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

How would you feel when 5 or more of your employees wish to leave their job? You will be shocked? Losing your employees means you need to find their replacements to keep you company on track. Finding replacements is a hectic process which takes time, energy and a bit of money too. Job postings, hiring processes, paperwork, and new hire training are some of the common expenses of losing employees and replacing them. If you keep losing and replacing your employees, you will always lack experience in your firm and collective knowledge base which will have a huge impact on your business. 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.

Companies hire a lot of employees every year. They invest time, energy and money for those employees and their existing employees as well. These programs aim to increase efficiency and effectiveness of their employees. This is where HR Analytics comes into picture. HR analysis helps HR professionals to know amount of efficiency and effectiveness of their employees. Based on insights of the analysis important decisions regarding the employees are taken. The main objective of HR Analytics is gathering data of the employees, analysing and to provide insights and give better returns on investments by increasing performance of the employees.

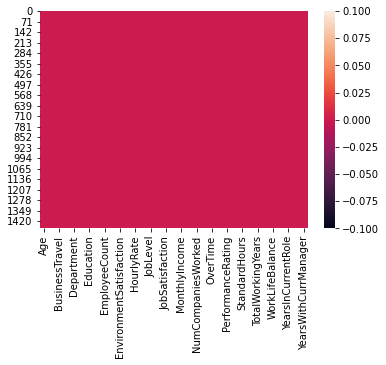
Gradual loss of employees refers to Attrition in Human Resource. In general, relatively high attrition is problematic for companies. HR professionals often assume a leadership role in designing company compensation programs, work culture, and motivation systems that help the organization retain top employees. So, as we read in the above paragraph that HR Analytics analysis the data gathered. Let us find out how does it help us analyse Attrition by writing code and using machine learning.

For analysing the IBM HR Attrition Rate Analytics dataset, we will use Jupyter Notebook for writing Python codes. Always starting with a python code, we firstly import pandas library and numpy into the notebook. Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages. Pandas works well with many other data science modules inside the Python ecosystem and is typically included in every Python distribution. Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data, including Loading the dataset, Data cleansing, Data normalization, Data visualization and much more. In fact, with Pandas, you can do everything that makes world-leading data scientists vote Pandas as the best data analysis and manipulation tool available. Numpy is a Python package which we have mainly used for linear algebra, apart from this Numpy is used for various other purposes.

