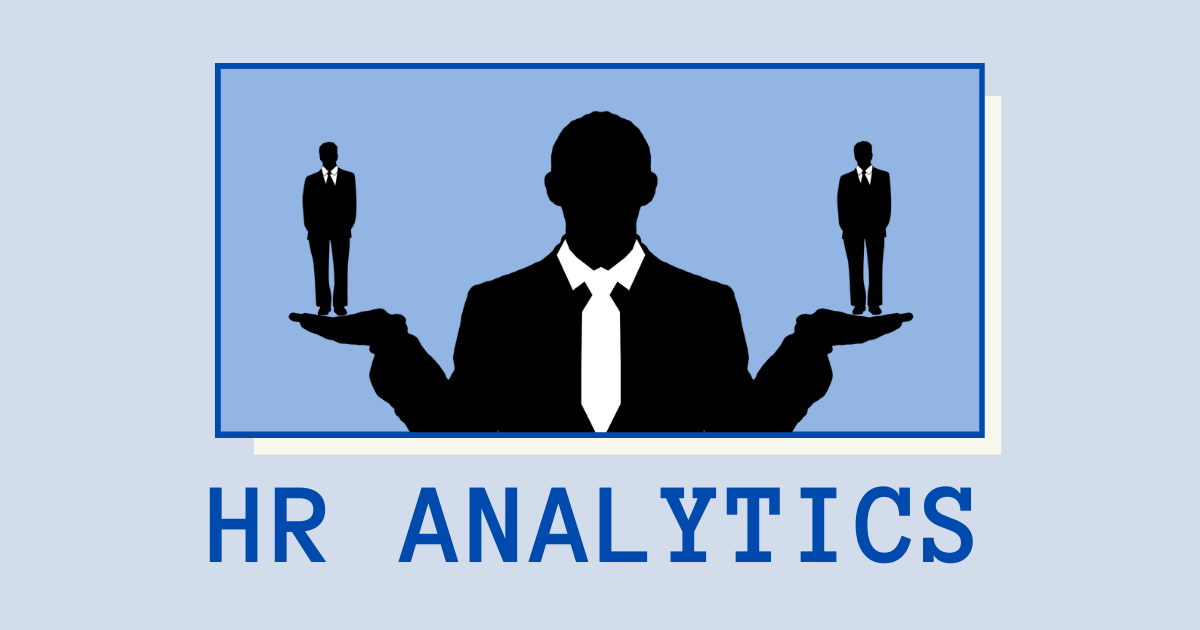
**HR Analytics Project- Understanding the Attrition in HR**

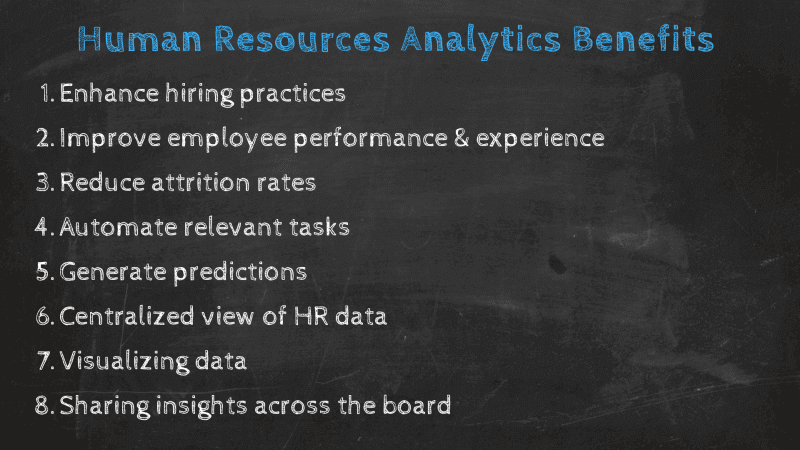
In this blog post, I will go through the whole process of creating a machine learning model on the famous IBM HR Attrition Rate Analytics dataset, which is used by many people all over the world. It provides information on the fate leadership role in designing company compensation programs, work culture, and motivation systems that help the organization retain top employees.

HR ANALYTICS

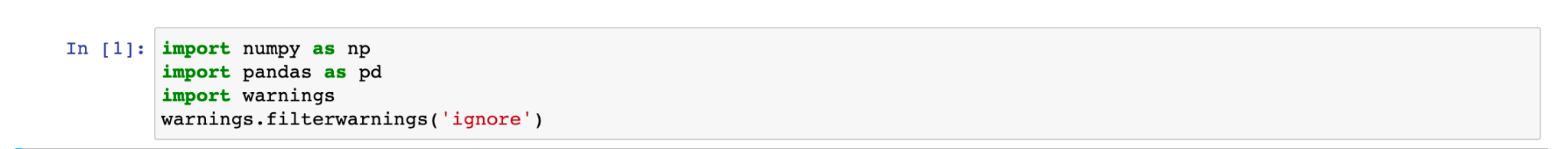
Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analytics does not just deal 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.

Attrition in human resources refers to the gradual loss of employees over time. 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.

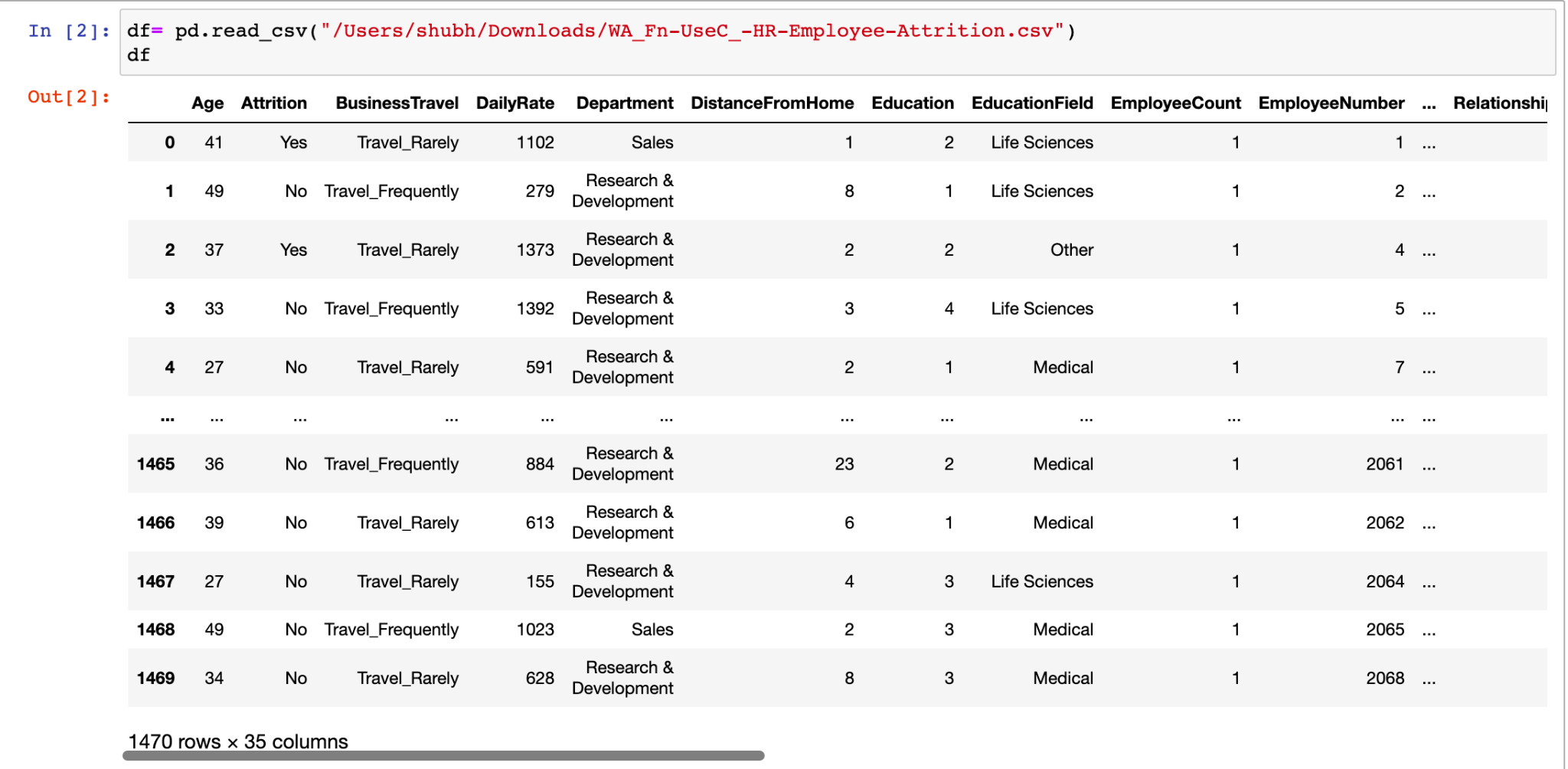
How does Attrition affect companies? and how does HR Analytics help in analyzing attrition? We will discuss the first question here and for the second question, we will write the code and try to understand the process step by step.



