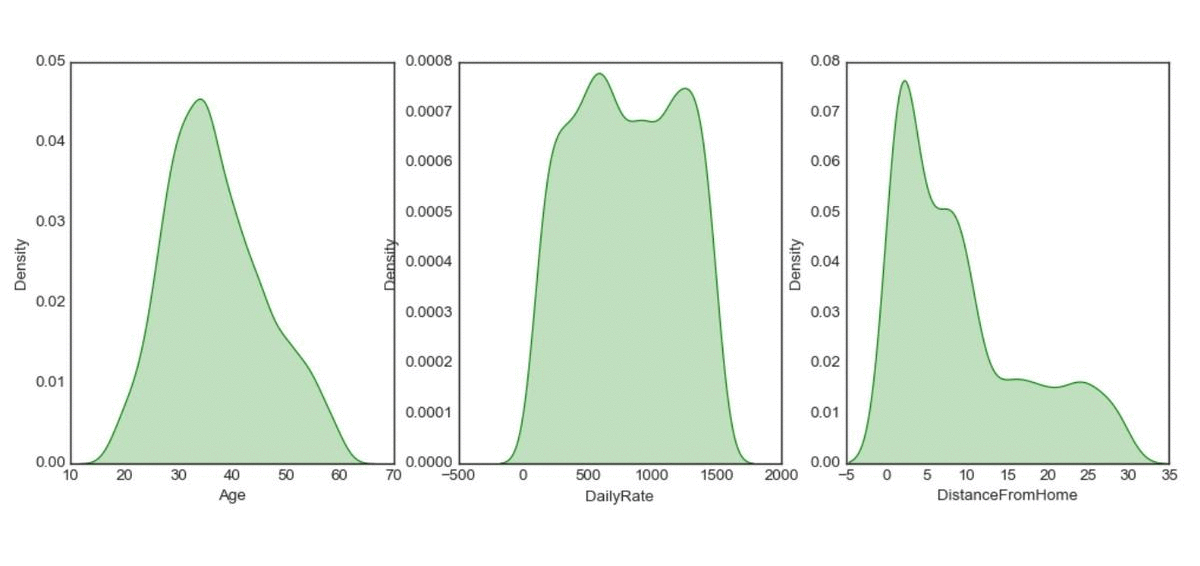
Output:



Code:



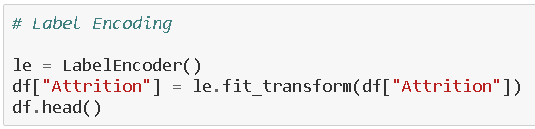
Output:

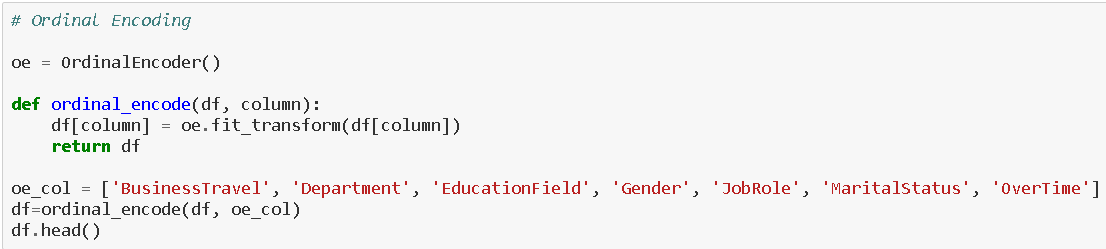


You can see that with the help of above codes and getting the outputs I was able to take a look at all the column values/counts, the boxen plots gave me a view on the presence of outliers and the distribution plots showed me the skewness information that will needed to be treated. These are like the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

