ALIAKBAR MOHSIN

This is a project on Machine learning using Python to analyse the Human Resource department of an organization. It aims to enhance the performance of the employees.

HR Analytics Project

Attrition prediction

**Problem Definition**

Every year hiring is done on a huge scale by lot of companies and these companies invest lot of time and resources in training these employees. Training sessions are conducted not only for the new hires but also for the existing employees to improve the efficiency of their employees and also for the overall well-being of the company. So, where does HR Analytics fit in here and what will be the impact of it?

**Attrition in an organization || why workers quit?**

Employees are the backbone of the organization. Organization's performance is heavily based on the quality of the employees. Challenges that an organization has to face due employee attrition are:

* Expensive in terms of both money and time to train new employees.
* Loss of experienced employees
* Impact in productivity
* Impact profit

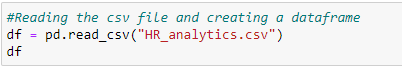
Before getting our hands dirty with the data, first step is to frame the business question. Having clarity on below questions is very crucial because the solution that is being developed will make sense only if we have well stated problem.

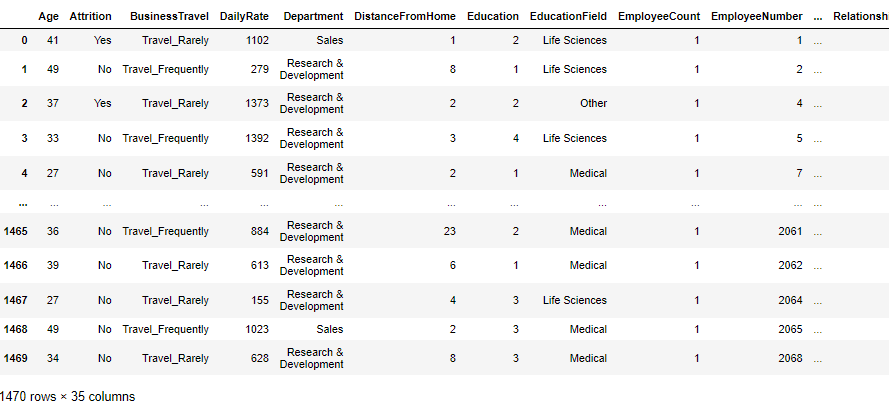
**Business questions to brainstorm**

* What factors are contributing more to employee attrition?
* What type of measures should the company take in order to retain their employees?
* What business value does the model bring?
* Will the model save lots of money?
* Which business unit faces the attrition problem?

**Data Analysis**

Let’s start by importing the dataset.



Here, I’ve read the CSV file which was saved in my disk as “HR\_analytics” and stored it in a variable “df”.

I got the output as a Dataframe. By looking into the dataset, I can roughly say that:

* There are 35 columns and 1470 rows.
* We have both numerical and categorical columns.
* There are some unnecessary entries.
* I know that “Attrition” is my target column. Also, target column data looks categorical. So I can conclude that this project is a **Classification problem**.

So now, I’ll clean the data.

**Data preparation & cleaning**

* Firstly, we have to do some statistical analysis like checking shape, null values, spaces, duplicates, nunique, value counts, info etc.

On performing the above checks, I got to know that 'EmployeeCount', 'Over18', 'StandardHours' has only 1 unique value and 'EmployeeNumber' has 1470 unique values which equals our total rows. These features aren't useful for us, hence I'm going to drop these columns.



* Next, I encoded the target variable using label encoder so that I can include it in all future analysis.



* After that, I categorized all the features according to the data type and unique values i.e. :
  + **Categorical Features** - Getting columns with dtype "object" and unique values less than 30
  + **Numerical Features(with unique values)** - Getting columns with dtype other than object and unique values less than 30
  + **Continuous Features(without unique values)** - Getting columns with dtype other than object and unique values more than 30