# **Importing the Libraries**



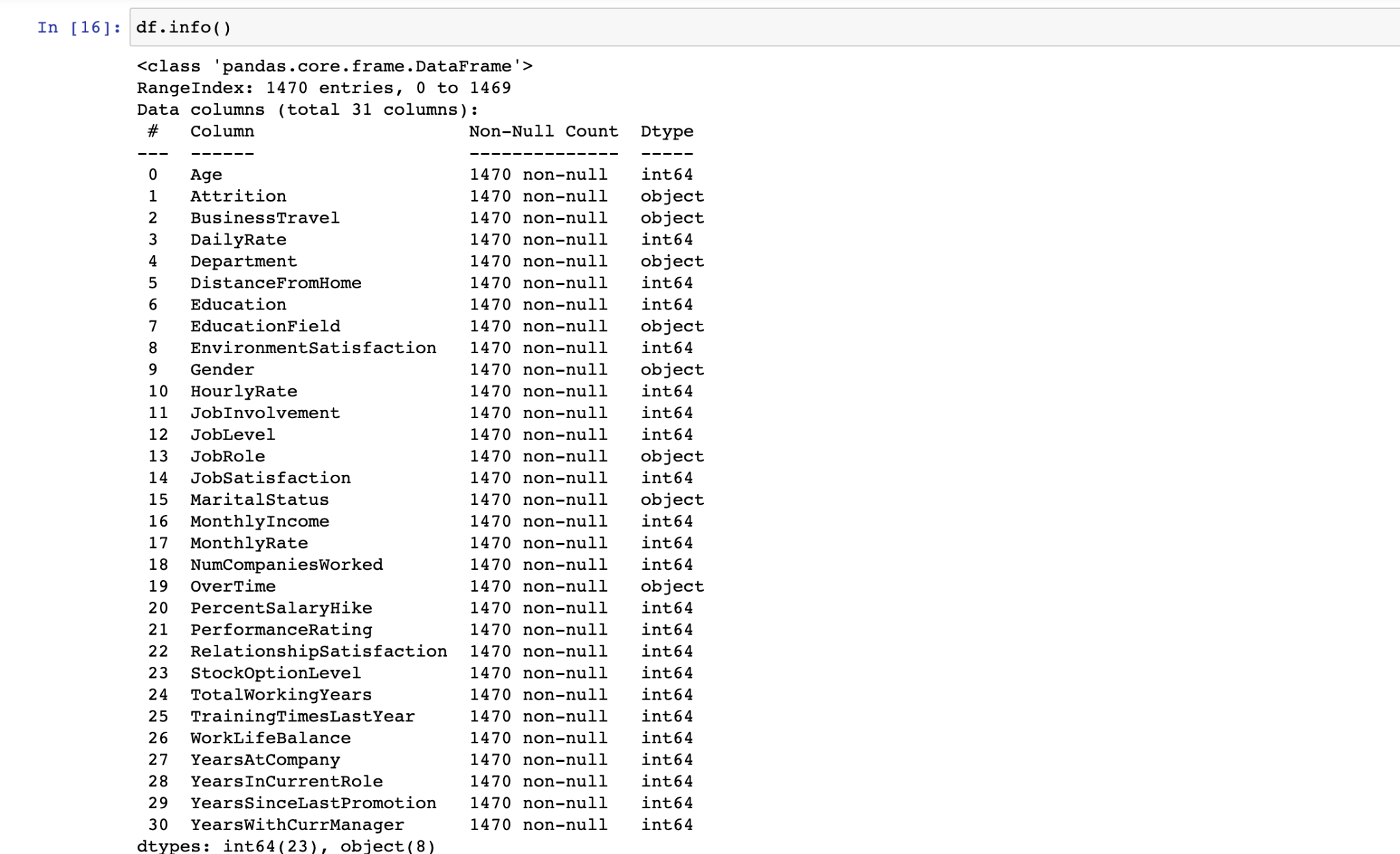
# **Getting the Data**



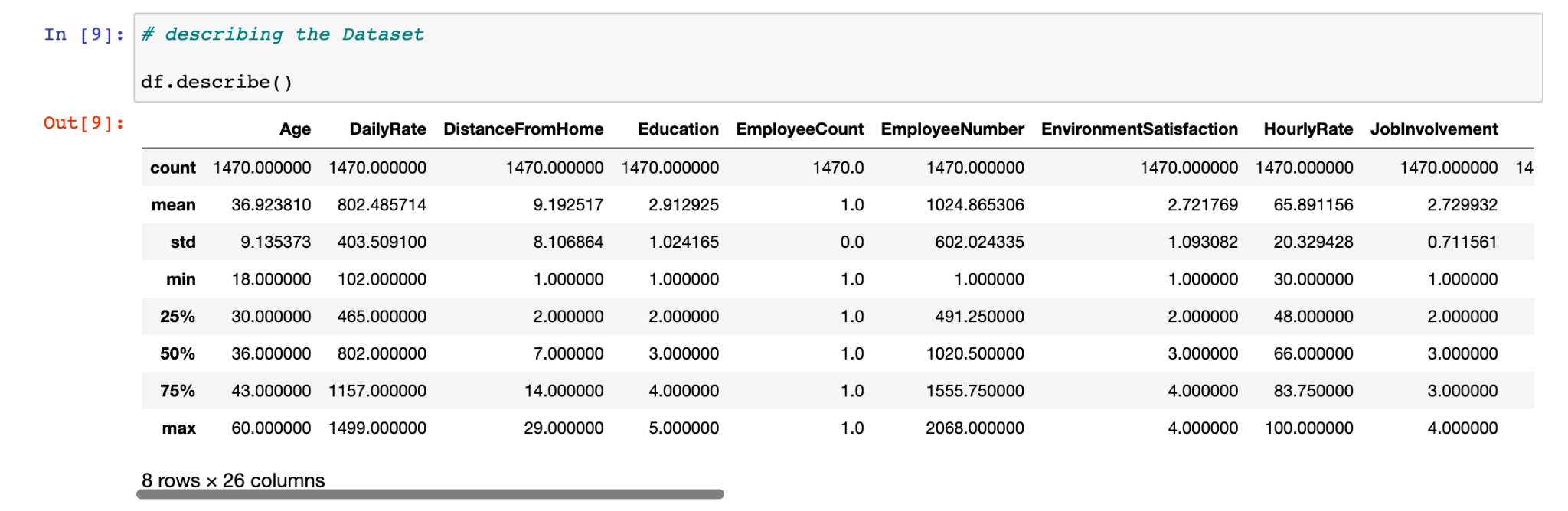
# 

# 

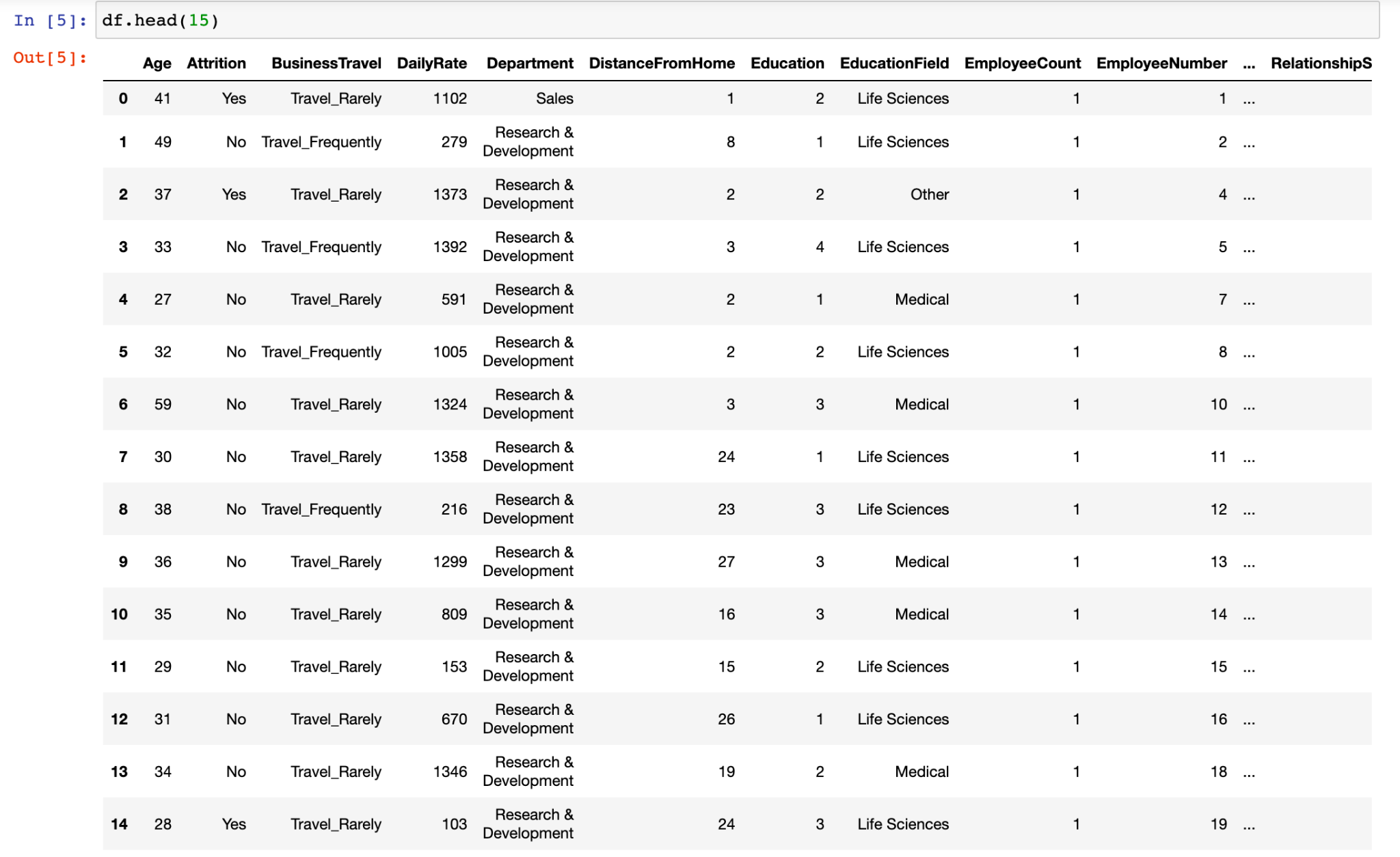
# **Data Exploration/Analysis**



The training set has 1470 examples and 30 features + the target variable (Attrition). 23 features are integers and 8 are objects. Below I have listed the features with a short description:



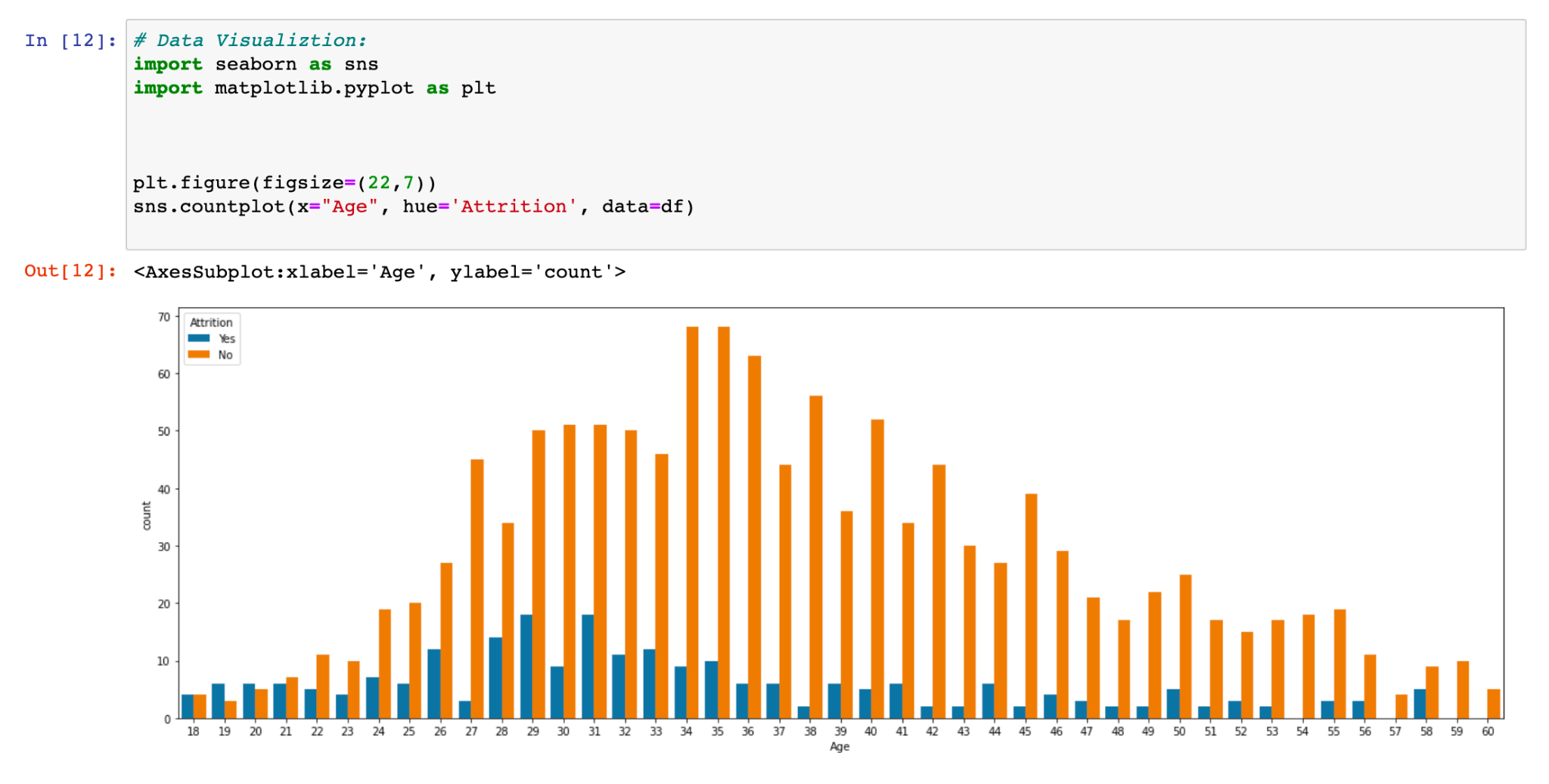
