***Human Resources Analytics Blog***

End-to-end machine learning project in Data Science.

# Project Name: HR Analytics Project- Understanding the Attrition in HR



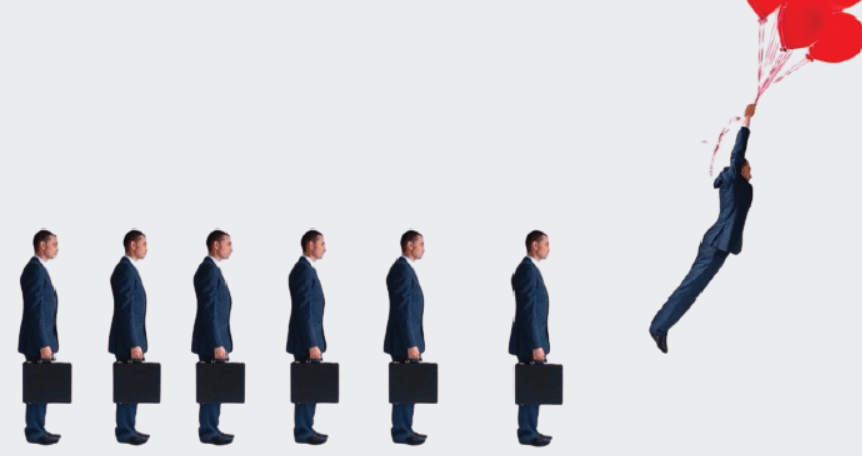
*"I had to make sure I didn't forget any of the crucial things to develop a comprehensive ML project, so this is a wee bit lengthy essay, but you have surely made the correct option to invest your valuable time in something instructive here."*

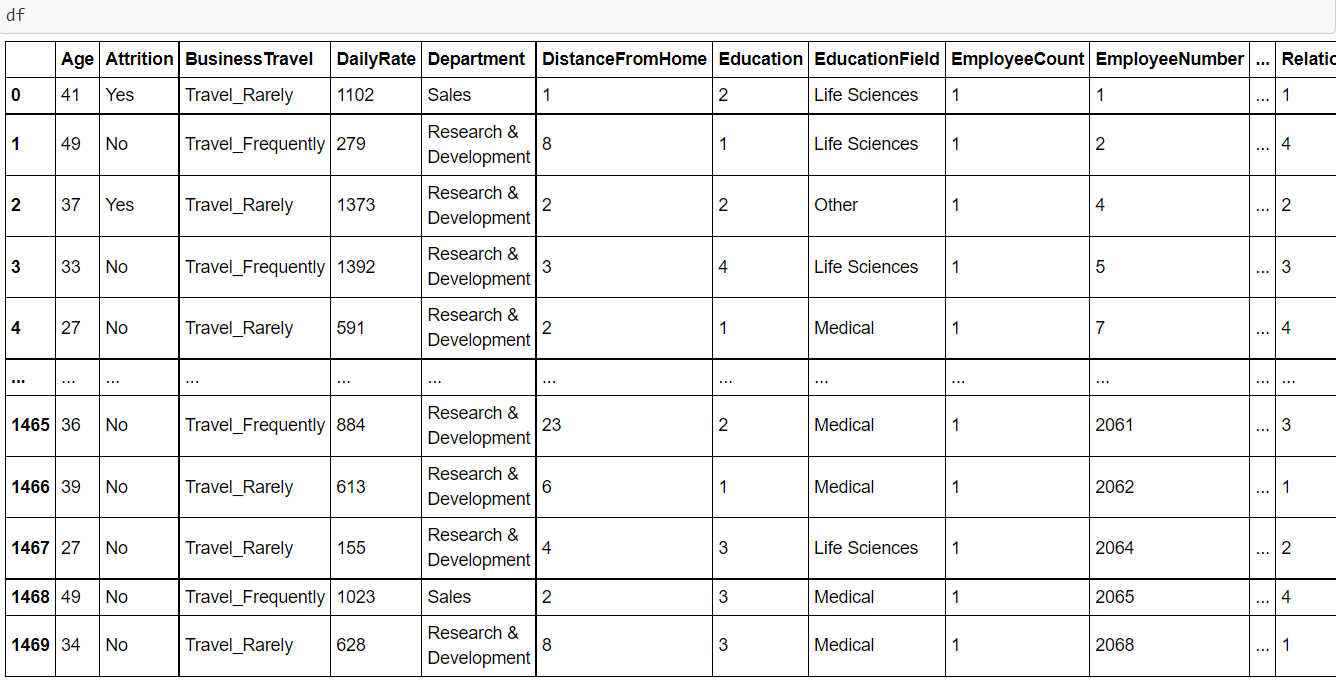
This is a full end-to-end HR Analytics project. Many prospective Data Scientists should use this as a guide. When I initially began studying Data Science, I was overwhelmed by the quantity of information accessible online, which was largely separated into parts highlighting deeper knowledge that a novice would be unable to understand or connect with. I agree that many projects are currently available on the deep web, hidden behind numerous clicks and pathways, like the core of an onion covered by multiple layers. But here, I'm simply doing my part to make it easier for a newcomer to comprehend the basic architecture that's needed in the real world to build a Data Science project.  
  
Please allow me to clarify the agenda for this blog post without further ado. I've broken down all of the strategies into sub-topics in this post, which I'll go through one by one. The following are the pointers:

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

Let's start with a definition of the problem or a brief overview of the project I've decided to expound on and why it was created in the first place.

1. **Problem Definition**

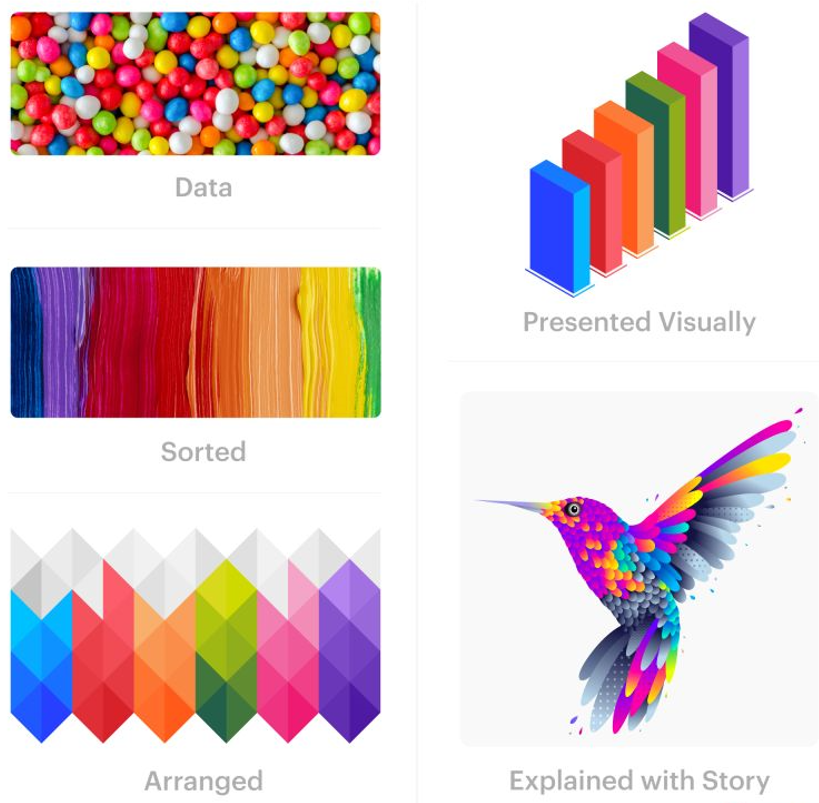
The project I'm utilising for this essay is for HR Analytics, which is essentially a made-up dataset developed by IBM's Data Scientists. The dataset may be found on the Kaggle website. Every year, a large number of firms hire new staff. Not only do the organisations devote time and money in educating those people, but they also have internal training programmes for its current personnel. The goal of these initiatives is to make their personnel more productive. Where does HR Analytics, on the other hand, fit into all of this? Is it only about increasing staff productivity?  
  