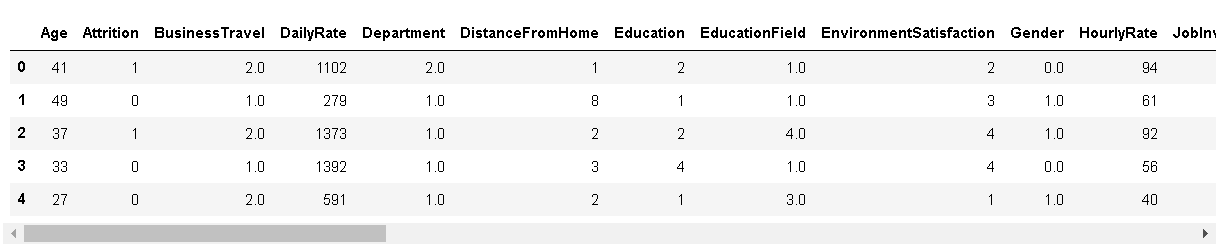
**Pre-Processing Data**

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 data type 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 data type 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.

Code:

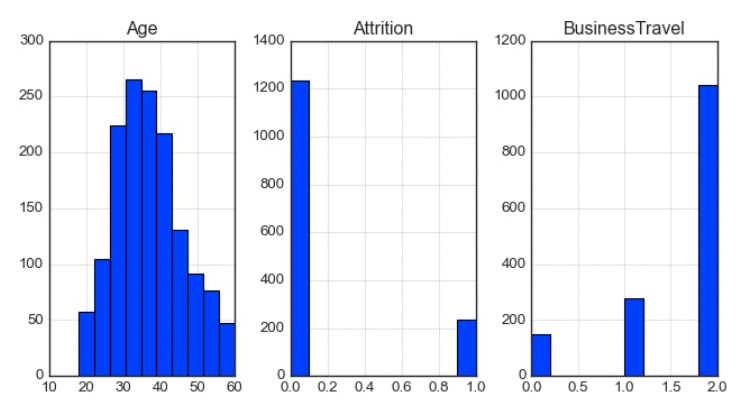


Output:

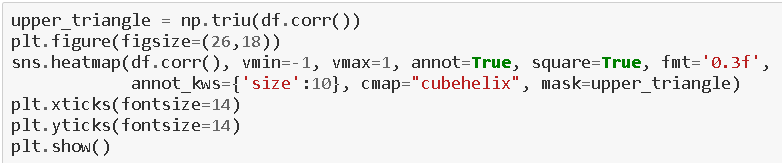


After I have encoded all the columns in our dataset, I am using a Histogram to view the data distribution. Since Histograms only consider numeric data it should be able to identify all the information from our encoded dataframe