Hence in this data set, there are no null values present. And the data is present in all columns of the dataset.



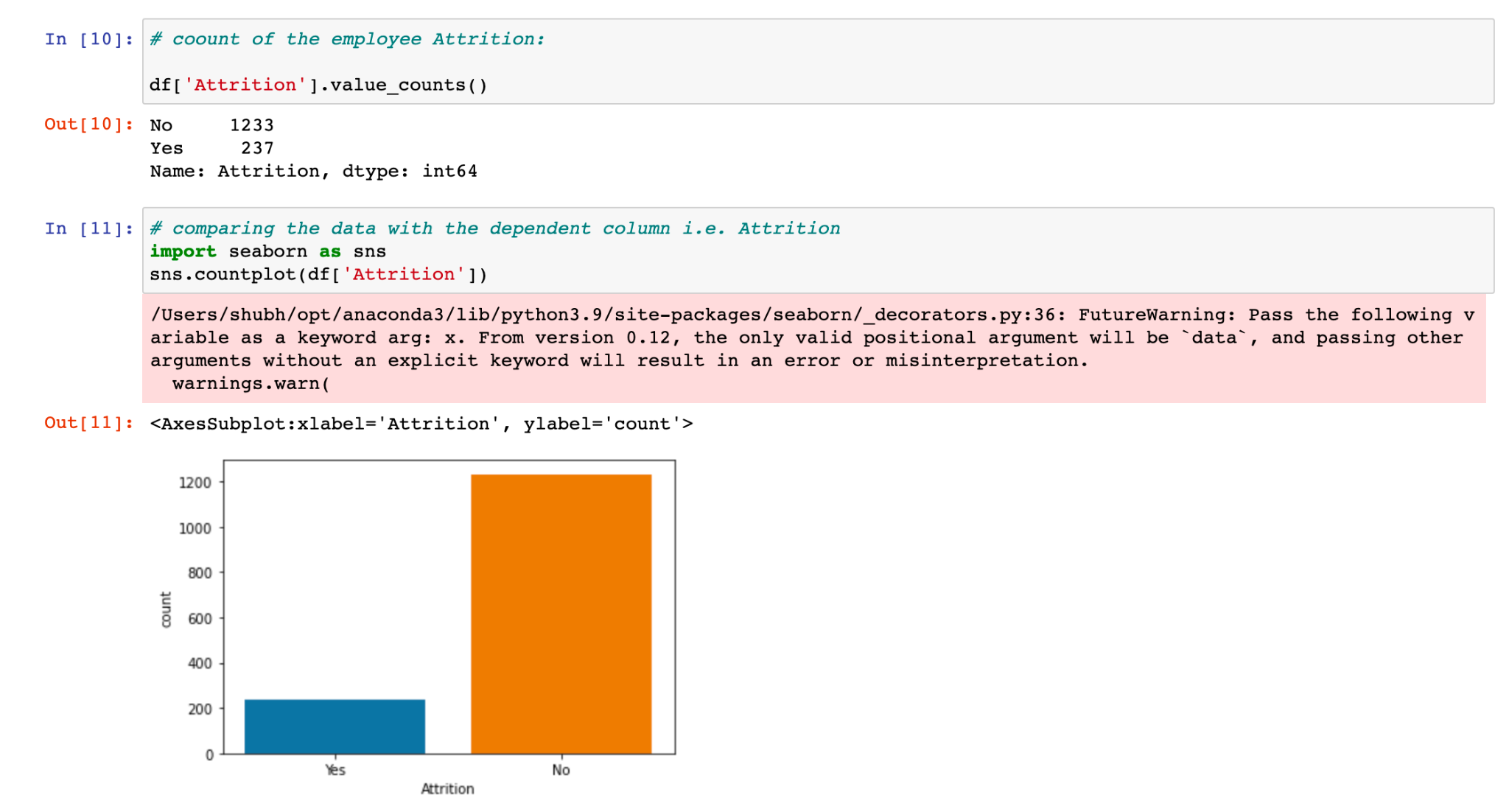
From the table above, we can note a few things. First of all, we need to convert a lot of features into numeric ones later on so that the machine learning algorithms can process them. Furthermore, we can see that the features have widely different ranges, that we will need to convert into roughly the same scale. We can also spot some more features, that we need to deal with.