Human Resource Analytics (HR Analytics) is a subset of analytics that involves applying analytic techniques to an organization's human resource department in the hopes of enhancing employee performance and thereby increasing return on investment.  
  
  
  
The steady loss of employees' overtime is referred to as attrition in human resources. In general, a high rate of attrition is an issue for any business. HR specialists frequently take the lead in developing firm compensation schemes, work cultures, and motivating systems that aid in retaining top talent. How does attrition effect businesses, and how can HR Analytics help with attrition analysis? We will cover the first question here, then we will build the code and attempt to comprehend the process step by step for the second question.  
  
Because excessive staff turnover is a cost to a company, attrition impacting businesses is a serious issue. Job listings, recruiting processes, documentation, and new hire training are all standard costs associated with losing and replacing staff. Furthermore, high personnel turnover makes it impossible for a company to grow its collective knowledge and expertise over time. This is especially problematic if your company is customer-facing, as customers like to connect with individuals they know. Errors and problems are more frequent if you're continuously hiring new people.  
  
As a result, the primary purpose of this project is to identify the "Attrition" rate as a simple Yes or No tag, therefore posing a classification challenge!  
  
  
  
First, we'll import all of the necessary dependencies that will be utilised in our project, and then we'll get the remainder as needed. Before we start any procedure, we need to get the dataset into our Jupyter Notebook, which is a one-step process.

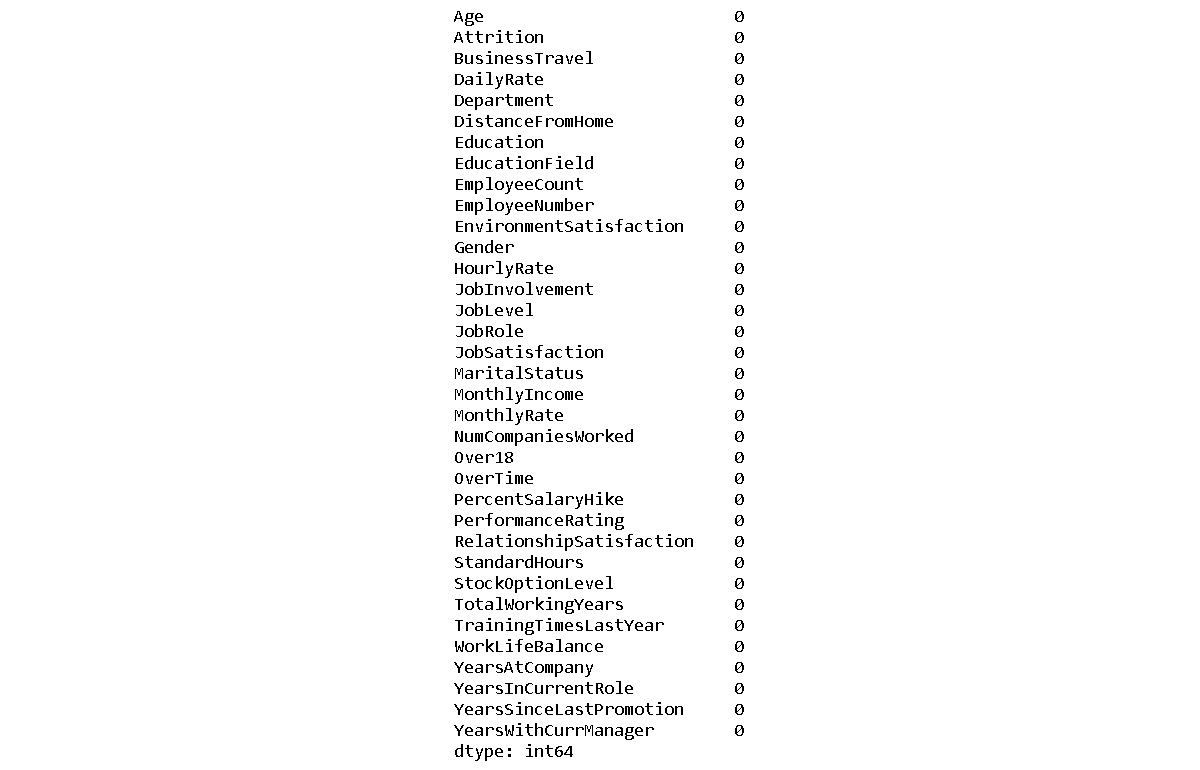
  
  
Our full dataset is now saved in the variable "df" for our data frame.  
  
**2. Data Analysis**  
  
We can just eyeball the contents of our dataset for data analysis, trying to make sense of certain columns, their connected values, and anything else that comes to mind.  
  
  
  
The entire number of rows in our data is 1470, and the total number of columns is 35, as seen in the above line of code. The visualisation is shortened since the dataset has a rather large number of rows and columns.  
  
  
  
You may try checking the data frame "df" again after writing this piece of code. It will display the whole row and column information straight in your Jupyter Notebook.  
  