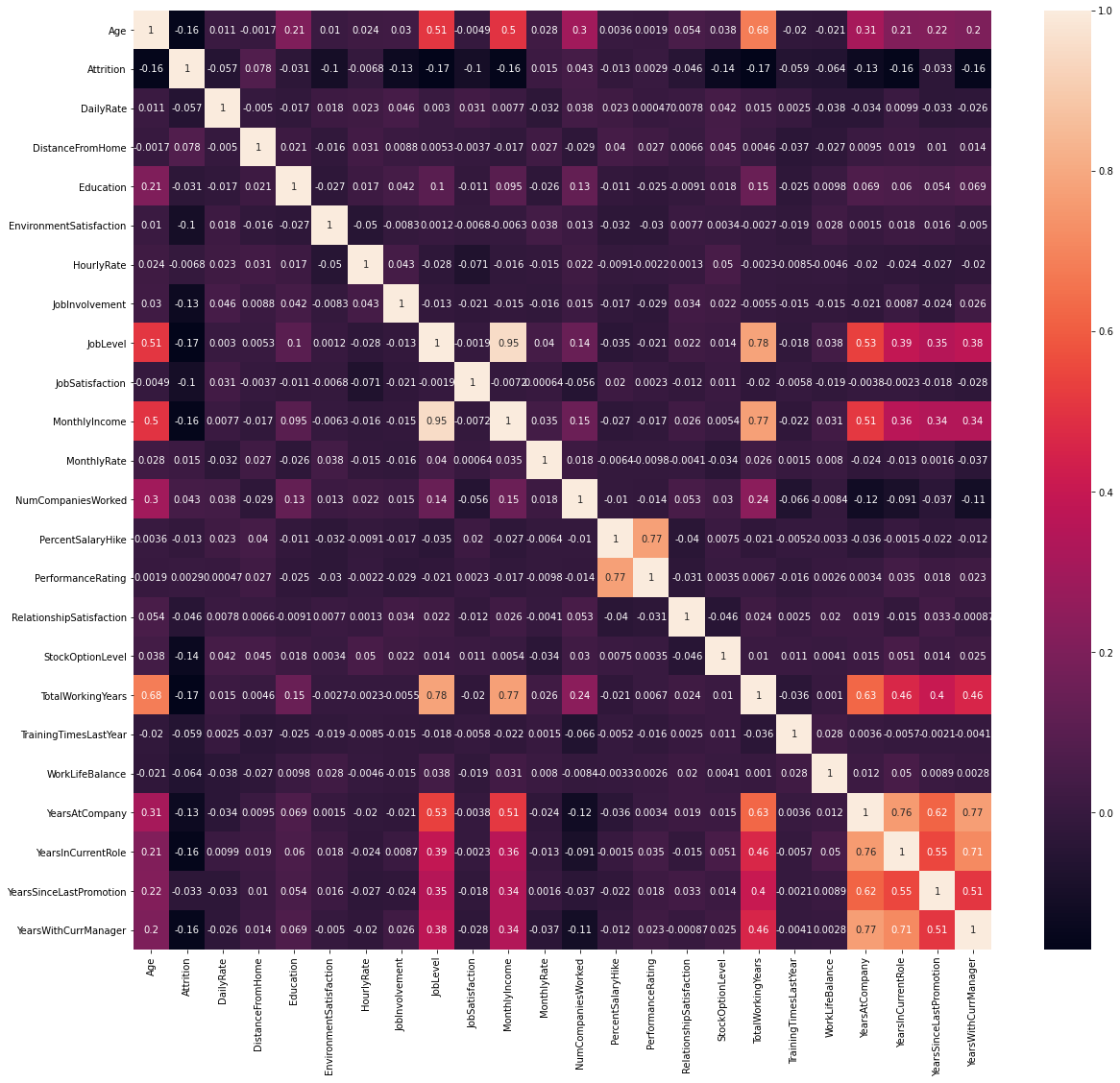
After importing libraries for mathematical and statistical analysis, libraries for visualization are imported, which include matplotlib and seaborn. Seaborn utilises fascinating themes, while matplotlib is used for making basic graphs. Also we can import warnings.filterwarnings(‘ignore’) which is called to not to print any warnings for the user in the program. Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program.

For analysing the data which we want to analyse, first thing we do is import the dataset into the notebook. Dataset come in various formats, but we convert them into an excel file that is “.xlsx” or a “.csv” file and upload it into Jupyter.

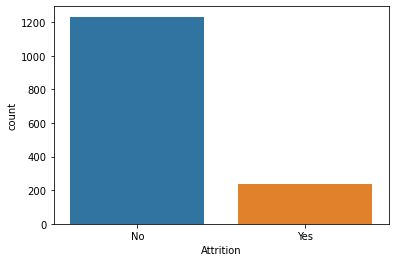
Once we upload it as a csv file, we can directly import using “ pandas.read\_csv(‘ Name of the file.csv ’) ”. If we want to import an excel file we can import the file by using “ pandas.read\_excel(r ” PATH OF THE FILE.xlsx ”) ”. In my file I have used “ csv ” file. I named the file as “ df ”. So my code goes as - df=pd.read\_csv(“ HR Analytics.csv ”) where pd is pandas as I have imported Pandas as pd and “ HR Analytics “ is the name of the dataset csv file. We start with df.head() and df.tail() which gives the output as the first 5 rows of the dataset and last 5 rows of the dataset respectively. The output contains the dimensions of the table at the bottom i.e., 5 rows x 35 columns. There are various columns specified in the table are as follows : Age, BusinessTravel, DailyRate, Department, DistanceFromHome, Education, EducationField, EmployeeCount, EmployeeNumber, Gender, JobInvolvement, Joblevel, JobRole, JobSatisfaction, MaritalStatus, MontlyIncome, MonthlyRate, NumCompaniesWorked, Over18, Overtime, PercentSalaryHike, PerformanceRating, RelationshipSatisfaction, StandardHours, StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, WorkLifeBalance, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion and YearsWithCurrManager. The dataset can be consider a good dataset as it cover many parameters which affects Attrition, from a person’s age, business travels, how much he/she has to travel from home, his/her qualification, qualification field and so on to his/her marital status, gender and many more. The dimension of the dataset is called by using df.shape() where df is name of the dataset that I have defined while importing and .shape() is the pre=defined function for calling out dimension of the dataset. The output was 1470 rows x 35 columns. In a dataset there are chances that we find some missing or null values present, so for checking on missing or null values I used df.isnull().sum() which shows the list of dataset columns and number of non null values in it. I did not find any null values in it. For better understanding sns.heatmap(df.isnull()) was used to visualize if there are any null values present. There is no null values present as there is no empty space is present (Figure shown below).