Above you can see the 30 features + the target variable (survived). What features could contribute to Attrition?

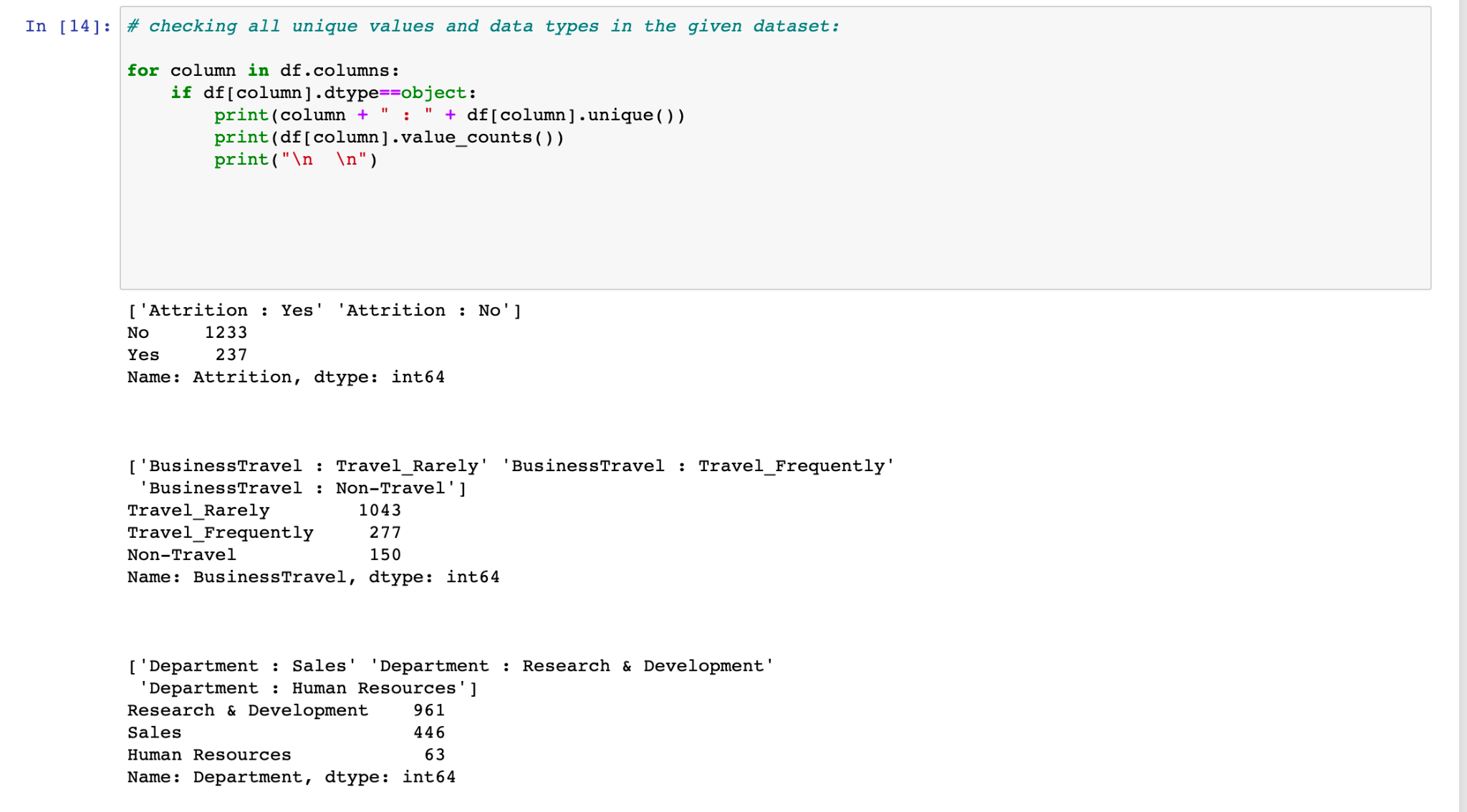


# Here we can see that employees of age 34,35 are the maximum in count that is currently working in the company. While the employee with ages 38,42,43,45,51 is in the maximum count of those who has left the company in the company.



Here we can see that in this count plot of the Attrition column, The employee attrition for the No is around 1233 while for the Yes is about 237. Where here our target is to find the attrition rate of the employees that affect the company.

Now check for all the unique values and data types in the given dataset of HR Analytics.