**3. EDA Concluding Remark**

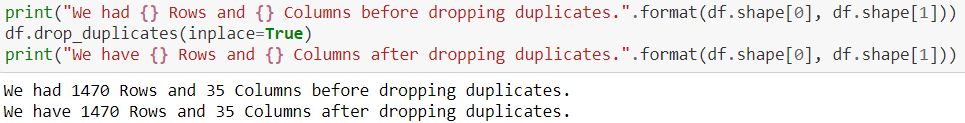
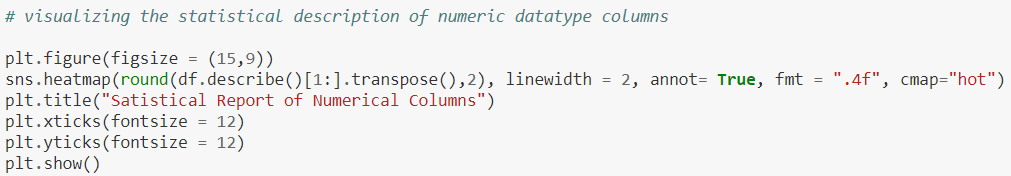
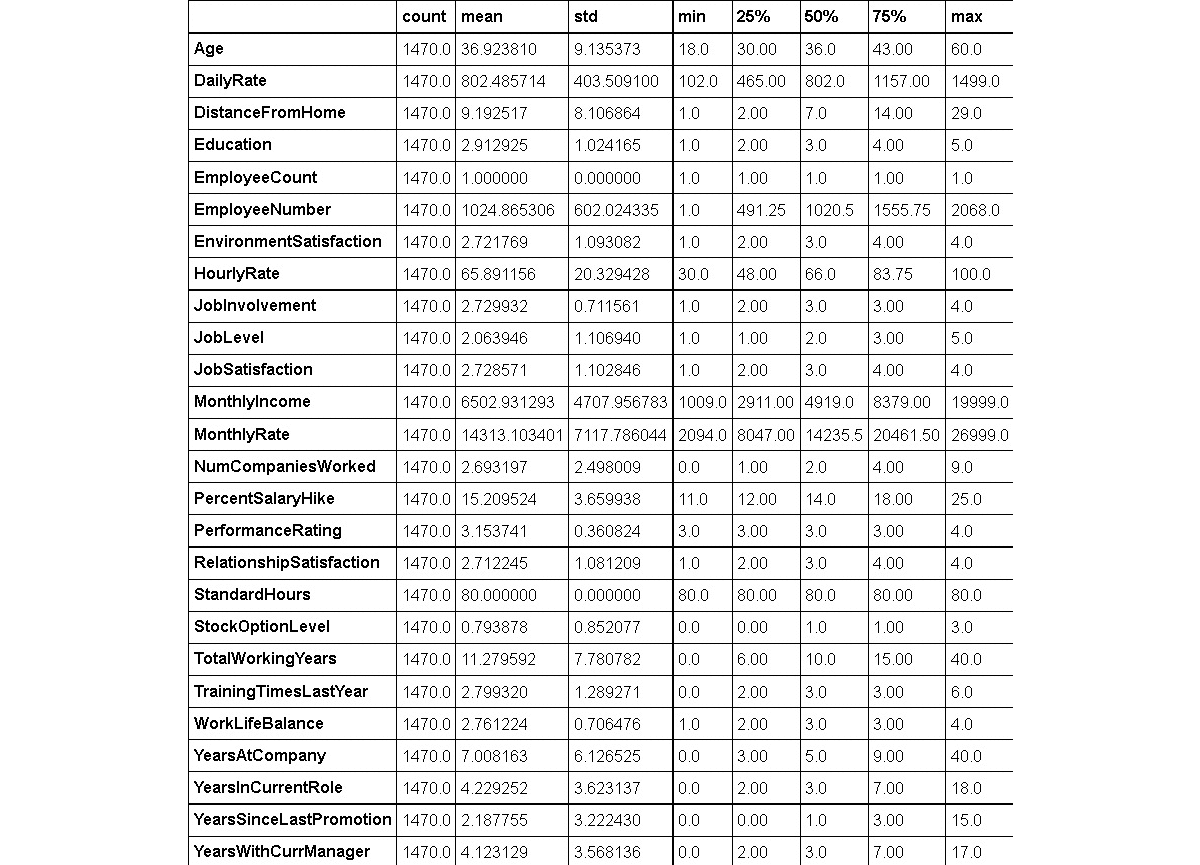
Many Data Scientists, including myself, believe EDA, also known as exploratory data analysis, to be the most significant part of Data Science. After following a large number of expert Data Scientist on various platforms, I can confirm one thing: it boils down to a single important thing: conveying a storey about how you were able to achieve each and every step via your code, including the provided problem statement, observation, challenges encountered, and what was done to address or rectify those issues.  
  
Only when you fully understand what you're doing and why you're doing it can you build a decent model. Ascertaining that you have clean data in a properly processed format to feed into your model and obtain the desired outcome. Because no amount of Machine Learning model use or hyper parameter tuning will assist until you spend time cleaning up and fixing your data, which is the only input you have.

There is a well-known phrase among Data Scientists and those who work with data that goes something like this: "Garbage in, garbage out." This simply means that if you are attempting to develop an automated model or make sense of data, you must fully clean and work on it before proceeding. I cannot emphasise the need of collecting clean data enough, because there is no such thing as clean data in the actual world. That's where data analysis comes in, and employing a variety of methodologies ranging from data cleaning to data pre-processing to data engineering helps us get closer to our intended label prediction.

Enough with the stories; let me show you the code, because talk is cheap, but it is sometimes important to illustrate what is actually going on. Using the scripts below, the first thing I'm going to look at is the missing data information in our dataset.

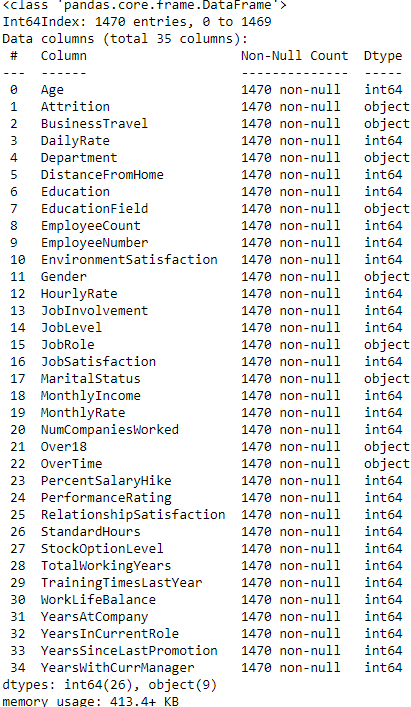
  
  
  

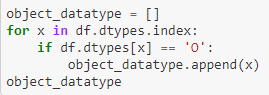
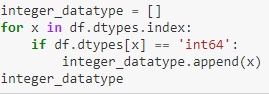

These two programmes provide a tabular and visual representation of the missing values information that looks like this.  
  
  
  
Now that we've confirmed that our dataset is clear of any missing data, we'll use the code below to remove any duplicates that may exist.

  
  
I was attempting to eliminate all duplicate data in our dataset using the drop duplicates option. However, we can observe that our dataset does not include any duplicate data. After that, we'll use the describe method to examine the count value, mean data, standard deviation information, and the minimum, maximum, 25% quartile, 50% quartile, and 75% quartile features. Because the describe method is suitable for numeric data, it ignores any object (text) data. Take a look at the code below for an example of how to utilise it.  
  
  
  
  
After you've run the code, the output will be in transpose format, which will accommodate all of the columns from our dataset in both tabular and visual formats.  
  
  
  
When we've figured out what the describe function is telling us, we can look at the datatype information using the code below, which will give us a list of all the columns and designate them as integer, float, or object datatypes based on the values in the columns.

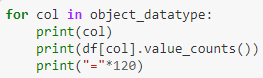
Code:

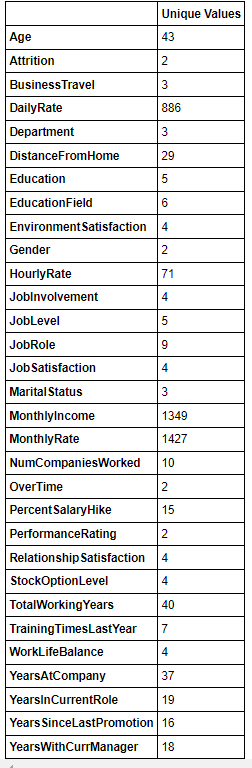




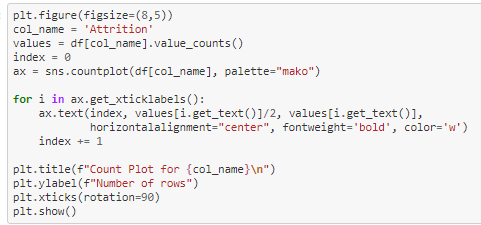
This is the result, which explains the datatypes of all the columns in our data frame. Here is also where we may eliminate or remove any unneeded columns from the data frame.  
  