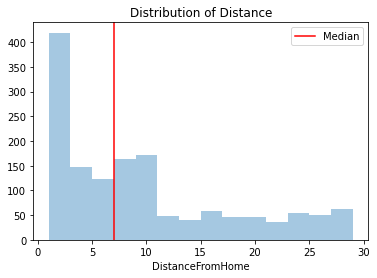
Df.info() gave the information about the dataset. Information was regarding no null values and data type(Dtype) of the values of each columns. It was found that the dtypes of columns were ‘int64’ (integers) and ‘object’(string). To check mean, median, standard deviation, 25th percentile, 75th percentile, total counts and maximum values of each columns I have used df.describe(). df.descirbe() helps to view all the above mentioned into a table altogether. There is no need to call all functions for individual columns. To check the correlation sns.heatmap(df.corr()) was used as it shows correlation between the dataset columns using colour scheme. Lighter the cell colour more the correlation between those two columns and vice versa. Figure mentioned below.



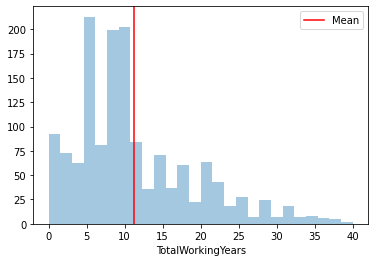
Some columns such as EmployeeNumber, EmployeeCount and StandardHours were dropped as they were not as important as other columns. Df.drop() is what was used for dropping the columns. In the Attrition column there were two values ‘Yes’ and ‘No’ , which were replaced with ‘1’ and ‘0’ respectively. They were replaced with numeric as it will be used for further machine learning process. To know value counts of the Attrition columns’ values, sns.countplot() is used it also outputs a visual for the value counts of the Attrition columns. Looking at the visuals it was observed that there were almost 1200 Nos and remaining were Yes.



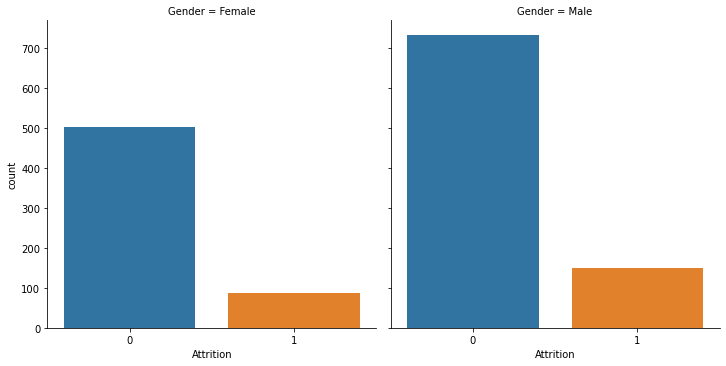
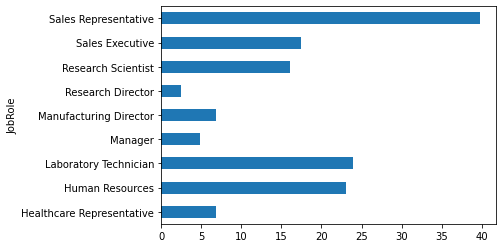
Now let us find how much does the people have to travel for their job by visualising it. So in the below image we can find

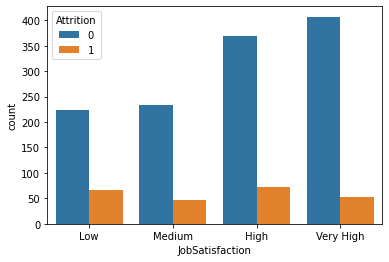
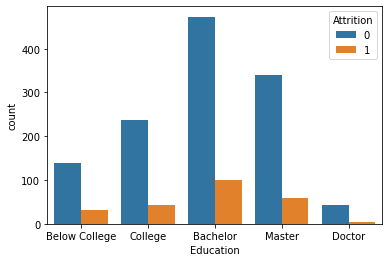


That nearly 400 people have to travel 2-3 kms, many people have to travel 5-10 kms. Mean distance people have to travel from their home to office is 6-7 kms. Some people travel more than 20-25 kms per day from home to their workplace.



