**HR Analytics Project**



**The Article contains following subtitles:**

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

**1.      Problem Definition**

The project I'm utilising for this essay is for HR Analytics, and it's a fake dataset generated by IBM's Data Scientists. Every year, several companies hire a large number of people. Companies devote time and money in educating their staff, and there are also training programmes within the companies for current personnel. These programmes strive to improve the effectiveness of their personnel. But where does HR analytics come into this? Is it just about enhancing employee performance?

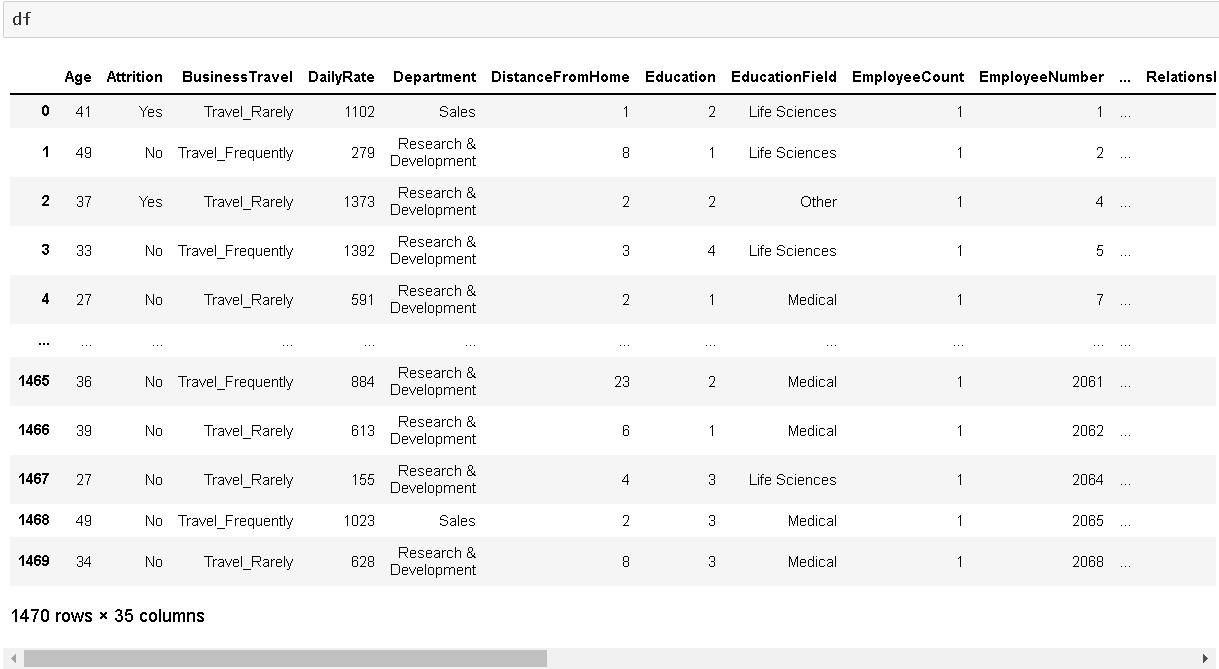
Human Resource Analytics is a subfield of analytics that refers to applying analytic techniques to an organization's human resource department in the hopes of enhancing employee performance and so achieving a higher return on investment. Attrition in human resources is the progressive loss of employees over time. Any organisation that experiences relatively significant attrition faces challenges. HR specialists frequently take the lead in developing firm remuneration programmes, work cultures, and motivating systems to help the organisation retain top talent. How does attrition effect businesses, and how can HR Analytics help with attrition analysis? We'll talk about the first question here, and for the second, we'll write the code and try to understand it step by step.  
  
Attrition is a key issue for businesses because high employee attrition is costly to the organisation. Job listings, hiring processes, documentation, and new hire trainings are some of the most frequent expenses associated with employee turnover and replacement. Furthermore, frequent personnel turnover prevents an organisation from expanding its collective knowledge and experience over time. This is especially concerning if your company is customer-facing, as customers want to contact with recognised faces. If you are frequently hiring new employees, you are more likely to make mistakes and have problems.

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.



First, we'll import all of the dependencies that will be used in our project, then obtain the remainder as needed. Before we start any procedure, we need to load the dataset into our Jupyter Notebook, which can be done in a single step.



This gives us our entire dataset stored in the variable name ‘data’ for our data frame.  
  
**2. Data Analysis**  
  
When it comes to the data analysis portion, we can just skim the contents of our dataset, attempting to make sense of certain columns, their associated values, and any other ideas that spring to mind.  
  
  
  
We can observe from the code above that there are a total of 1470 rows and 35 columns in our dataset. The dataset has a comparatively larger number of rows and columns, hence the visualisation is shortened.  
  
**3. EDA**

For many data scientists, including myself, exploratory data analysis, or EDA, is the most crucial component of data science. I've followed a tonne of knowledgeable data scientists on different platforms, and I can attest to the fact that everything comes down to telling a story through your code about how you were able to accomplish each step, including the problem statement that was provided, the observation, the difficulties encountered, and the steps taken to address or resolve those problems.  
  
Only until you have a clear understanding of what you are doing and why, can you build a good model. ensuring that the clean data you have in the right format after processing it will feed your model and produce the desired results. Since the only input you have available is your data, sorting and fixing it will not help no matter how many machine learning models you employ or how much you tweak the hyperparameters.

A well-known proverb among data scientists and other data workers goes something like this: "Garbage in, garbage out." This basically means that before you even begin using your data more extensively, you should clean it up and work on it thoroughly if you are trying to build an automated model or make sense of it. The act of gathering clean data is something I cannot emphasise enough, but in reality, clean data does not exist. This is when data analysis enters the picture, and employing a variety of techniques—such as data pre-processing, data cleansing, and data engineering—brings us closer to achieving the intended label prediction.

Talk is cheap, but sometimes you need to know the truth to understand what is actually happening, so enough with the narrative telling, let me show you the code. Using the instructions below, I will start by looking at the missing data information in our dataset.

These two codes give us the missing values information in a tabular and visual format that looks something like this.