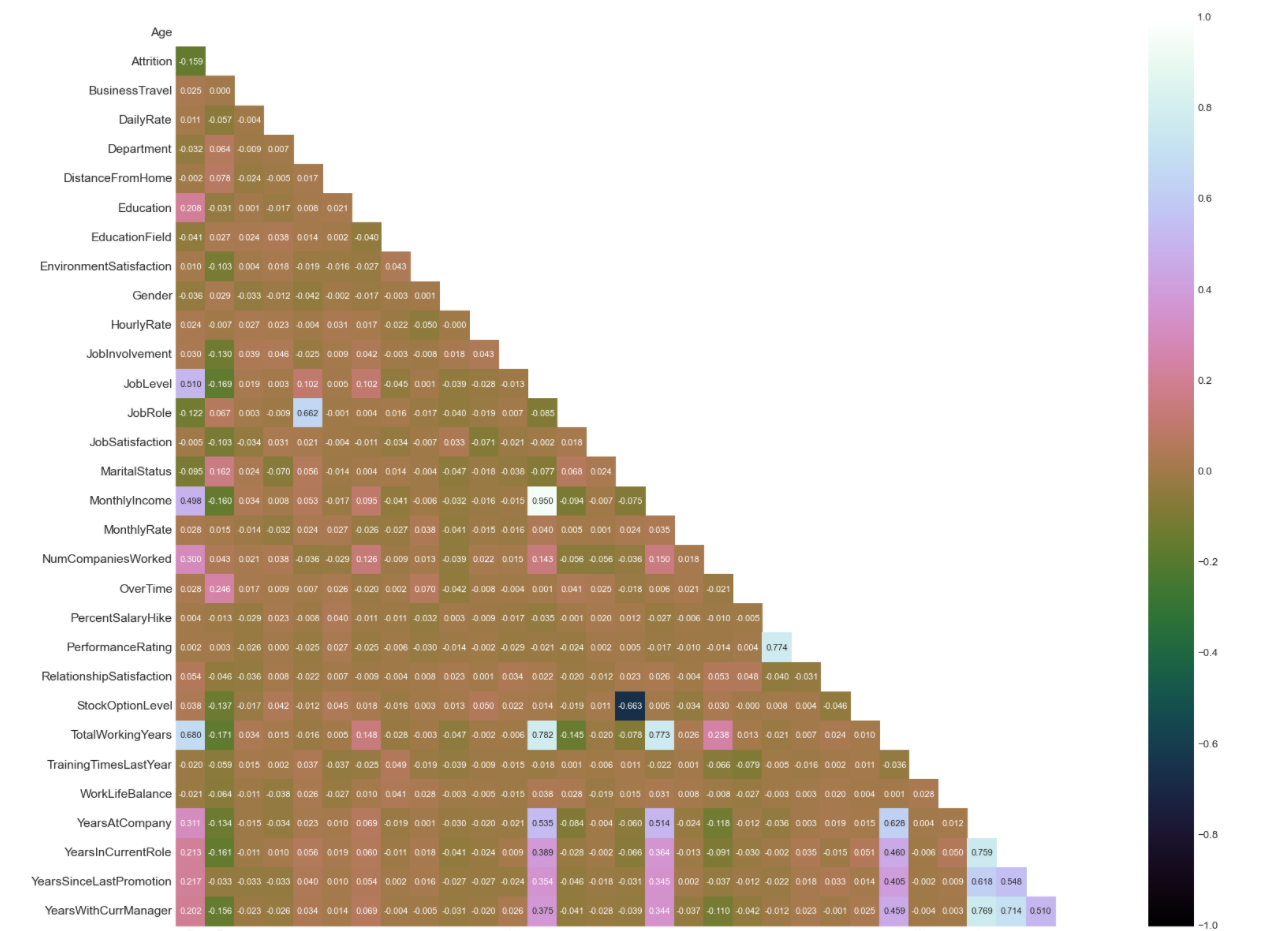
Code:

Output:

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.

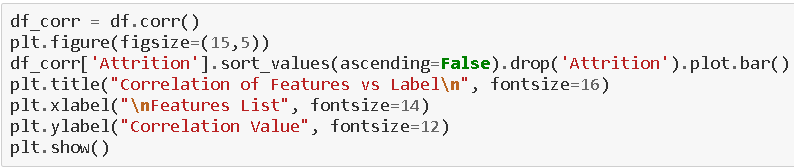
Code:  


Output:

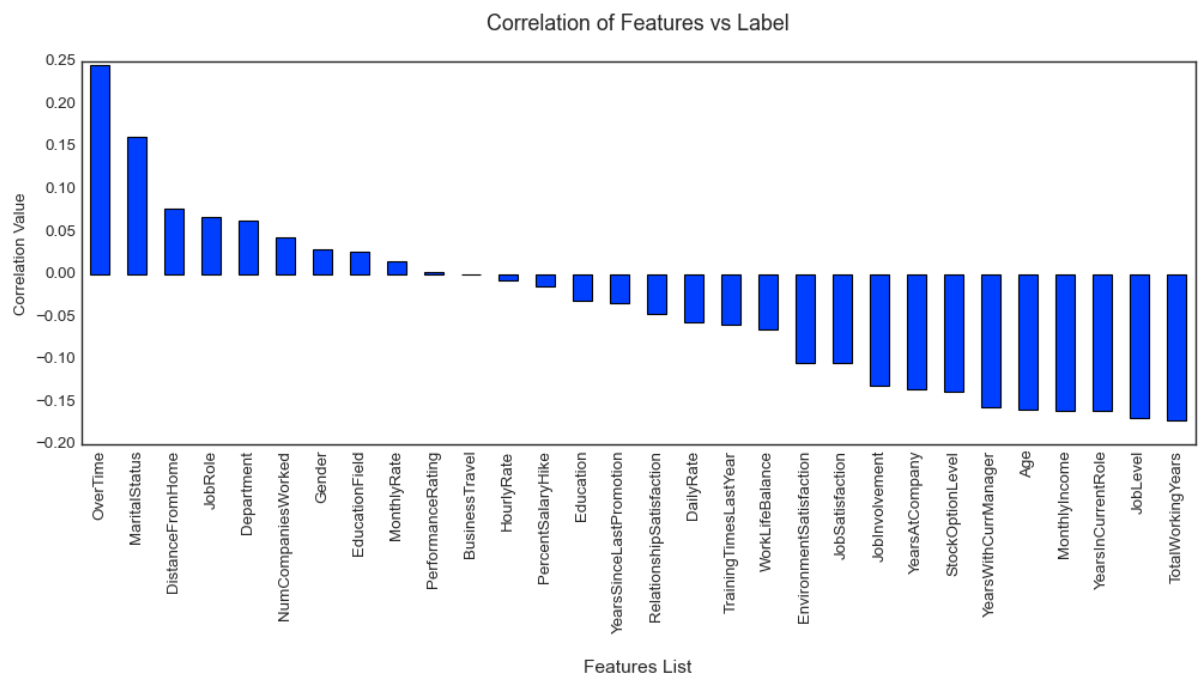


So, in such scenarios I majorly look at the colors to understand if there is any multicollinearity issue among the feature columns and if there is still any column that I can drop. But to clearly view the correlation between our label and feature columns I use a Bar Plot comparison and you can find its code here.

Code:

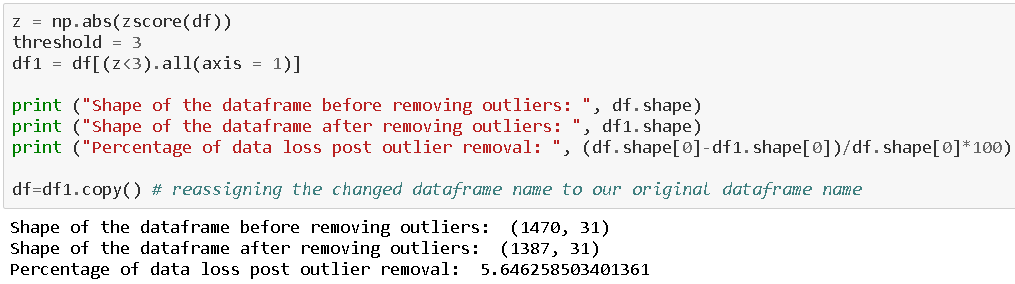


Output:

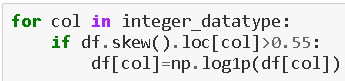
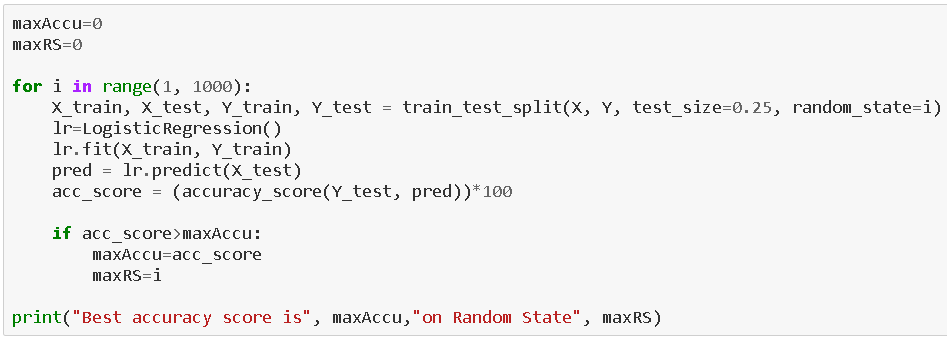
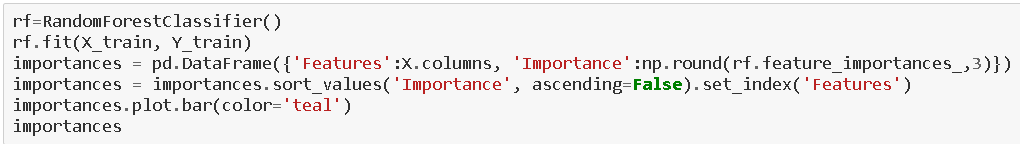