Separating the object datatype and numeric datatype values is one of the things I prefer to do to make future processing easier. The code to accomplish this is a simple for loop.  
  
  


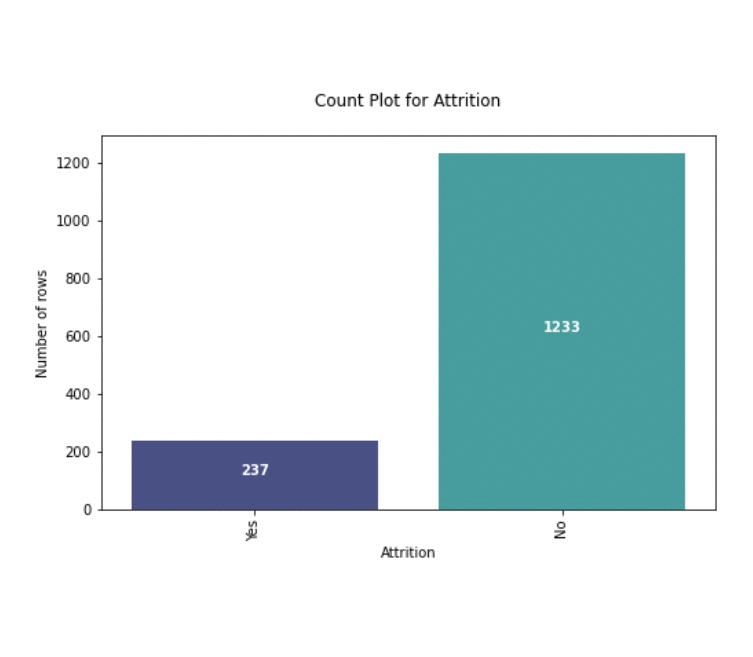
This allows us to store the column names in the variables object datatype and integer datatype in a list format.

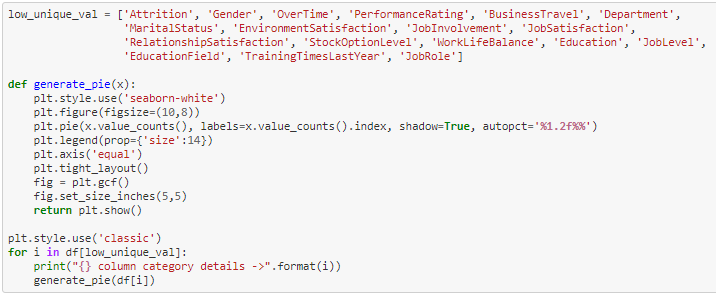
We'll look at the overall unique values for all the columns and then the data numbers for only the object datatype columns using the below scripts after I've split the datatype column names into two lists.  
  
  


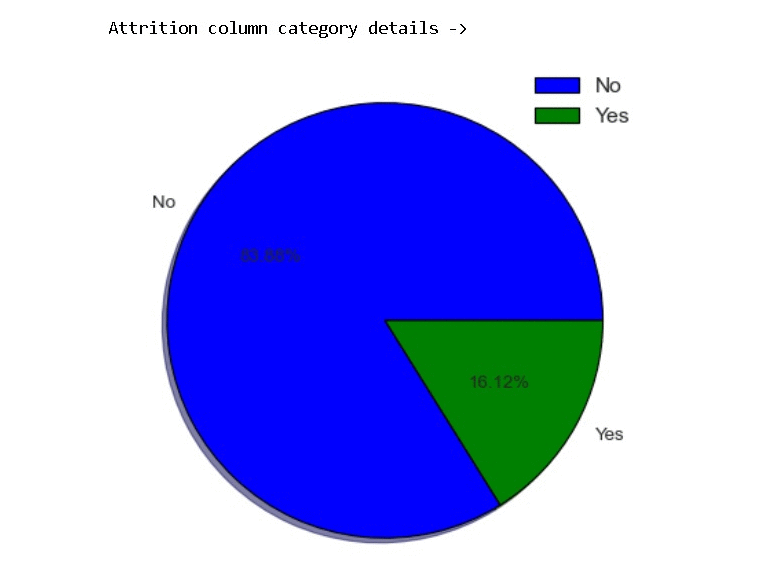


We acquire a whole list of column names with unique data covered in the dataset rows supplying numerical data and then a description of those values for categorical object datatype columns with these lines of codes.  
  
I then see how many rows or count of rows these values span in our data collection, taking into account the separation of object data. Using various visualisation approaches, I am able to further refine and analyse the columns. It offers me an idea of where data pre-processing will be required and where data removal would be beneficial. To be honest, all of this can only be learned by working on various projects, and as they say, the more you work, the more you learn in that subject, working like a sixth sense in project development. This is an example; however, it does not imply that these are the only phases in the project creation process. The project's architecture, or backbone, will remain the same, but the approaches used may vary based on the data you're dealing with.

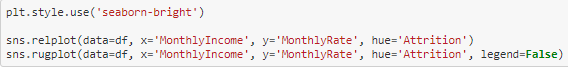
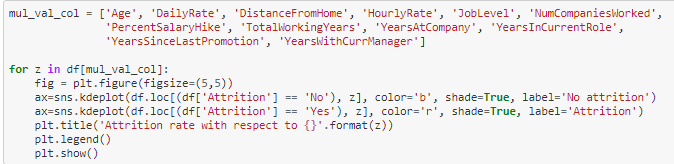
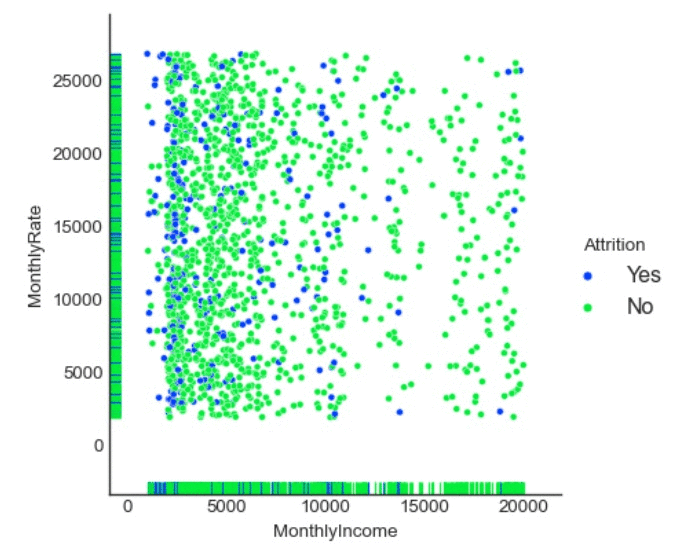
For example, because I did not receive any missing values in this project, I did not worry about handling them. However, there are datasets with a lot of missing data that are filled using various methods and are occasionally discarded as a last resort if it will only bias our machine learning model towards one data value or category.  
  