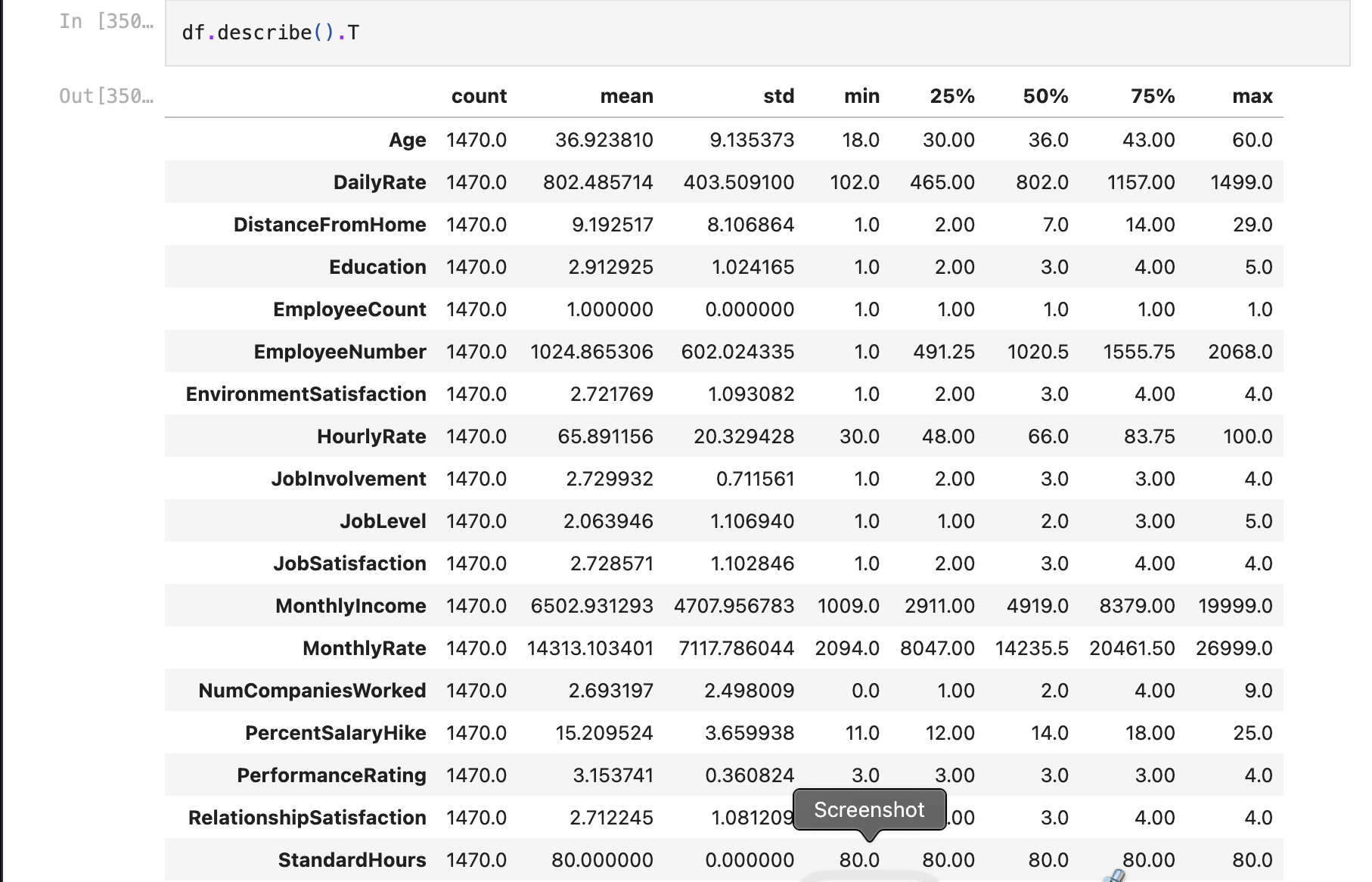
A screenshot of a computer

Description automatically generated

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.  
  
A white rectangular object with black text

Description automatically generated

I was attempting to eliminate every duplicate piece of data in our dataset by using this option. On the other hand, our dataset appears to be free of duplicate data. After that, we examine the count value, mean data, standard deviation details, and the minimum, maximum, 25%, 50%, and 75% quartile details using the describe method. All object (text) type data is ignored by the describe method since it is most effective with numerical data. You can see how to utilise the code by looking at the example below.



After using the code, all of the columns from our dataset are accommodated in both tabular and visual format with the output being in transposition format. After we are able to extract useful information from the describe method, we can use the code below to examine the datatype information. This will provide us with a list of all the columns and indicate whether they are integer, float, or object datatype based on the values that are present in the columns.