In the above fig. Total working years of employees are shown. The figure elaborates count of people working of number of years. Many people work 5-10 years but some work less and some work more than 10 years. Some people work for more than 20 years for a company. But mean of a person working for a company is 11-12 years.



Comparing genders for No and Yes for Attrition in the first image. The image was made using sns.countplot() where we can nearly 500 Females vs 700 Males have not experienced Attrition and 100 Females and 150-170 Males have gone through Attrition.

In the second image, it is observed that 40% of the Job Roles are Sales Representative, followed by Lab Tech, HR, Sales Executive, Research Scientist, healthcare, manufacturing Manager and last is Research director.

In the 3rd picture, around 400 people as No are with extremely high job satisfaction and only 50 as yes. Around 350 people as No are with high job satisfaction and only more than 50 as yes. Around 250 people as No are with medium job satisfaction and only less than 50 as yes. Around 400 people as No are with low job satisfaction and only 50 as yes.

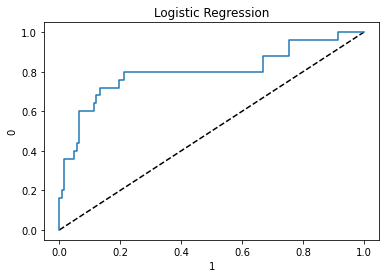
In the last graph above, Bachelor educated people have highest number of Nos and also highest Yes compared to others. Doctors have least number of Nos and Yes.

Later using pd.get\_dummies() I have converted columns with dtype as ‘object’ to dummies and dropped the true columns, to get rid of the columns having dtype as ‘object’. For quick data cleaning and EDA, it makes a lot of sense to use pandas get dummies. However, if I plan to transform a categorical column to multiple binary columns for machine learning, it’s better to use OneHotEncoder(). There are some outliers usually present in the dataset, so I have checked if there are outliers present in our dataset with the help of boxplot and found some outliers in few columns of the dataset. To handle outliers I took help of zscore imported from scipy.stats. So checking zscore, I have removed rows which had zscore more than 3 because zscore more than 3 means the value is away from the other values which we call as an outlier. Also checking skewness and handling skewed data is as important as handling outliers as they affect the results. So for handling skewness I have checked skewness using df.skew() and for handling skewness, PowerTransformer from pre.processing was imported. In powertransformer I have used (yeo-johnson method). Also from pre.processing I have imported MinMaxScaler for scaling the data. Transform features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

Let us separate our independent variables altogether from our dependent variable i.e. x and y for train test split. So to all our independent variables I have denoted x and for dependent variable as y that is ‘Attrition’. It’s time for splitting the data into training data and testing data. For splitting the data train\_test\_split was used where test\_size was 20% and random state=123. So finally we got out x\_train, y\_train, x\_test and y\_test and their shape is 665x44, 665x1 , 167x44 and 167x1 respectively. x\_train contains the training independent variables where as x\_text contains testing independent variables, y\_train contains training dependent variable and y\_test contains testing dependent variable.

For modelling the data, I have used Logistic Regression, Random Forest Classifier, Decision Tree Classifier and KNeighbors Classifier. While checking the model most importance is given to accuracy score followed by confusion matrix and to rest. So, while evaluating most accuracy score was of Logistic Regression with 87% accuracy. Worst performing model was Decision Tree Classifier as its accuracy was 74%. Also while using cross validation Logistic regression came out to be the best performing model with score mean of 87.14% and worst was Decision Tree Classifier with an improved score mean of 78.12%.

Hyper parameter tuning concluded that Logistic Regression was the best performing model after training the model with best extracted parameters such as 'C': 1.0, 'multi\_class': 'multinomial', 'penalty': 'l2' and accuracy score came out to be 87%. While after Hyper parameter tuning Random Forest Classifier’s accuracy score was 84%. So, finally concluding that Logistic Regression was the best model for our dataset which predicted with accuracy score of 87% and also True Positive was 133, False Positive 13, False Negative was 9 and True Negative was 12. As concluded, I saved Logistic Regression model with parameters as 'C': 1.0, 'multi\_class': 'multinomial', 'penalty': 'l2. We found tpr, fpr and threshold to get the AUC-ROC Curve. AUC-ROC Curve for Logistic Regression looks good as its score is more than 70%

.