In the above Bar Plot we are able to clearly define the feature columns that are positively correlated with our label and the feature columns that are negatively correlated with our label. Now coming back to the outlier and skewness concern in our dataset I will be using the Z score and Log transformation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

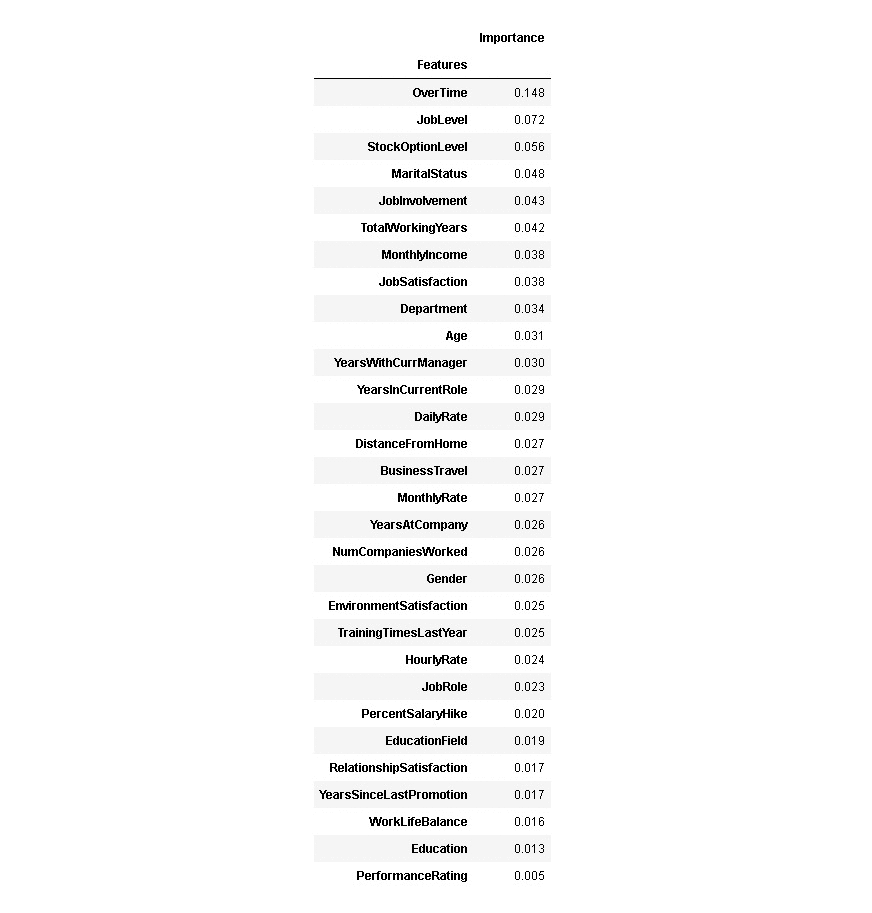
Code:



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

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


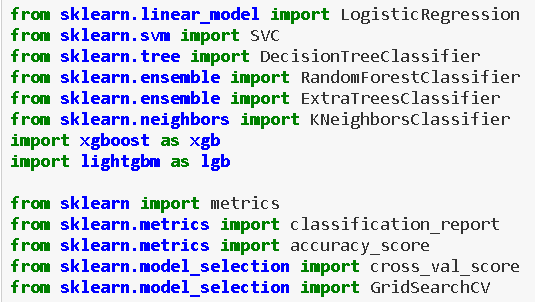
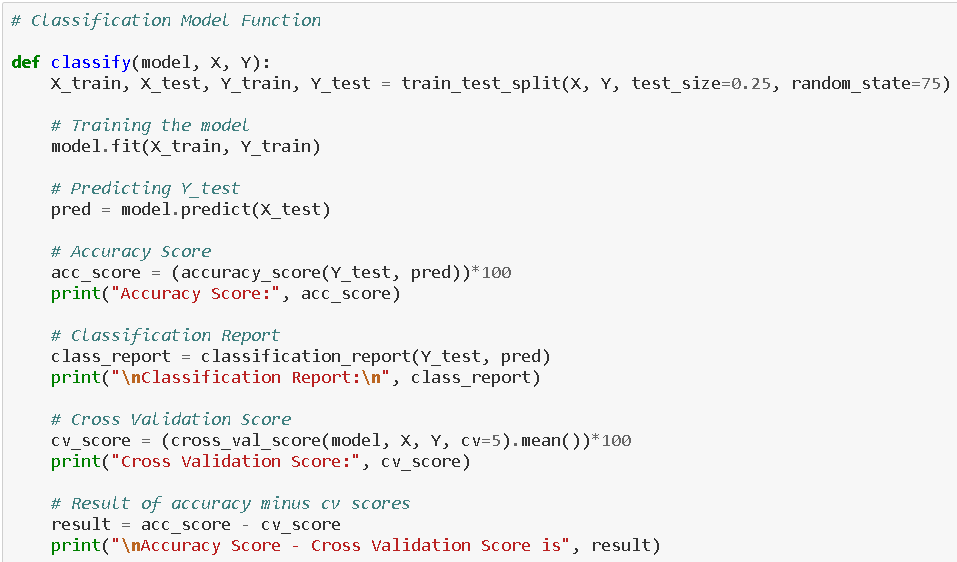
Output :

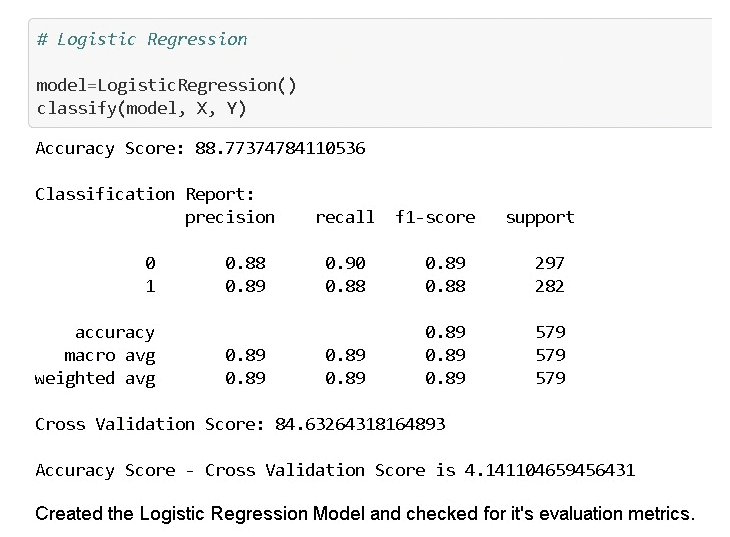


Once we have invested enough time in doing EDA and Pre-processing our data comes the step for which all the previous hard work was performed. That is to finally start building our Machine Learning model for classification purpose.