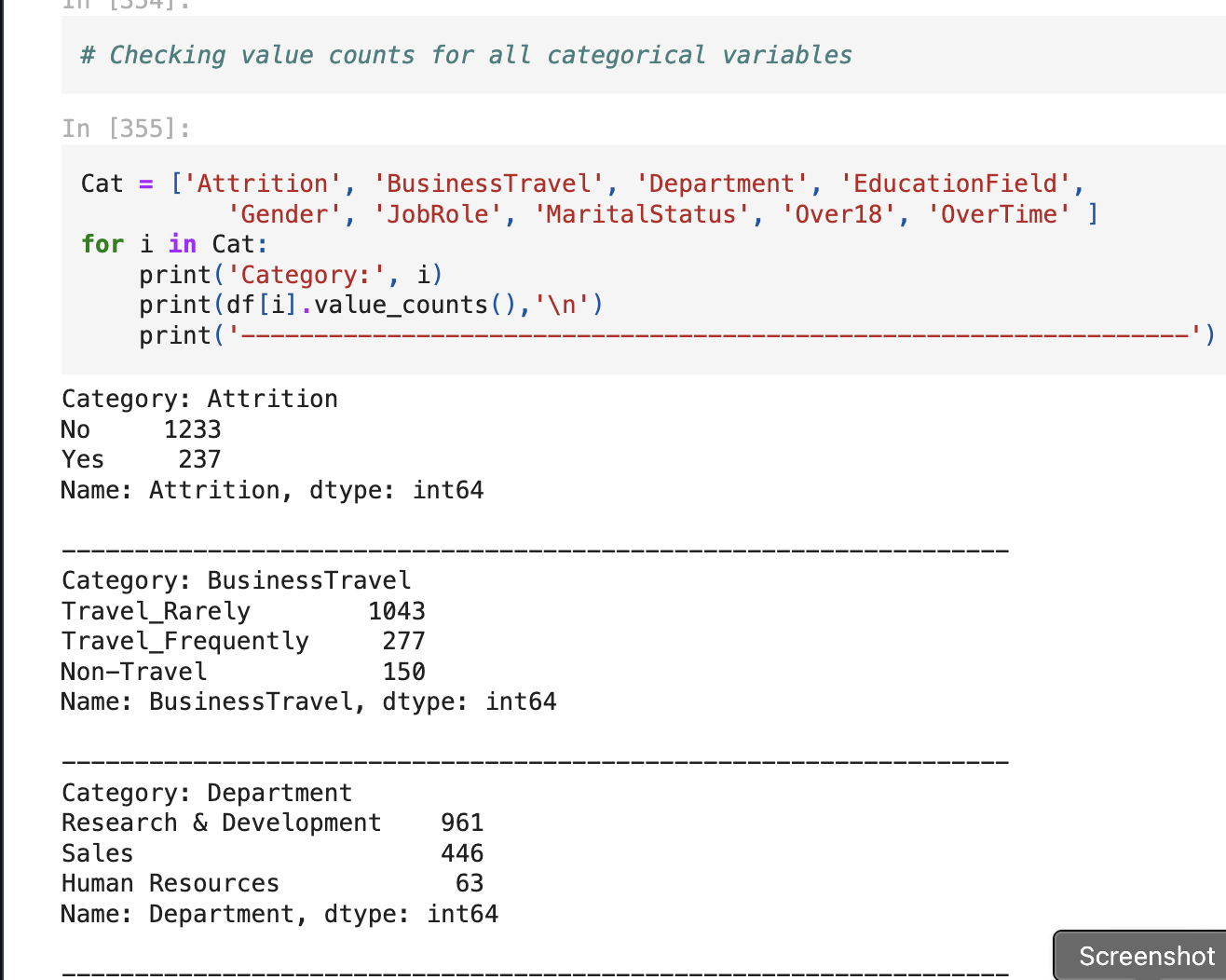
**df.info()**

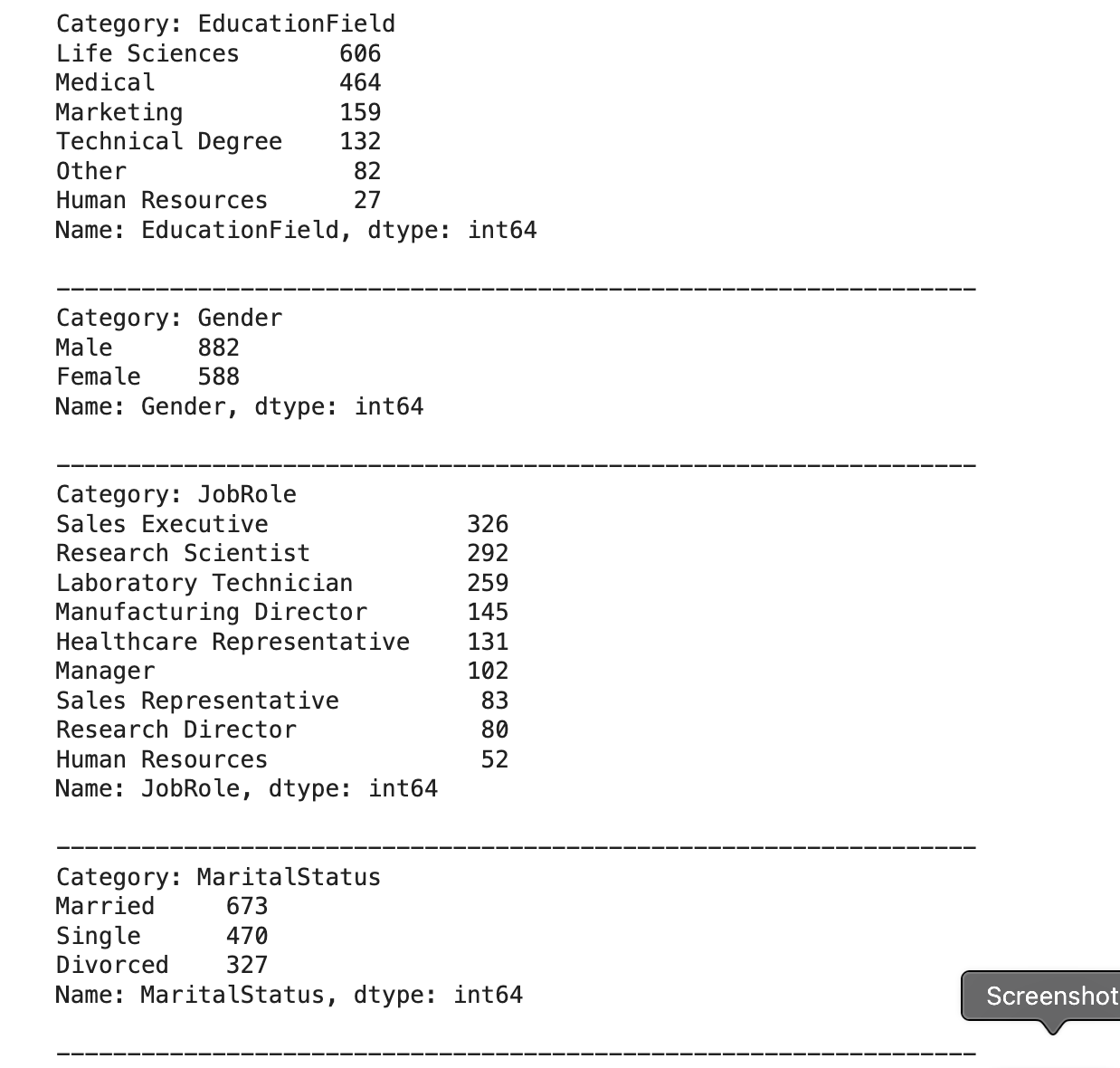
A screenshot of a computer program

Description automatically generated

This is the output that I receive, which describes the datatypes of every column in our data frame. Here, we also have the chance to eliminate any unnecessary columns from the data frame.   
  