For your convenience, I'll go ahead and include all of the visualisation programmes and their output.  
  
Code:  


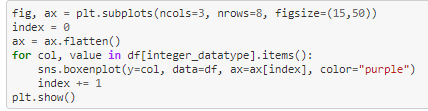
Output:  
  
  
Code:

  
  
Output:

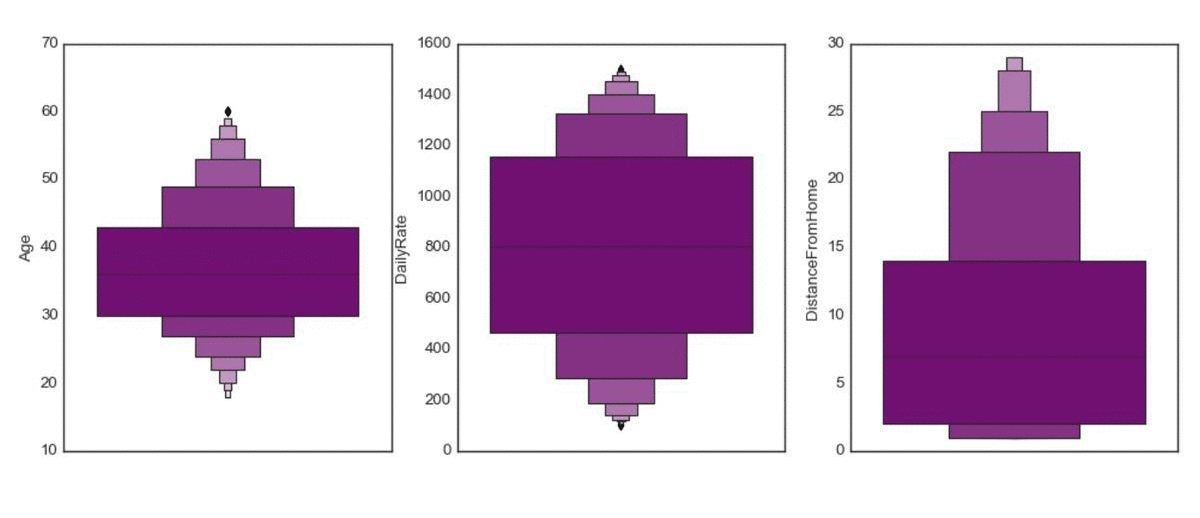


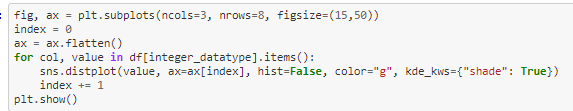
Codes:

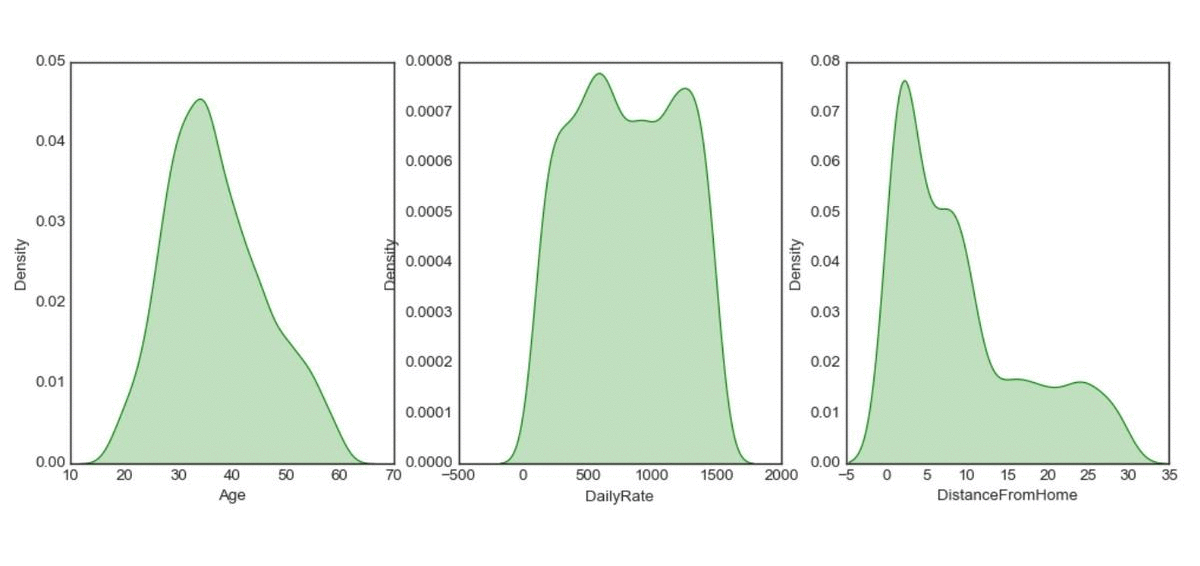
  
  
  
Output:  
  
  
Code:



Output:

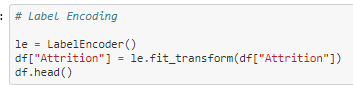
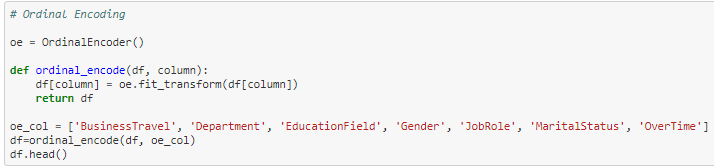
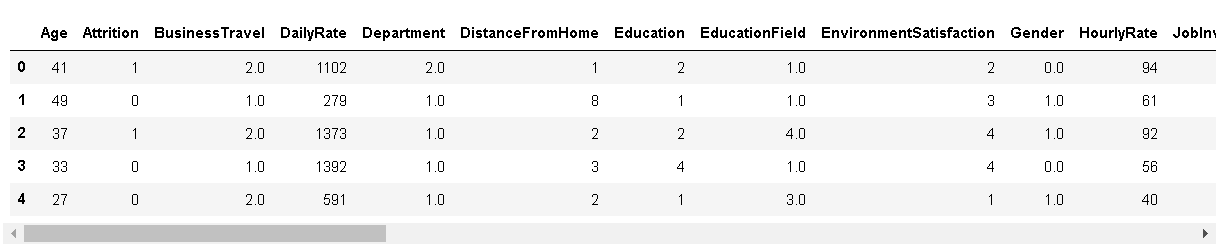
  
  
Code:

  
  
Output:

  
  
You can see that by using the aforementioned methods and obtaining the outputs, I was able to examine all of the column values/counts, the boxen plots revealed the existence of outliers, and the distribution plots revealed the skewness information that would need to be addressed. These are the obstacles I'll have to overcome before I can start creating my Classification Machine Learning models.