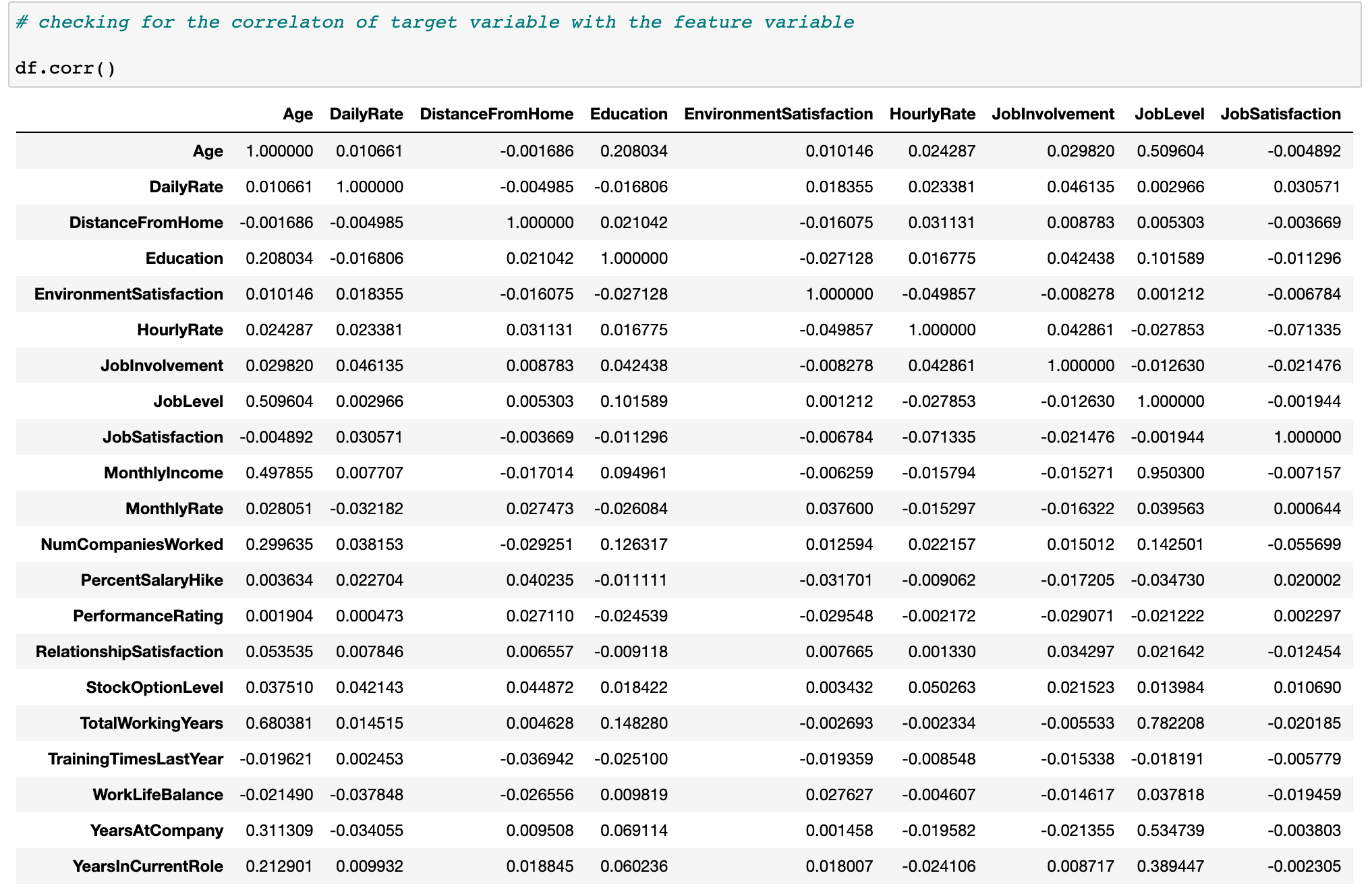


Data Cleaning

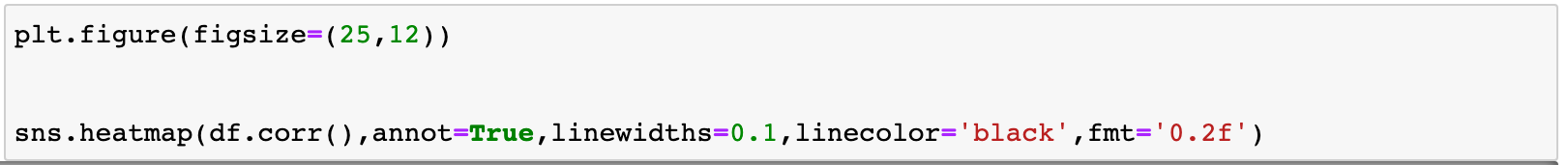


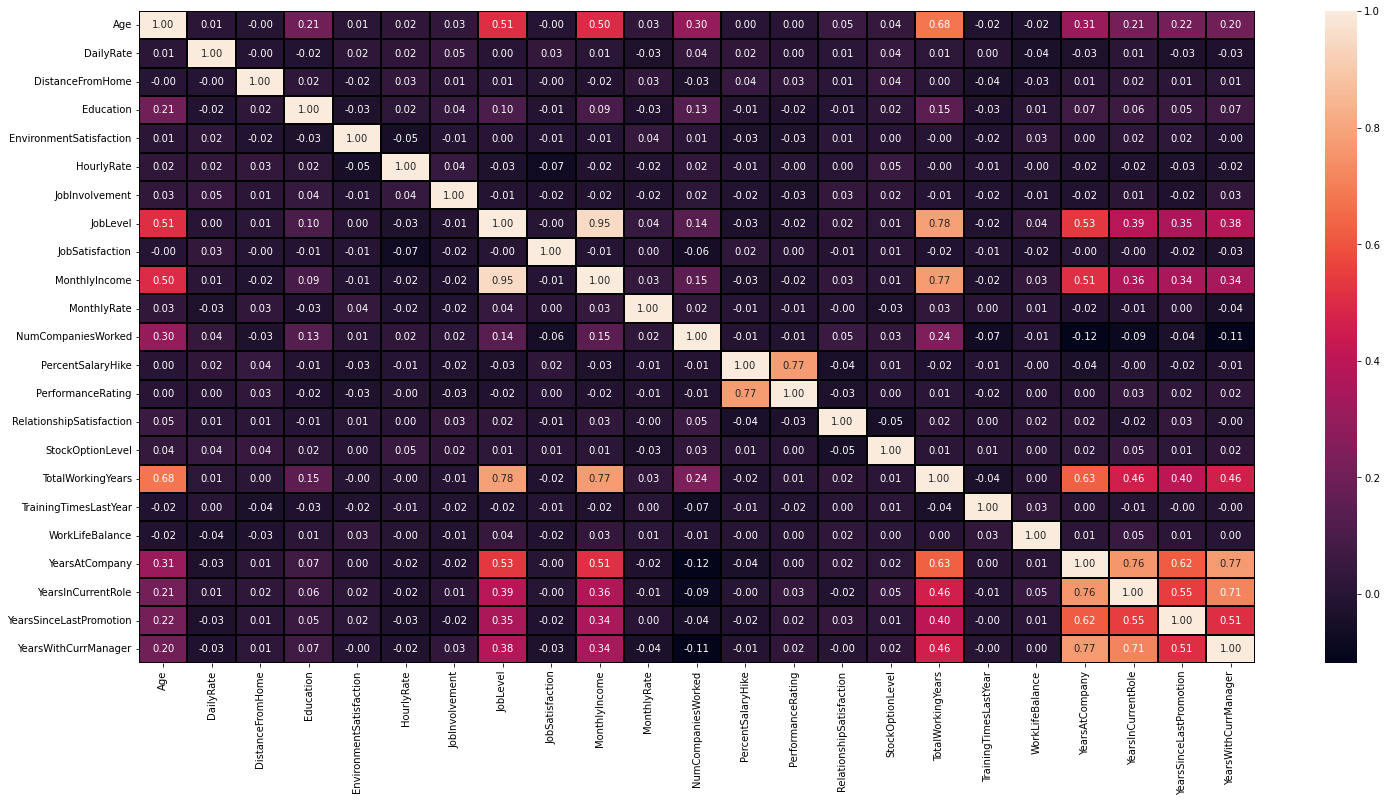
Dropping the unwanted data from the dataset. Here we are dropping the ‘Over18’, ‘StandardHours’, ‘EmployeeNumber’, and ‘EmployeeCount’ these four columns which do not affect our target variable or don’t have much more correlation with the target variable (i.e. Attrition).

Checking for the correlation of the remaining data columns with the target column.



Looking for the heatmap of the correlation.





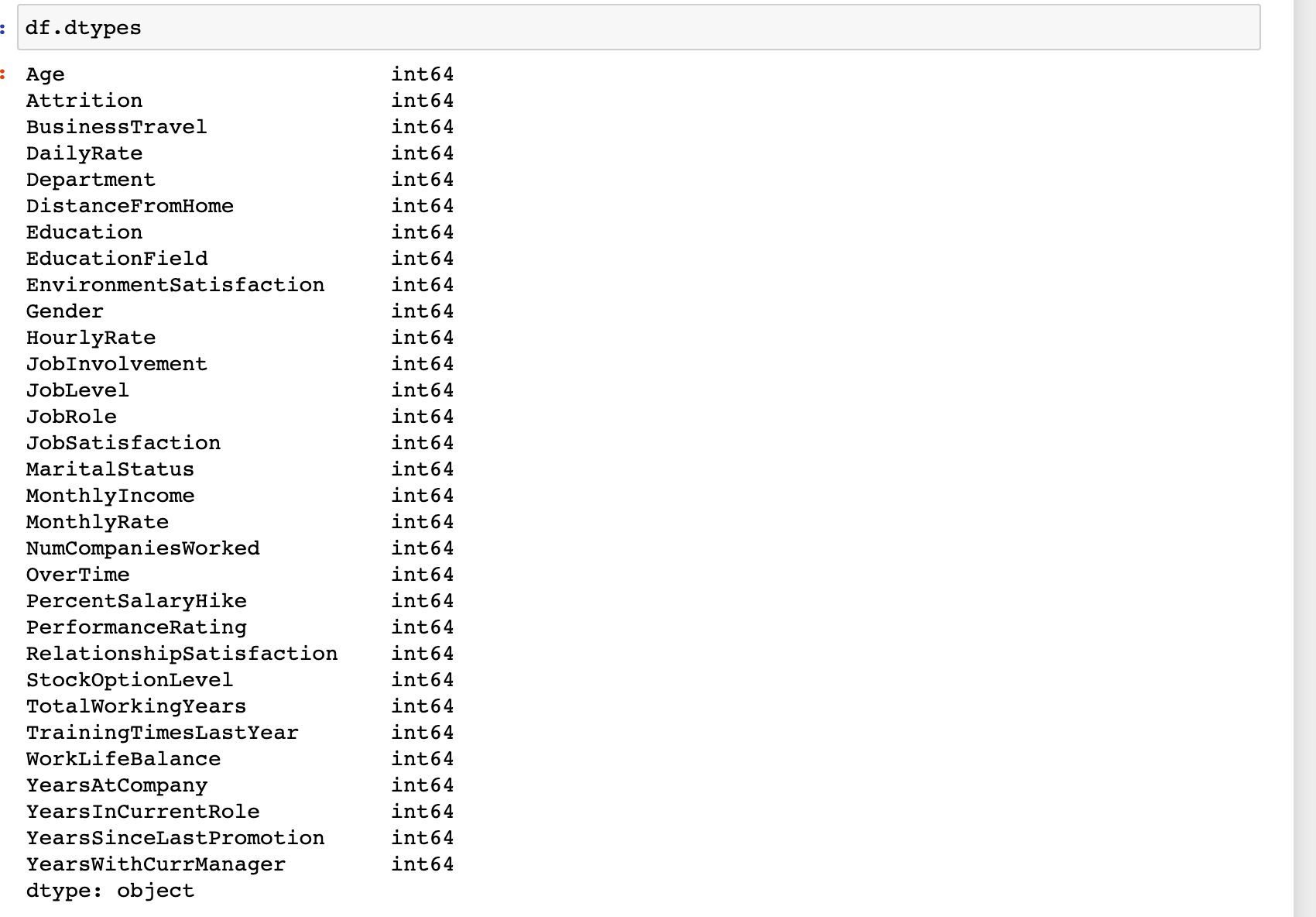
1. Here age is well correlated with the total working years which is up to 68%.
2. Also job level and monthly incoming are goodly correlated with total working years up to 78% and 77% resp.
3. Monthly income is highly correlated with the job level up to 95%.