In order to facilitate simpler processing in subsequent phases, I like to separate the values of the object datatype and the numeric datatype. To accomplish that, the code simply uses a for loop.

This enables us to save the column names in the object datatype and integer datatype variables in a list manner. We'll look at the total unique values for all the columns after I've divided the datatype column names into two lists, and then we'll use the codes to get the data numbers for just the object datatype columns.





The output that these programmes give us is a list of all the column names that have distinct data covered in the dataset rows that contain numerical data, followed by a description of those values for columns with categorical object datatype. I then visualise the number of rows or count of rows that these values span in our data set, taking into account the separation of object data. By utilising diverse visualisation approaches, I am able to better optimise and examine the columns. It helps me determine where preprocessing of the data will be necessary and where its removal will be advantageous. To be honest, you can only learn all of this by working on various projects; after all, as they say, the more you work, the more you.

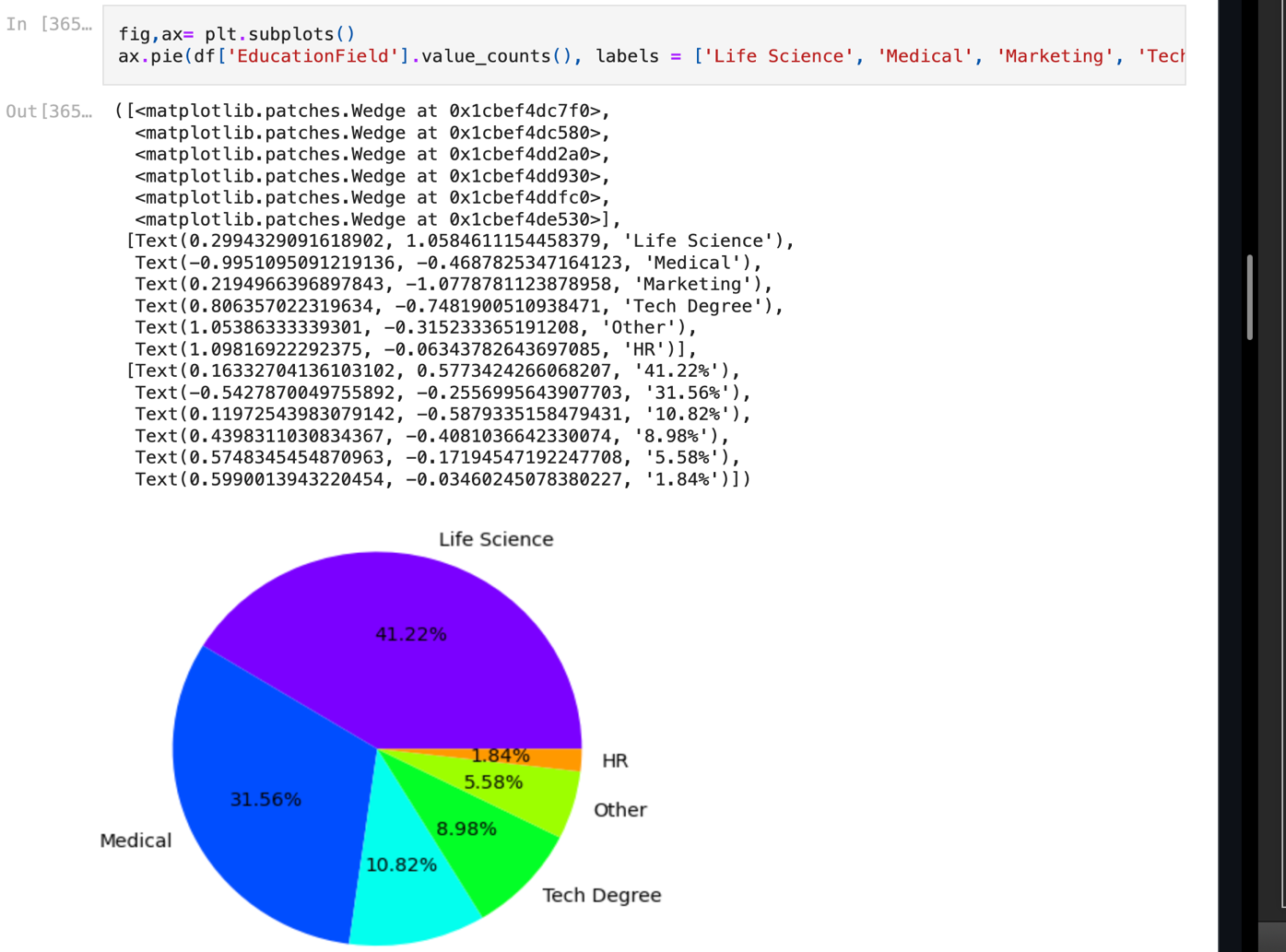
gain expertise in that area by acting as a sixth sense when creating projects of that nature. Although I am providing this example, it does not imply that there are only these stages involved in starting a project. The project's architecture, or framework, won't change, but the methods you use will vary based on the data you are working with.

For instance, I didn't have any missing values for this project, so I didn't worry about handling them. However, there are datasets with a lot of missing data, which are then filled in using a variety of techniques and occasionally even thrown out as a last resort if doing so would only serve to bias our machine learning model in favour of one data value or category.  
  
For your reference, allow me to go ahead and list every visualisation code along with its results.   
  