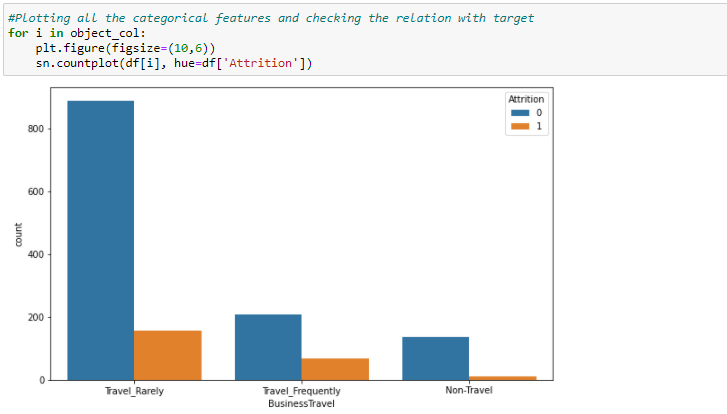
**Visualization**

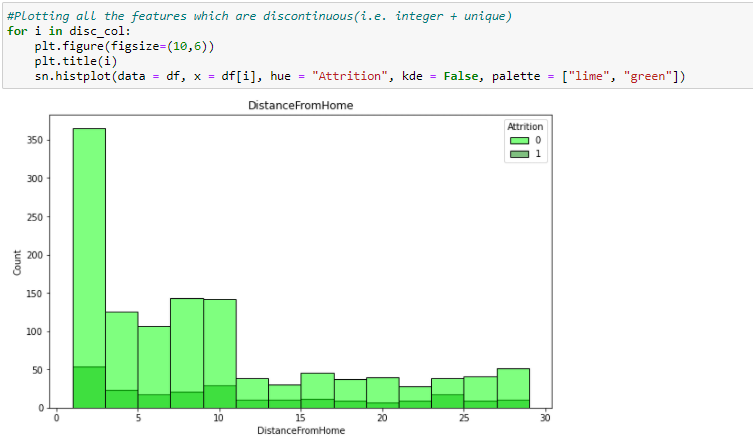
Finding patterns in data through data visualization and will reveal hidden secrets of the data through graphs, analysis and charts.

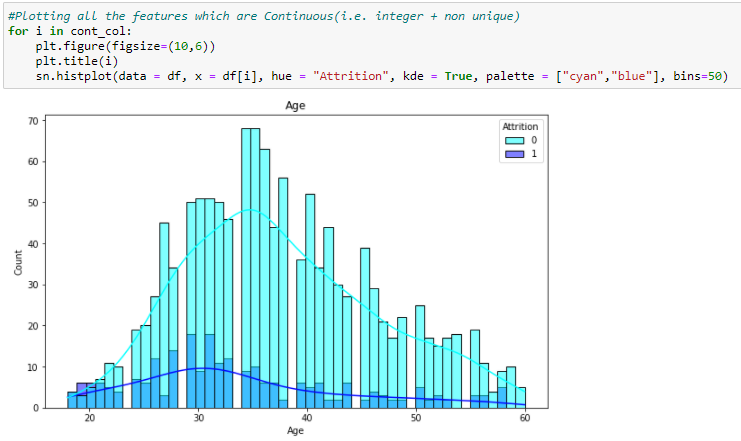
**Univariate analysis**

For this, I’ve used histogram which gives us understanding about the central tendency and spread. I’ve also taken hue of our target variable “Attrition” which will help us to check the Attrition in each category.

* Plotting all the **categorical** features and checking the relation with target



* Plotting all the features which are **discontinuous**(i.e. integer + unique)
* Plotting all the features which are **Continuous**(i.e. integer + non unique)

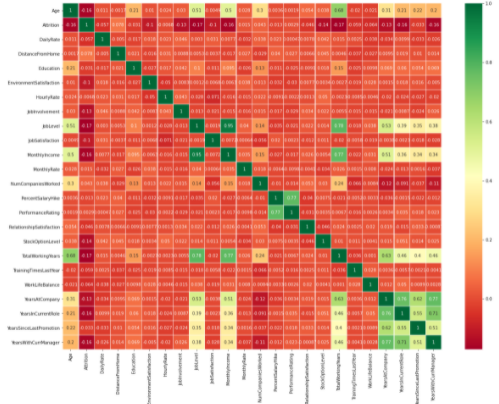


**Multi-variate analysis**

For this, I’ve used heatmap and bar plot which gives us understanding about the correlation of features with target variable.

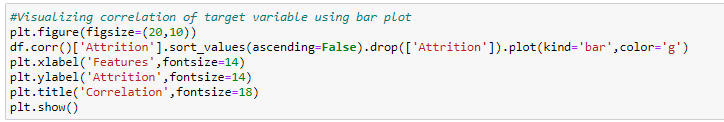
* **Heatmap**

****

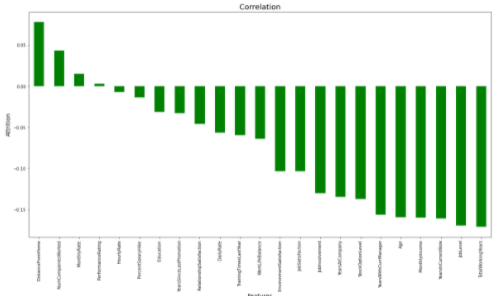
I get the output as a heatmap:

* I don't see any feature highly correlated with my target
* Job level is highly correlated with MonthlyIncome & TotalWorkingYears.
* TotalWorkingYears is highly correlated with Age, JobLevel, MonthlyIncome.
* YearsAtCompany is highly correlated with YearsInCurrentRole, YearsWithCurrentManager
* YearsInCurrentRole is highly correlated with YearsWithCurrentManager.
* **Bar plot(with correlation)**

I’ve used bar plot with corr() to display the correlation of features with the target variable. I’ve also used sort function to sort the features from highly positive to highly negative.