Data Processing

Label Encoding:

Label Encoding refers to converting the labels into a numeric form to convert them into a machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

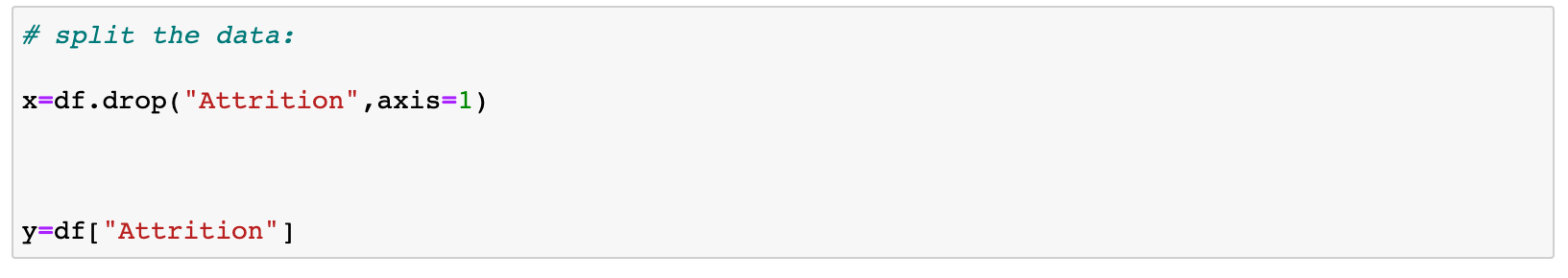




Here we can see that or we can confirm that by importing the label encoder from sklearn we have successfully encoded all column data into the integer.

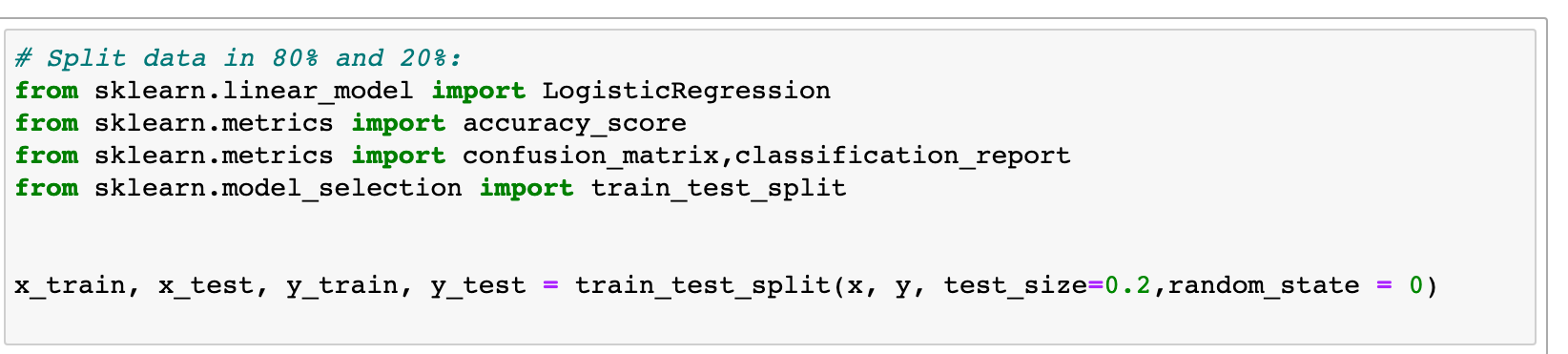
**Building Machine Learning Models**

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross-validation.



After pre-processing, we split our data into training, validation, and test datasets. From a total of 1470 observations, we choose:

1. 80% observation for *Training Dataset.*
2. 20% observation for *Test Dataset.*



Now by splitting the data or separating the data into the feature variables (i.e. x) and target variables (i.e. y ) and our target variable is Attrition. As our Target variable is categorical So using the **Random Forest Classifier.**

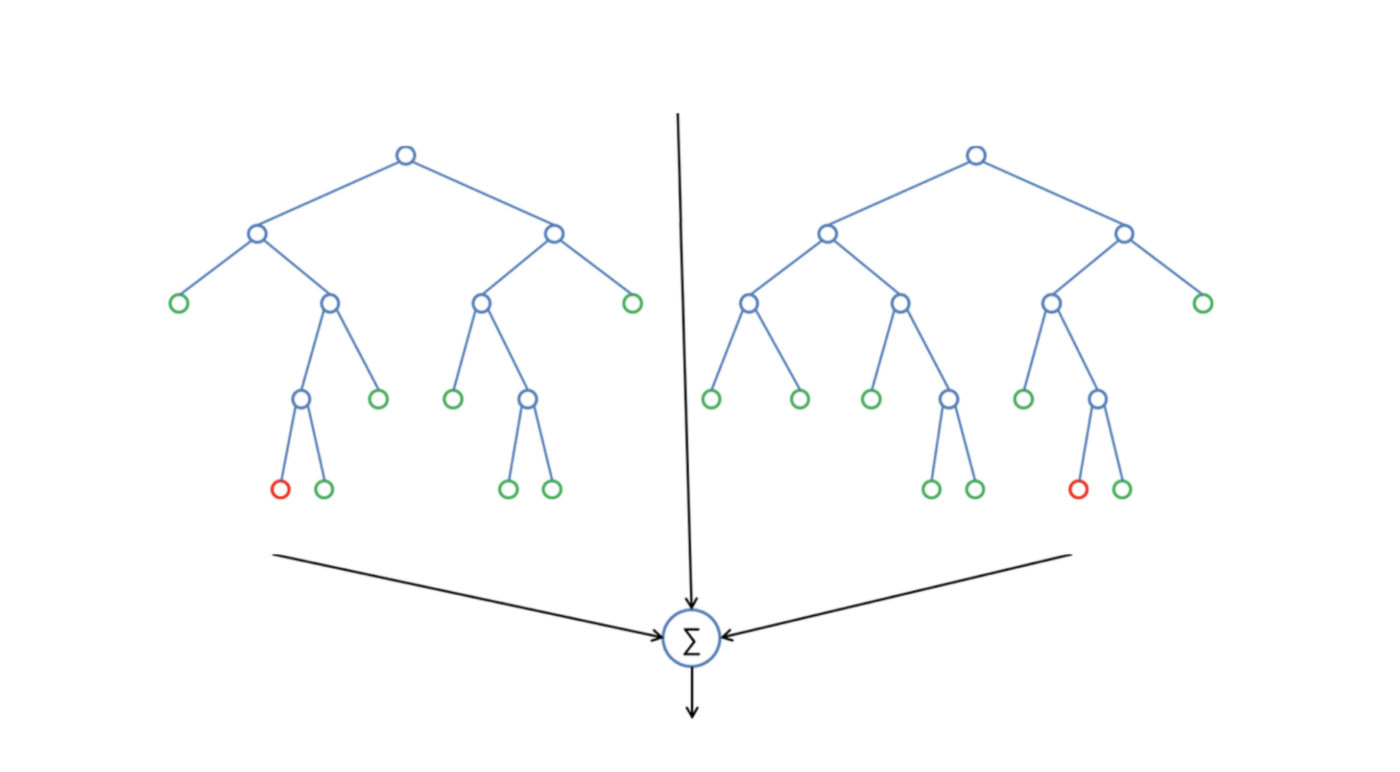
**What is Random Forest ?**

Random Forest is a supervised learning algorithm. As you can already see from its name, it creates a forest and makes it somehow random. The „forest“ it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