A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated  
  
  


A graph of different colored bars

Description automatically generated

A screenshot of a computer

Description automatically generated

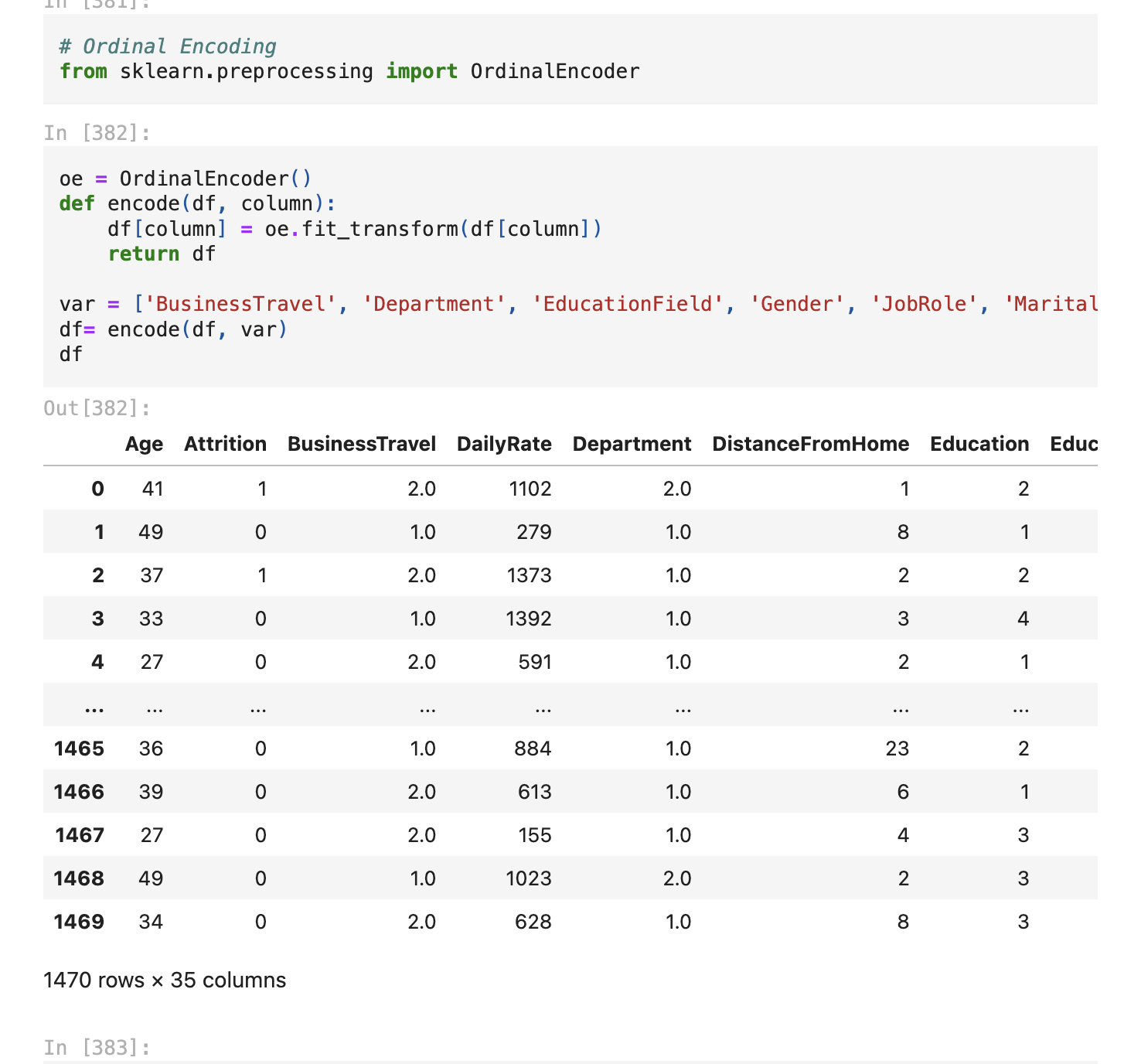
A screenshot of a graph

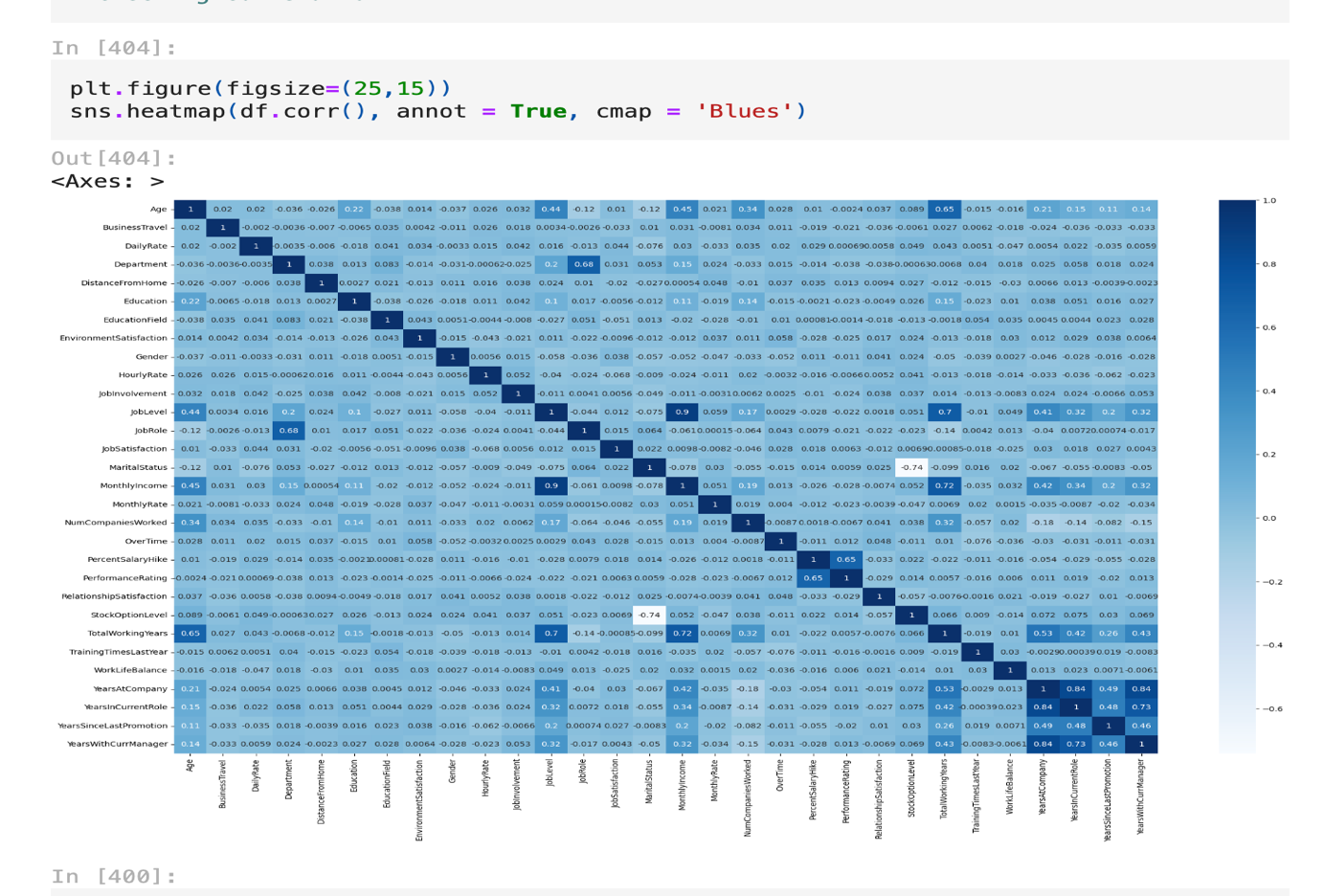
Description automatically generated

A screenshot of a graph

Description automatically generated

You can see that I was able to examine all of the column values and counts with the aid of the aforementioned algorithms and the outputs; the boxen plots provided me with an insight into the existence of outliers, and the distribution plots provided me with the skewness information that will need to be addressed. These are similar to the obstacles that must be overcome before I can begin developing my machine learning models for classification.

**4. Pre-processing Data**  
  
As part of the pre-processing procedure, I will address each miss fit individually, beginning with the issue that our machine learning models can only interpret numeric values, despite the fact that our dataset contains values of object datatype. I'm converting all of the object datatype values using the encoding techniques. I use the Ordinal Encoder for other categorical feature columns and the Label Encoder for our label. I could have utilised The One Hot Encoder in place of the Ordinal Encoder, but as I previously stated, it's all preference and trial and error. The Ordinal Encoder applied to data values that provide an order seems like a better alternative to me than the One Hot Encoder approach, which increases the number of columns. The name Label Encoder itself indicates how much of a specification is needed to comprehend that it is only for our label(s) columns, thus I don't get why so many people are using it on feature columns as well.  
  
  
  
Upon encoding every column in our dataset, I'm examining the data distribution with a histogram. Histograms should be able to recognise every piece of information from our encoded data frame because they only take into account numerical data.  
  
I now feel compelled to use a heatmap to look for specific association information in our dataset. To clear up any doubt on the specifics of the correlation, let me simplify it into two main aspects. Positive correlation: When two variables move in the same direction together, there is a complete positive correlation, indicated by a correlation of +1. Negative correlation: When two variables have a perfect negative correlation of –1, it means that one variable rises, the other falls. The code to view this data is shown below.



Believe me, the large number of columns makes everything appear tiny even in the Jupyter Notebook. In order to determine whether there is a multicollinearity problem among the feature columns and whether there are any more columns that I can eliminate, I mostly look at the colours in these situations. However, I utilise a Bar Plot comparison, the code for which can be found here, to clearly examine the link between our label and feature columns.  
A graph with blue squares