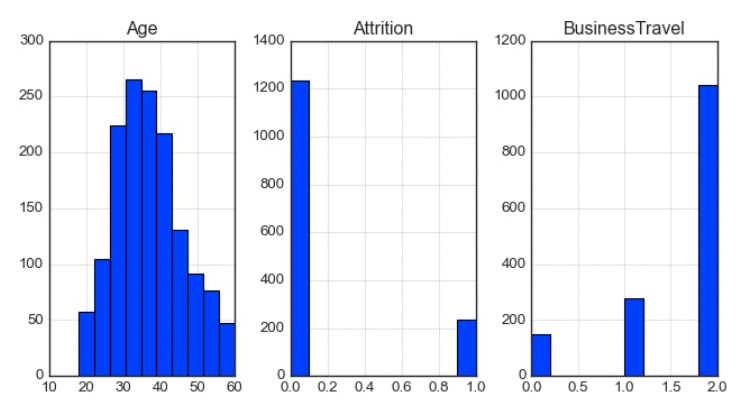
**4. Pre-processing Data**  
  
In the pre-processing stage, I'll address each mismatch one by one, starting with the issue that our dataset contains object datatype values; however, our Machine Learning models can only interpret numeric values. To transform all of the object datatype values, I'm using the encoding methods. I'm using Label Encoder for our label, and Ordinal Encoder for the other category feature columns. I could have utilised The One Hot Encoder instead of the Ordinal Encoder, but as I previously stated, it's all about personal choice and trial and error. The One Hot Encoder approach increases the number of columns, but using Ordinal Encoder on data values that give an order appears to me to be a preferable alternative. Many people have also used Label Encoder on feature columns, which I don't get because the name itself states Label Encoder. How much of a definition is necessary to realise that it is just for our label(s) columns? I hope none of my readers make the same mistake!

Code:

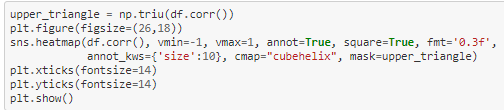
  
  
  
Output:  
  
  
I use a Histogram to see the data distribution once I've encoded all of the columns in our dataset. Because histograms only evaluate numeric data, they should be able to recognise all of the data from our encoded data frame.

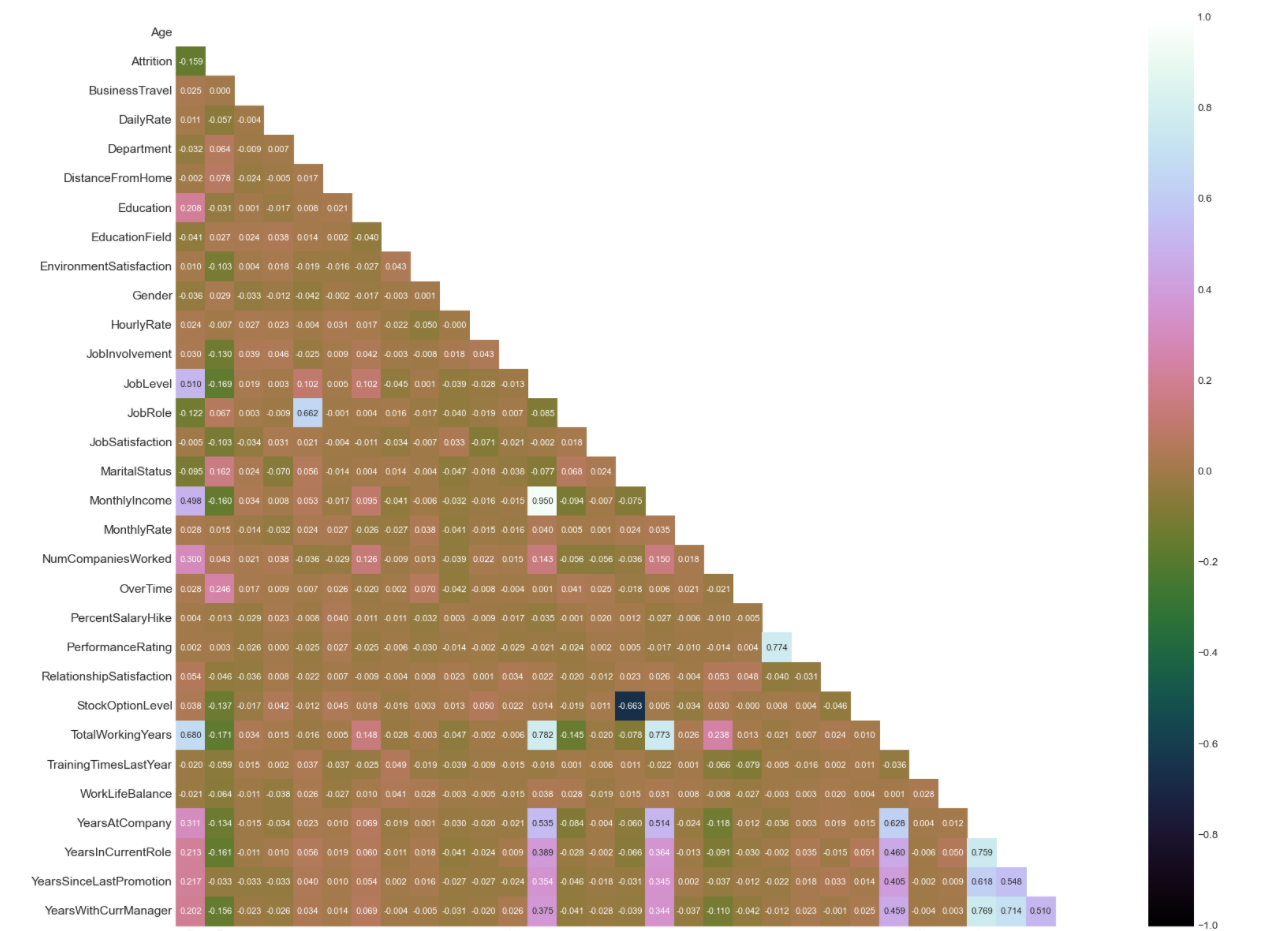
Code:

  
  
Output:

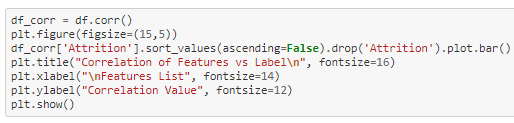
  
  
I now feel compelled to use a Heatmap to look for correlation information in our collection. For those who are still perplexed about the correlation specifics, let me explain it down into two simple points. Positive correlation - A correlation of +1 denotes a perfect positive correlation, in which both variables move in the same direction at the same time. A negative correlation of –1 implies a complete negative correlation, which means that when one measure rises, the other falls. The code to see this data may be found below.