****

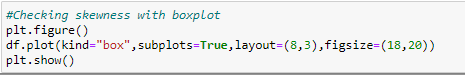
I get the following result:



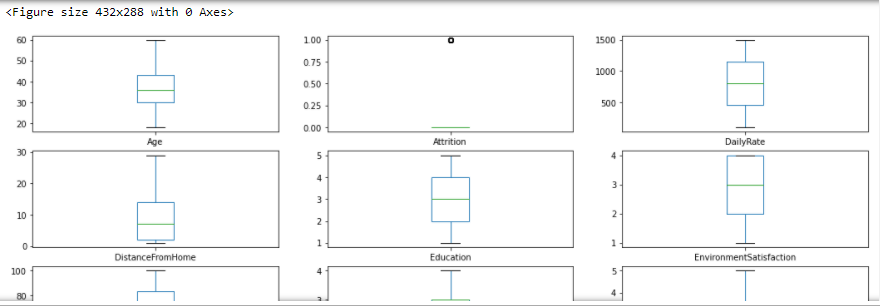
From the above plot, we can see that most of our features are negatively correlated. Also, Performance rating shows least relation with my target. So we can drop that if needed.

* **Box plot**

I’ve used the box plot of all the features to check the outliers in columns.



I get the below output:



As observed from boxplot, there are outliers present in the below mentioned columns:

MonthlyIncome, NumCompaniesWorked, PerformanceRating, StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager

An important point to note here is that **we’ll remove the outliers and skewness only in continuous columns**.

**EDA Concluding Remarks**

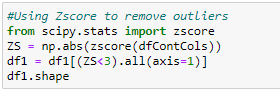
* The workers with low JobLevel, MonthlyIncome, YearAtCompany, and TotalWorkingYears are more likely to quit their jobs.
* BusinessTravel: The workers who travel a lot are more likely to quit then other employees.
* Department: The workers in Research & Development are more likely to stay then the workers on other department.
* EducationField: The workers with Human Resources and Technical Degree are more likely to quit then employees from other fields of educations.
* Gender: The Male are more likely to quit.
* JobRole: The workers in Laboratory Technician, Sales Representative, and Human Resources are more likely to quit the workers in other positions.
* MaritalStatus: The workers who have Single marital status are more likely to quit the Married, and Divorced.
* OverTime: The workers who work more hours are likely to quit then others.

**Pre-processing Pipeline**

* **Treating Outliers**

I’ll be using **Zscore** to treat the outliers. Below are the steps to do so:

1. Copy df in df1(new variable)
2. Making a variable as dfContCols and listing all columns with outliers.
3. Using Zscore to remove outliers

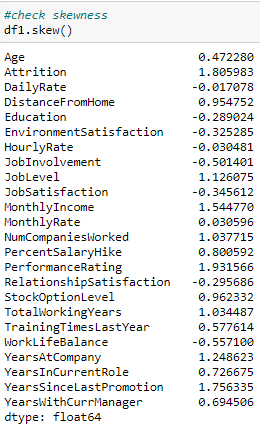


1. Check the data loss

After performing all these steps, I got the data loss of **5.64%**. As the data loss is below 10%, I can proceed with Zscore. Else, I had an option to use IQR method.

* **Treating Skewness**

To check the skewness in all our columns, we can use **skew()**

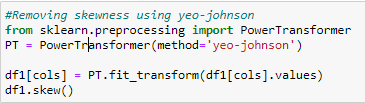


Please note that the skewness range I’ve considered is **+/-0.5**

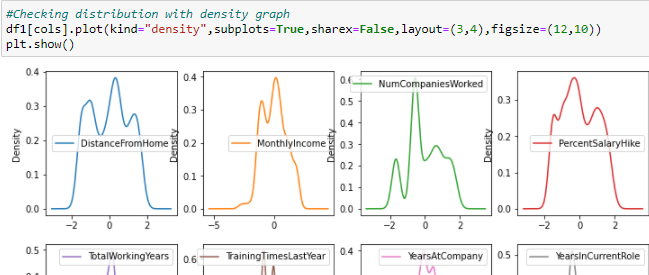
So anything above or beyond this range is considered skewed. Again, we’ll only treat continuous columns.