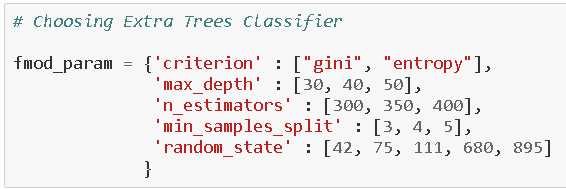
**Building Machine Learning Model**

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



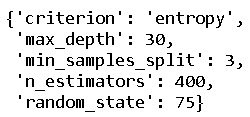


It is always advisable to build more than 5 machine learning models so that you can choose from the best performing model and then apply hyper parameter tuning to make it perform even better. I am going to use the Extra Trees Classifier as my choice of classification model as I see it is doing better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

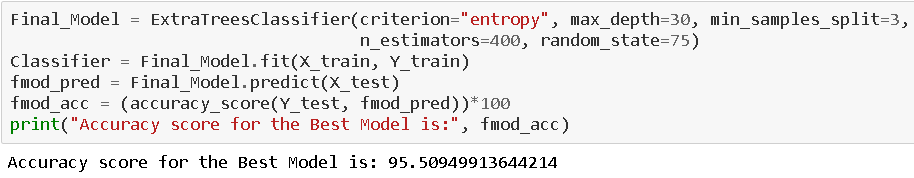
Code:  


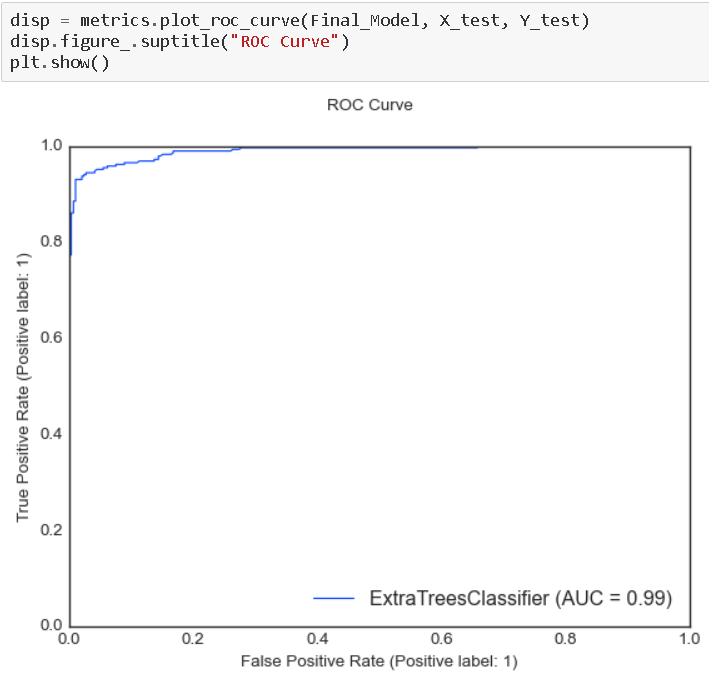
  


Output:



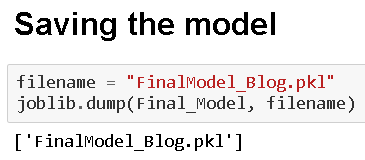
After applying the above steps to get the best parameters list, I simply have to plug it into my final model and receive the output of it. I have created an ROC curve plot and Confusion matrix for the final model.