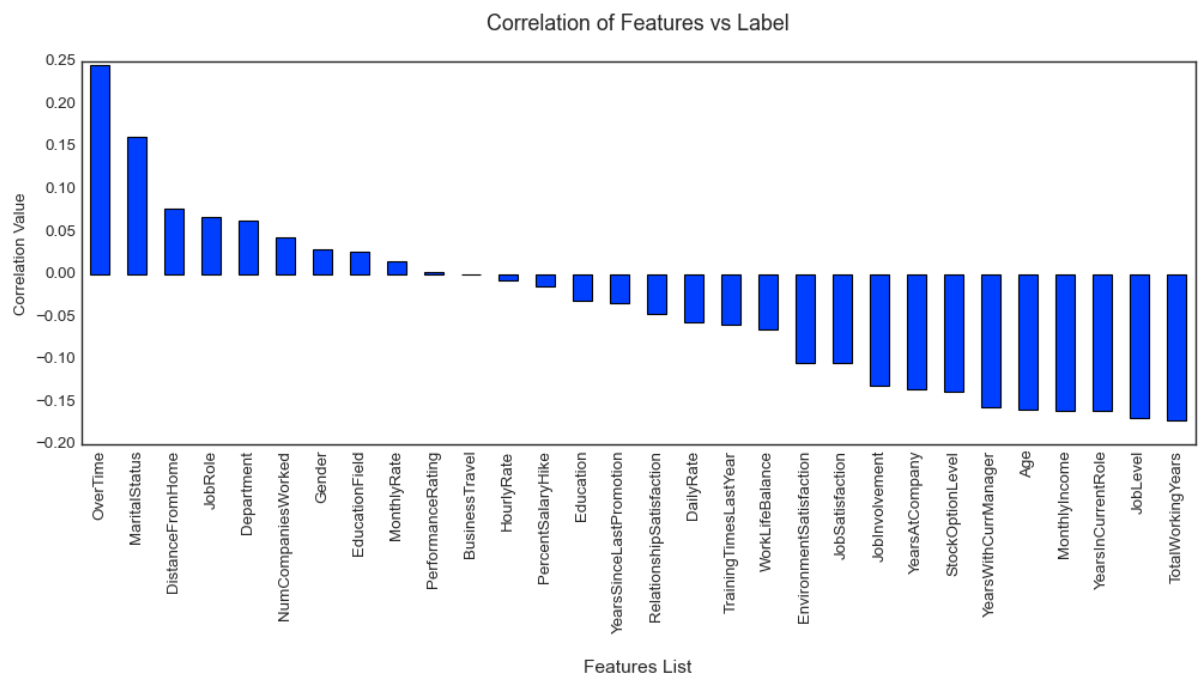
Code:

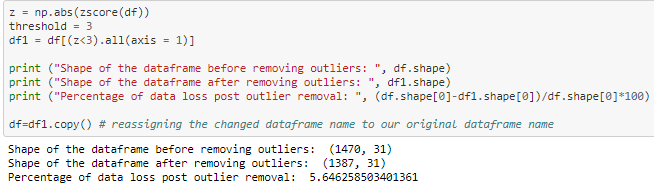


Output:  
  
  
Because of the large number of columns, it seems small even on the Jupyter Notebook. So, in these situations, I pay close attention to the colours to see whether there is any multicollinearity among the feature columns and if there is still any column that I can remove. However, to see the relationship between our label and feature columns clearly, I use a Bar Plot comparison, which you can download the code for here.

Code:



Output:  
  
  
The feature columns that are favourably associated with our label and the feature columns that are negatively connected with our label are clearly defined in the above Bar Plot. Returning to our dataset's outlier and skewness concerns, I'll use the Z score and Log transformation approaches.  
  
Code:

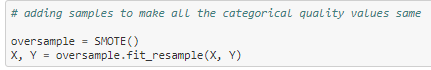
  
  
When I utilised the Z score approach, I was able to lose just approximately 5% of the data, but when I used the IQR method, I believe I lost 30% of the data. As a Data Scientist, it is always more important to save useful data and correct it rather than just delete it unless it is the final choice. After that, I use the Log transformation to cope with the skewness because the allowable range for each column is within +/-0.5.

Code:



After dealing with the data issues, I'll divide our columns into feature and label categories. The feature columns are stored in the X variable, whereas the target label column is stored in the Y variable.  
  
Code:

  
  
However, there was a disparity in the label classes. The figure given in the count plot previously showed a significant difference between the "Yes" and "No" values. As a result, I'll have to fix it because the imbalance might cause our machine learning model to be biased toward the "No" answer.  
  
Code:

  
  
Then, to eliminate any form of bias over column values, I'll scale the feature columns that are contained in the X variable. Because some integers cover thousands of places while others cover hundreds or tens, the machine learning model may believe the column with thousands of places has a larger relevance than it actually does owing to the unit range disparity.  
  
Code:

  
  
I'd like to provide a small piece of code that lets us select a suitable random state for machine learning models.

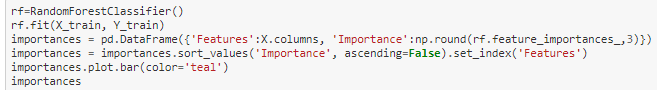
Code:



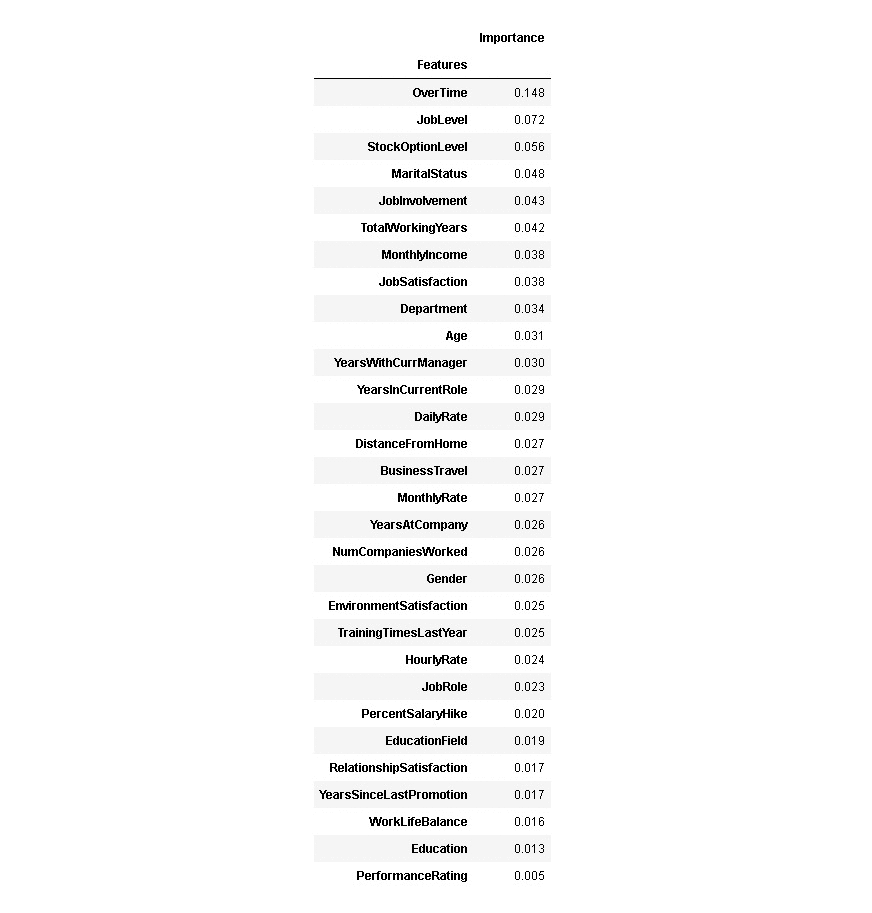
Then I'll divide our complete data set into training and testing data using the train test split. In this case, I'm utilising 75% of the data for training and 25% for testing. Some individuals also offer training and test data separately, so it's entirely up to you how you want to use this stage.  
  
Code:

  
  
Now, before I begin developing my machine learning model, I examine the significance of my feature columns. This enables me to see how the feature columns are used and how much weight they have in predicting my target label.