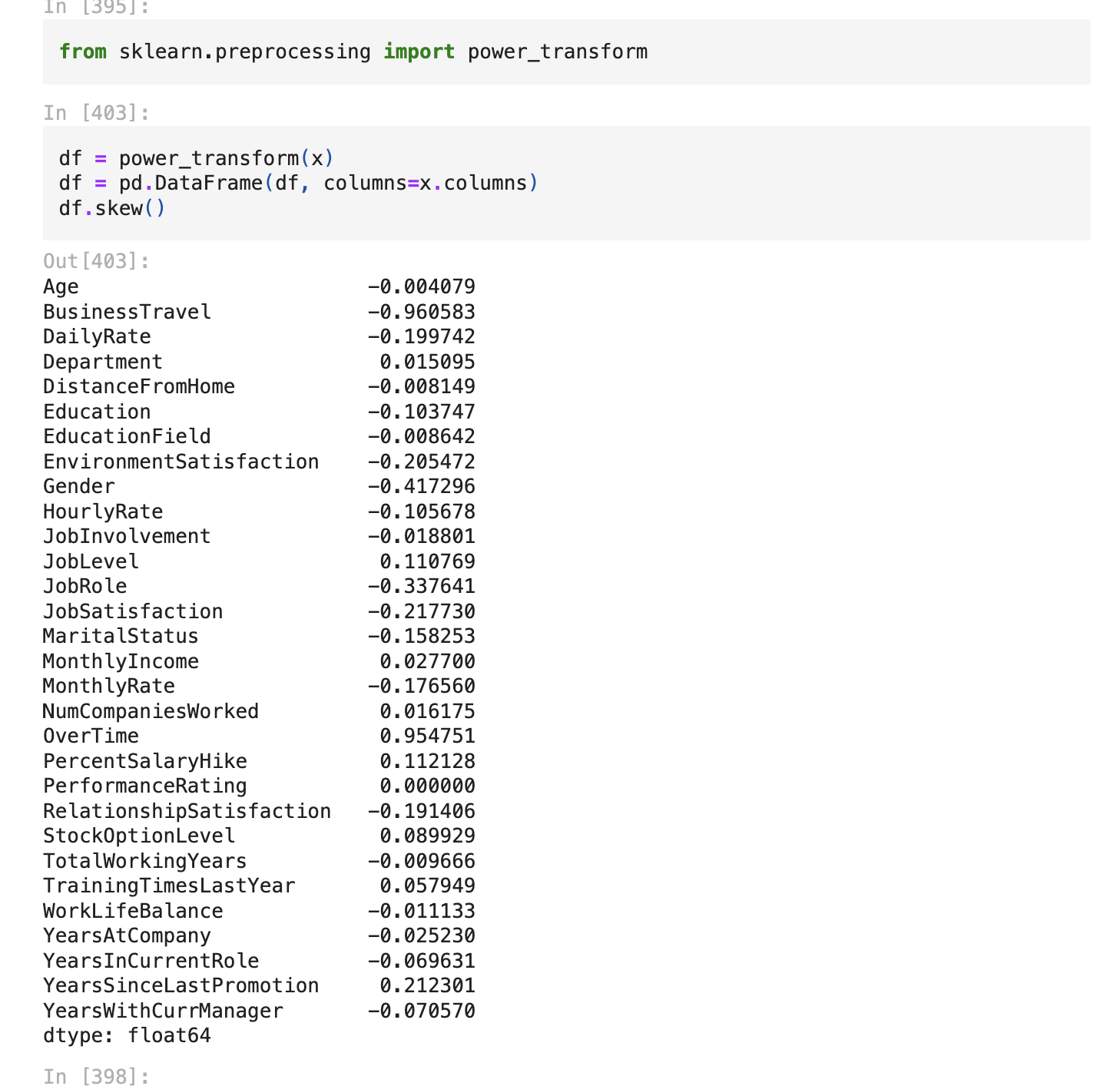
Description automatically generated

We can easily identify the feature columns that have a positive correlation with our label and the feature columns that have a negative correlation with our label in the bar plot above. Returning to our dataset's outlier and skewness concerns, I will be utilising the Z score and Log transformation techniques.

A screenshot of a computer program

Description automatically generated

When I utilised the IQR approach, I think I lost approximately thirty percent of the data, but when I used the Z score, I was able to lose only about five percent. Furthermore, as a data scientist, saving important data always comes first, followed by correcting it rather than eliminating it altogether until absolutely necessary. After that, since the allowable range for each column is between +/-0.5, I am applying the log transformation to address the skewness.



I shall divide our columns into feature and label after addressing the data issues. I have the target label column in the Y variable and the feature columns in the X variable.   
  
However, there was an uneven distribution among the label classes. As you can see from the value shown in the count plot previously, the "Yes" and "No" values were very different from one another. As a result, I will have to fix it because the imbalance can skew our machine learning model in favour of the "No" result.

A screenshot of a computer

Description automatically generated

In order to prevent bias in the column values, I will then scale the feature columns that are kept in the X variable. Due to differences in unit range, the machine learning model may believe that the column with thousands of places has a bigger importance than it actually does. This is because some integers cover thousands of places, while others cover hundreds or tens of places. We can select a suitable random state for the machine learning models using a straightforward piece of code that I would like to share with you.

A screenshot of a computer program

Description automatically generated

After that, I'll divide our whole data set into training and testing subsets using the train test split. Here, I'm utilising 25% of the data for testing and 75% of the data for training. It is entirely up to you how you wish to use this phase as some people also supply training and test data separately.   
  
At this crucial stage, I consider the significance of my feature columns before developing my machine learning model. This helps me understand the role that the feature columns play and how much weight they have in predicting my desired label.

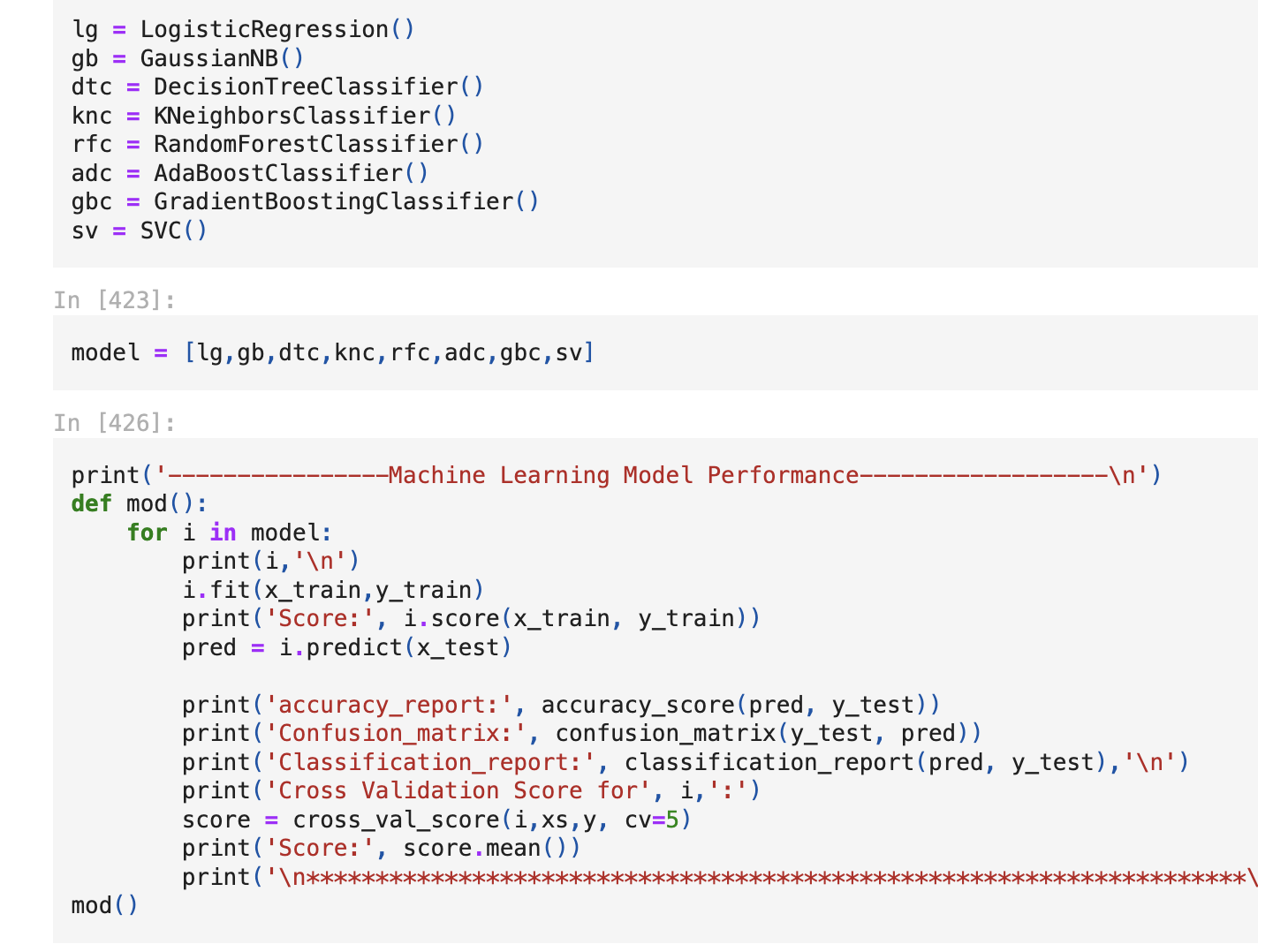
After dedicating sufficient time to EDA and pre-processing our data, the culmination of our earlier laborious efforts is reached. That is, to at last begin developing our machine learning model for categorization.