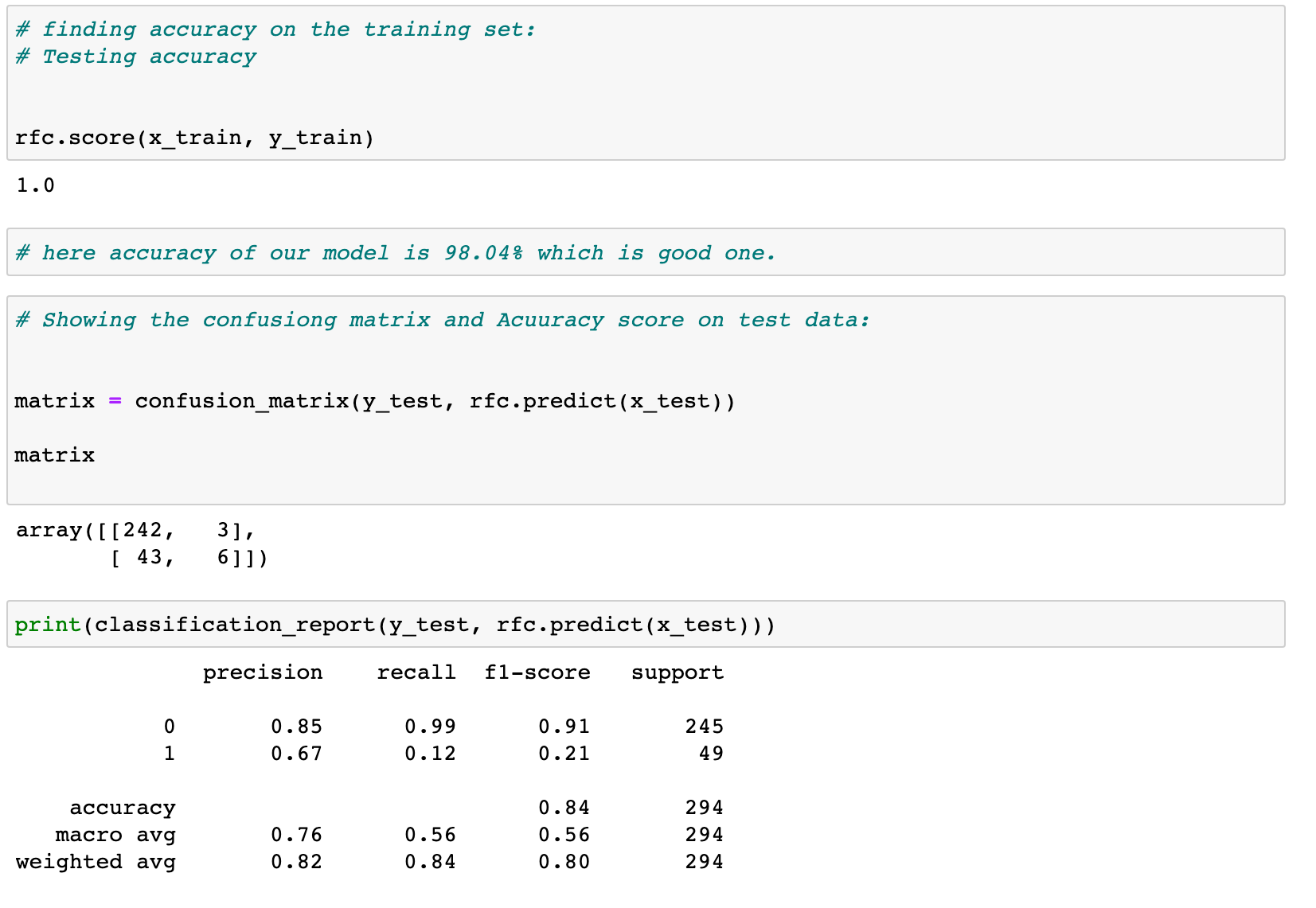
To say it in simple words: Random forest builds multiple decision trees and merges them to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. With a few exceptions, a random-forest classifier has all the hyperparameters of a decision-tree classifier and also all the hyperparameters of a bagging classifier, to control the ensemble itself.

The random-forest algorithm brings extra randomness into the model when it is growing the trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model. Therefore when you are growing a tree in a random forest, only a random subset of the features is considered for splitting a node. You can even make trees more random, by using random thresholds on top of it, for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Below you can see what a random forest would look like with two trees:

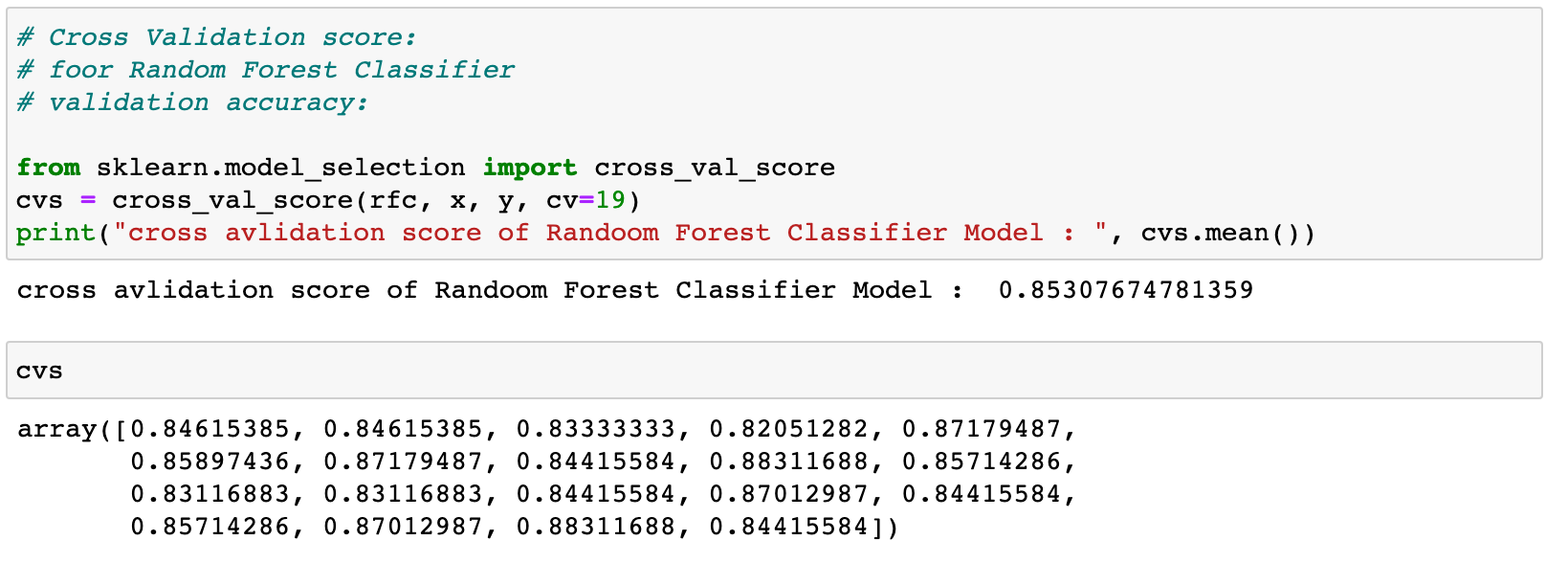




85% Precision Accuracy.

Our random forest model predicts as well as it did before. A general rule is that the more features you have, the more likely your model will suffer from overfitting and vice versa. But I think our data looks fine for now and hasn't too many features.