Code:

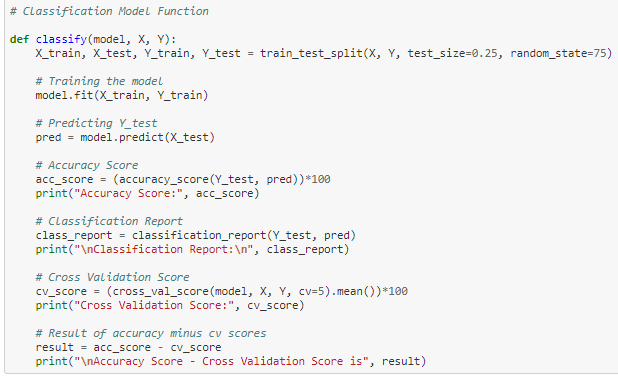
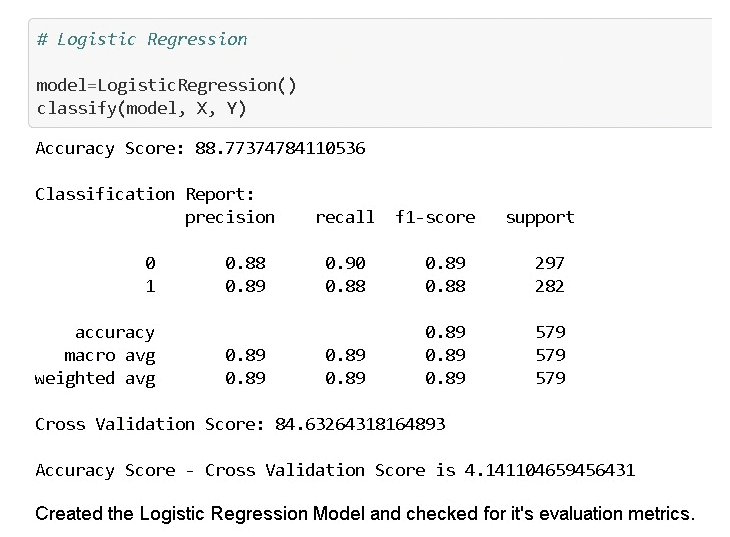


Output:

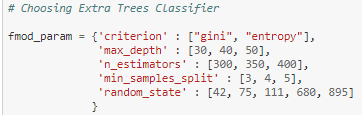
  
  
After we've spent enough time conducting EDA and pre-processing our data, we'll go on to the step that culminated all of our prior efforts. That's when we'll finally start working on our Machine Learning model for categorization.

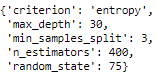
**5. Building Machine Learning Models**  
  
I loaded the relevant libraries and constructed a function that covers all of our machine learning model development and evaluation metrics phases in order to build a classification technique. This makes our work simpler later on since we only have to supply the model's name and obtain the answer instead of repeating/rewriting the same code.

Code:

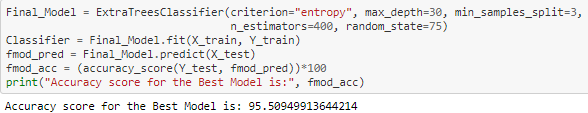
  
  
Output:  
  
  
It's always a good idea to create more than 5 machine learning models so you can pick the one that performs the best and then apply hyper parameter tweaking to improve it even more. I'm going to choose the Extra Trees Classifier as my classification model of choice because it appears to outperform the other models I tried.

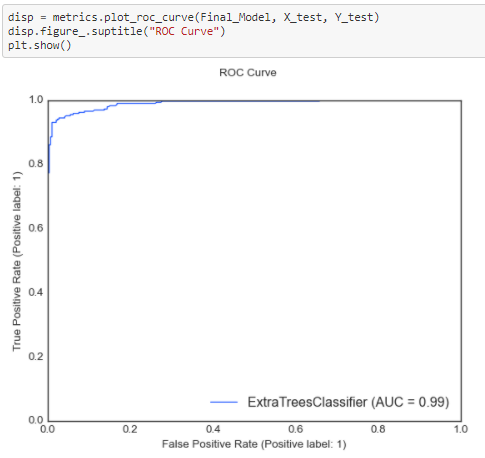
Code:

  
  
  
  
  
Output:

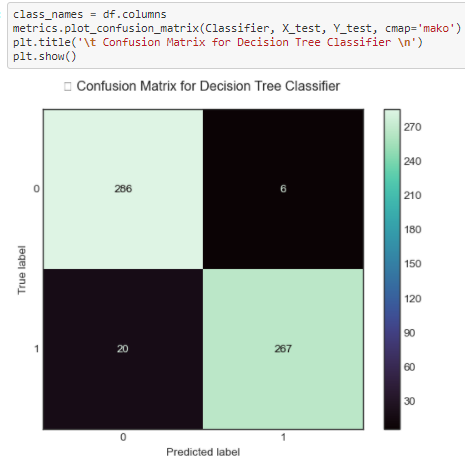
  
  
After following the procedures above to get the optimal parameter list, all I have to do now is input it into my final model and wait for the results. For the final model, I built a ROC curve display and a Confusion matrix.

Code:

  
  
Code:

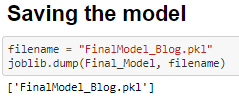


Code:

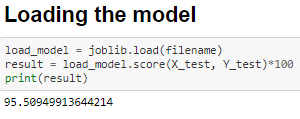
  
  
I neglected to change the print statement for my confusion matrix, so it still shows the Decision Tree model name instead of the Extra Trees model name, and the data for the former are still kept. This also demonstrates that your best-performing model can change over time even if you don't change your code and just run it numerous times. But I'm sure you can see that the print statement may be adjusted to suit your needs, and the main thing was to use the code to produce the desired effect.

After you've completed all of the preceding processes and are happy with the results, you may save the finished model using either joblib or pickle. I saved and then loaded my model from the same saved filename using the joblib technique.

Code:



Code:

  
  
  
**6. Concluding Remarks**  
  
Please allow me to offer a short overview of all of the stages we took, beginning with understanding the Problem Definition and continuing through the Data Analysis and EDA procedures. Before moving on to the last phase of Building Machine Learning Models, we had to go through the essential Pre-processing Data procedures.

What I do is code my whole project on my own, then search for inspiration on the internet by looking at other people's coding styles to see if there's anything I can add to improve accuracy or improve the graphics. However, I've seen a lot of people do the exact opposite: they don't practise or create their own unique coding style first, instead copying and pasting lines from the web and performing some sort of haphazard patchwork, and when asked to explain, they might not be able to convey the functionality or usage of those code blocks.

Before I conclude this, my sole piece of advice to everyone is "No pain, no gain," which means you'll have to get your hands filthy by writing your own code and experimenting with all the different combinations. Create your own commandment list for creating data stories and follow it along with the normal project life cycle. I hope that this lengthy essay has provided you with the necessary information to begin developing your first project from the ground up.

**Disclaimer:** I'm a rookie in this profession with some gathered knowledge from a year of studying Data Science, and I figured that because sharing is caring, someone fresh to the field may benefit from my expertise. I'm also open to receiving input from anyone who can help me improve! The stuff I've written is simply my opinion of the project, however it was undoubtedly influenced by others who have worked on similar projects before me on the internet.