**5.      Building Machine Learning Models**  
  
I have imported the required libraries and made a function that consists of all the stages involved in creating our machine learning model and determining its assessment metrics in order to construct a classification technique. As a result, we may obtain the result later on by simply feeding the model's name, saving us the trouble of repeatedly creating or repeating the same code.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Building more than five machine learning models is usually advised in order to select the top performing model and then optimise its performance via hyperparameter tuning. Since the Extra Trees Classifier appears to be doing better than the other models I tried, I will be using it as my classification model of choice.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

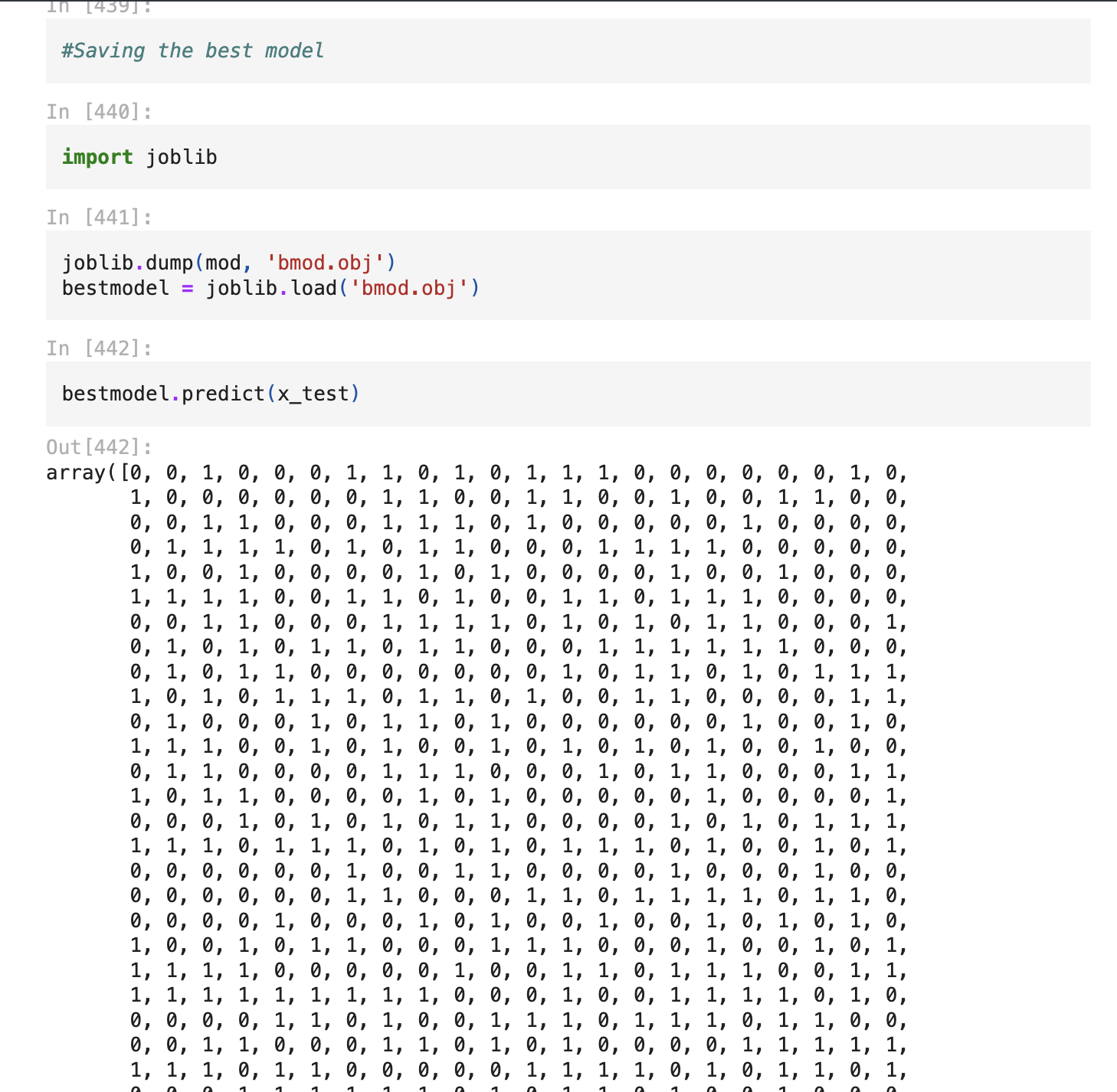
Description automatically generatedA screenshot of a computer

Description automatically generated



I just need to enter the optimal parameters list into my final model after following the preceding methods to obtain it, and I'll get the output. For the finished model, I have made a confusion matrix and ROC curve graphic.  
  
  


The top-performing model may occasionally fluctuate even if you simply run it several times without altering your code. However, I'm sure you can see that the print statement is editable to your preference and that using the code that produces the desired outcome was crucial.   
  
You can use either joblib or pickle to store the finished model after completing all the preceding processes and being happy with the results. I have saved my model using the joblib technique, and I have used the same saved filename to load it.



**6.  Concluding Remarks**  
  
Before moving on to the last phase, which involved building machine learning models, we completed the required Pre-processing Data processes.   
  
I personally code every project from start to finish, then I browse other people's coding styles online for ideas and see what I can use to improve the correctness or aesthetics of the final product. But I have observed a lot of people doing the exact opposite, where they copy and paste lines from the internet and do some kind of haphazard patch work instead of practicing or developing their own distinctive coding style, and when pressed to explain, they may not be able to convey how those code blocks work or are used.

Before I sign out, I would just like to give everyone the advice, "No pain, no gain." You will have to put in the effort to write your own code and experiment with all the possible configurations. Make your own commandment list for conveying a unique data story and adhere to it together with the typical project life cycle. I hope this lengthy post provides you with the foundational knowledge you need to start constructing your first project from scratch.