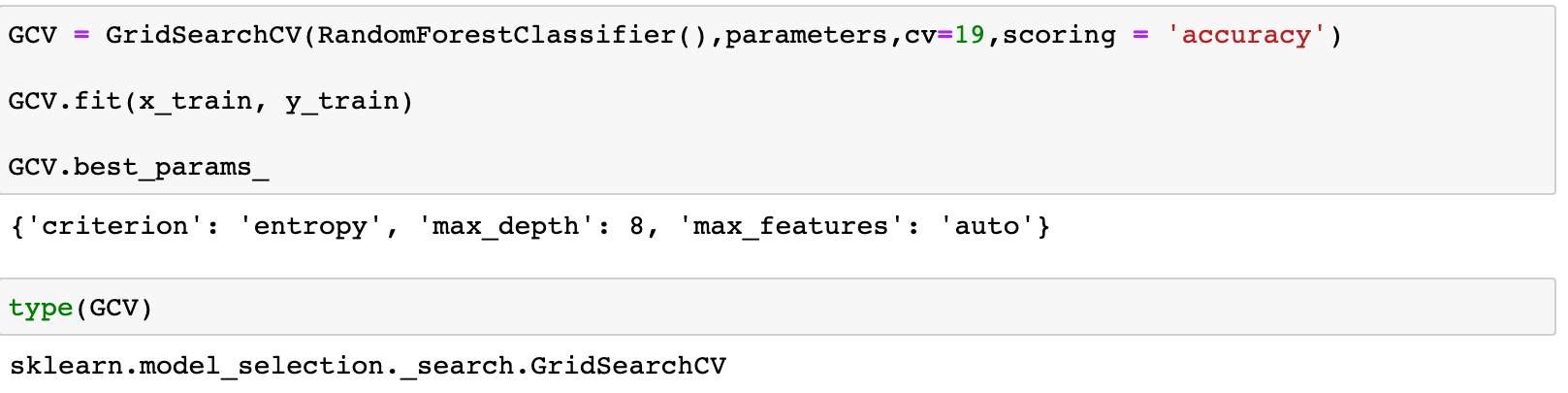
There is also another way to evaluate a random-forest classifier, which is probably much more accurate than the score we used before. What I am talking about is the out-of-bag samples to estimate the generalization accuracy. I will not go into details here about how it works. Just note that an out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set-aside test set.



Hyper Parameter Tunning of For The Model:

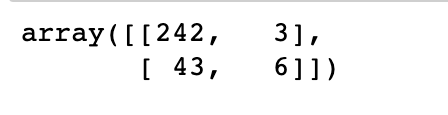
Below you can see the code of the hyperparameter tuning for the parameters criterion, min\_samples\_leaf, min\_samples\_split, and n\_estimators. I put this code into a markdown cell and not into a code cell because it takes a long time to run it. Directly underneath it, I put a screenshot of the grid searches output.



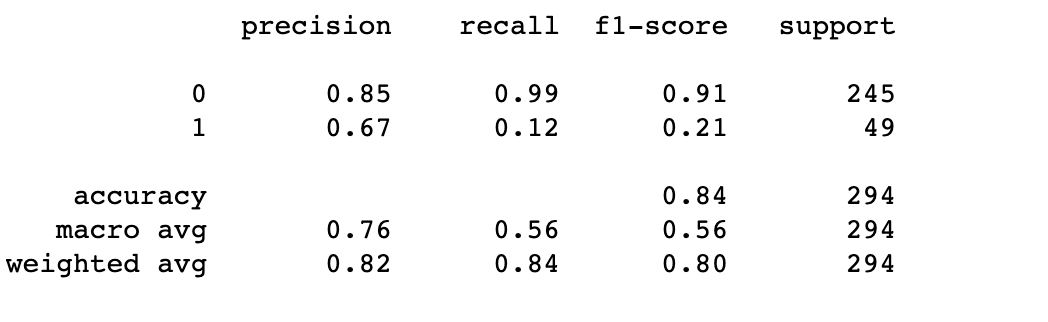


From this hyperparameter tunning we did the Grid SearchCV for our classifier under the cv as 19 and we got the results ‘entropy’ as criterion, ‘8’ as max\_depth, and ‘auto’ as max\_feature. Now that we have a proper model, we can start evaluating its performance in a more accurate way. Previously we only used accuracy and the oob score, which is just another form of accuracy. The problem is just, that it’s more complicated to evaluate a classification model than a regression model.

**Confusion\_Matrix:**

****

**Precision And Recall:**

****

Our model predicts 85% of the time, employee attrition (precision). The recall tells us that it predicted the attrition of 99 % of the people who actually survived.

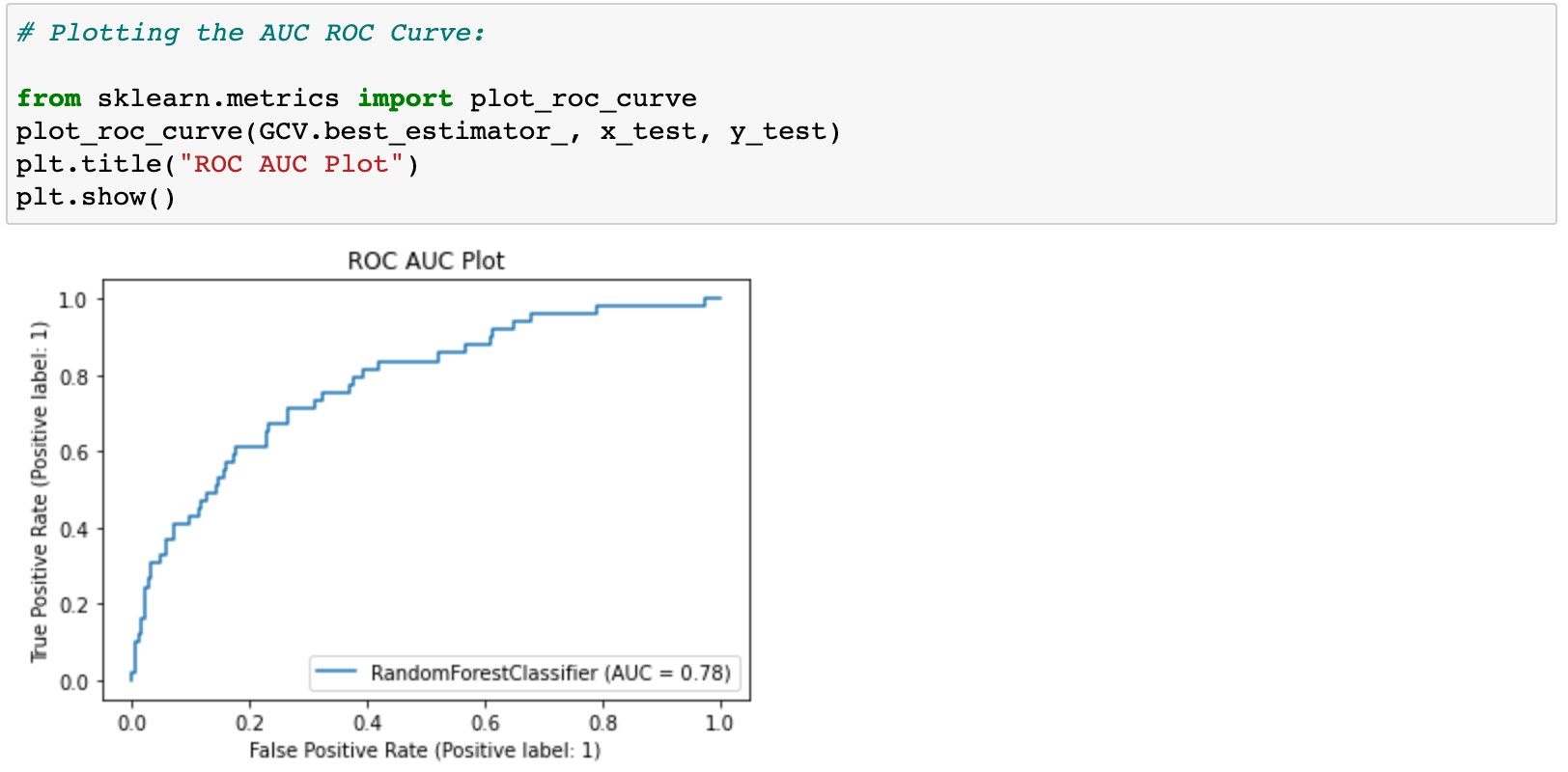
## 

## 

## 

## **ROC AUC Curve :**

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve. This curve plots the true positive rate (also called recall) against the false positive rate (ratio of incorrectly classified negative instances), instead of plotting the precision versus the recall. Our Random Forest model seems to do a good job.



**Conclusion:**

The employees are more concerned with the materialistic objects that they get directly in hand. Then comes the psychological variables that determine if an employee might leave the organization. Hence, HR can focus on such aspects and understand them from the viewpoint of the employees. Once that is followed, the project that is called the Attrition project can be used as a Retention project. This can immensely help the organization.

Secondly, the model needs to be tuned from time to time as and when a new dataset is received. In case any new input variable is introduced, it is important that the information is retrieved for the employees who participated in the initial study.

**Summary**:

We started with the data exploration where we got a feeling for the dataset, checked about missing data, and learned which features are important. During this process, we used seaborn and matplotlib to do the visualizations. During the data preprocessing part, we converted features into numeric ones, grouped values into categories, and created a few new features. Afterward, we started training the machine learning model, (random forest) and applied cross-validation to it. Then we discussed how random forest works, took a look at the importance it assigns to the different features, and tuned its performance by optimizing its hyperparameter values. Lastly, we looked at its confusion matrix and computed the model's precision, recall, accuracy\_score, and ROC AUC curve.