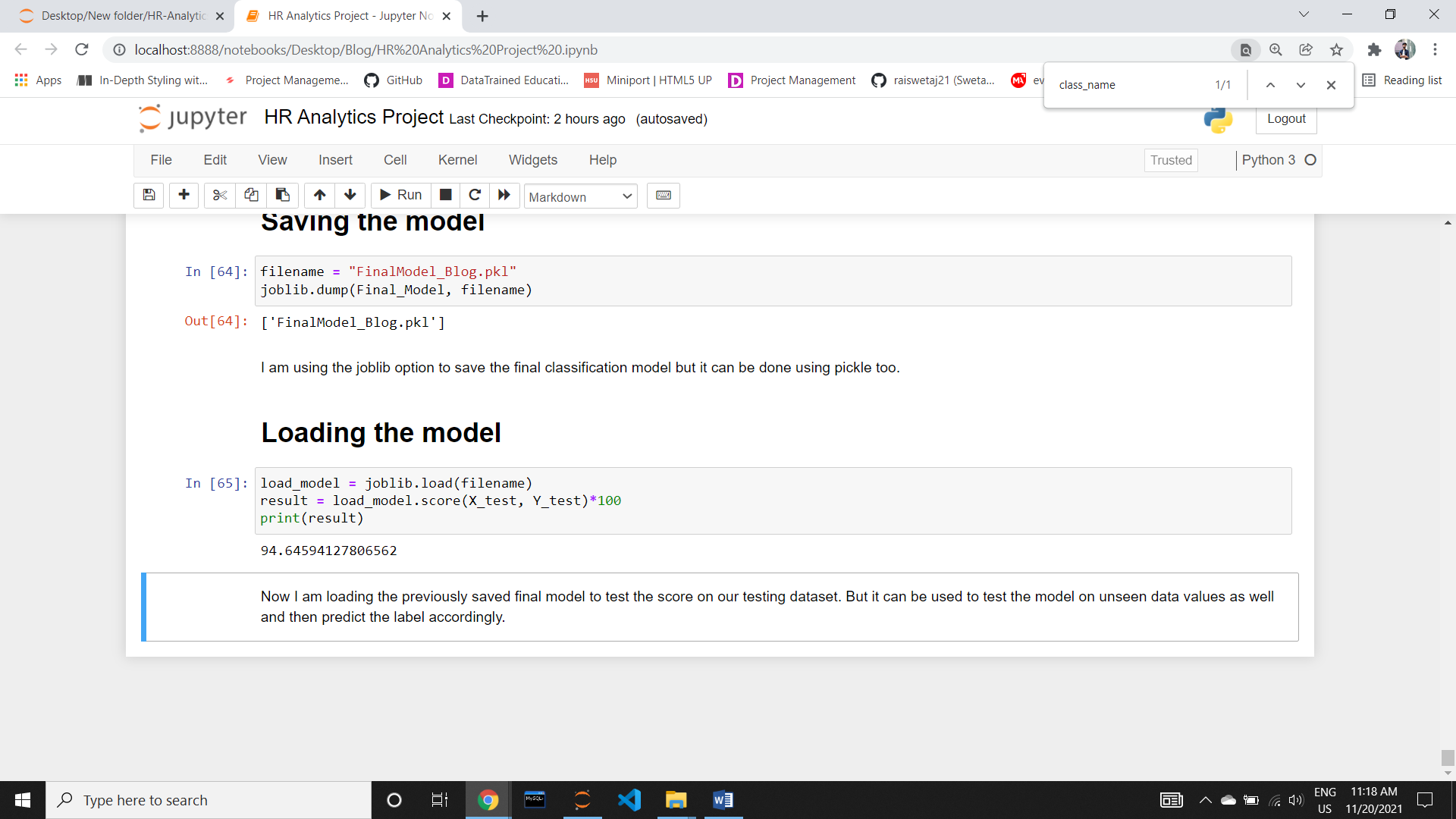






Accidentally I forgot to edit the print statement for my confusion matrix and it still shows the earlier Decision Tree model name instead of the Extra Trees model name and has the values stored for the latter. This also proves that your best performing model can keep changing at times even without changing your code and simply running it multiple times. But I am sure you can understand that the print statement can be changed as per your liking and the important stuff was on the utilization of the code showing the correct result.  
  
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.





**Concluding Remarks**

Throughout this post, we saw Data is important in Human Resource department (actually in most of places it is important). We saw how we can avoid using correlated values and why it is important not to use those while modelling. We used Random forest and learned how it can be very advantageous over other available machine learning algorithm. Most of all we found factors which are most important to employees and if are not fulfilled might lead to Attrition.

**References**

1. Github
2. Kaggle Dataset
3. HR Attrition
4. HR Analytics