Below mentioned are the steps to treat skewness:

1. Create list of all the columns with skewness. Here we have ["DistanceFromHome","MonthlyIncome","NumCompaniesWorked","PercntSalaryHike","TotalWorkingYears","TrainingTimesLastYear","YearsAtCompany","YearsInCurrentRole","YearsSinceLastPromotion","YearsWithCurrManager"]
2. Use PowerTransformer to remove skewness. The method I used here is **yeo-johnson**



1. Check distribution with density graph.

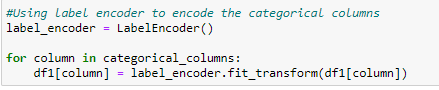


* **Label Encoding**

As we all know, we need to encode the categorical columns. The reason is that our machine will only accept numeric values. To do this, we have multiple methods but in this project, we’ll use **LabelEncoder.**

Steps to encode our columns are:

1. Create a list of non-numeric columns(i.e. object etc)
2. Import Label encoder
3. Create instance of Label encoder and fit\_transorm on categorical columns



Label encoder will assign integer values starting from 0 according to unique values in that particular column.

After this, we can notice that all our variables are numeric.

* **Splitting independent & target variable**

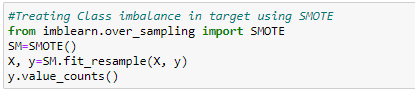
In this section, we’ll split the dataset in 2 parts. First is **X** where all our independent columns will be placed and other will **y** here our target variable will be available.



* **Treating class imbalance**

Class imbalance is nothing but the significant difference in value count of classes in the target column.

In this case, we found our target column has class imbalance problem, so we needed to make the target balanced. For this, we applied over\_sampling on the dataset.



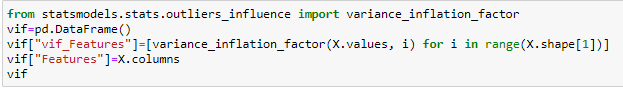
* **Scaling**

I have to scale my independent features to get the same range in all the columns. If I don’t scale my independent columns then there is a chance that my model may get baised. So In this particular case, I have used Standard scaling as I have removed all outliers and skewness from the dataset it is good to use standard scaling else we have MinMax scaler.

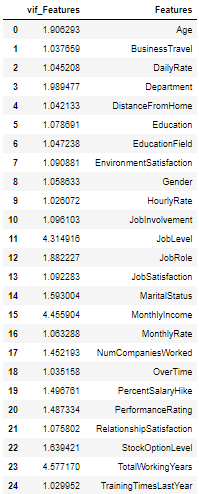


* **Using VIF(Variance inflation factor) to remove multicolinearity**

Now scaling part is done. But I’m left out with multicolinearity. I have to check VIF(variance inflation factor) now.



I get the below output:

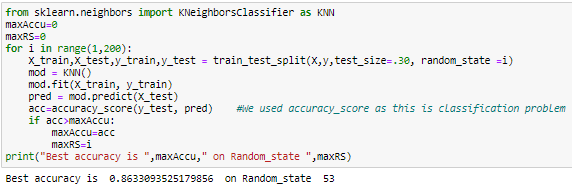


I notice that entire features show VIF below 10. Hence, I can conclude that there is no multicolinearity issue in the dataset.

* **FIND BEST random\_state**

It’s important that we find the best random state so that our model gives better accuracy.

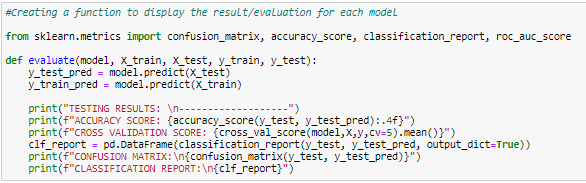
We first created two variables to store the score and the random state and assigned value 0 to them. After that we took a range of 1 to 200 for the random state and created the train test split and then performed model fitting using each random state. We then kept the highest score in one variable and the random state for that score in another. Finally, we printed the score and the random state.

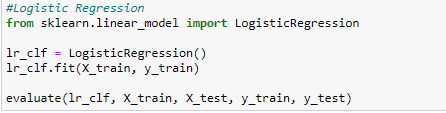
****

Now, using the random state that we found giving highest score, we made the train and test split keeping the test size of 30%.

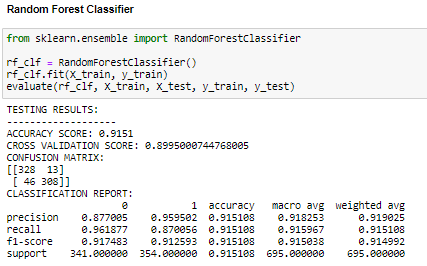
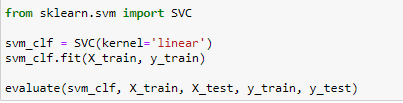
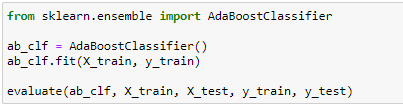
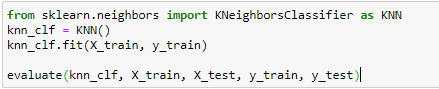
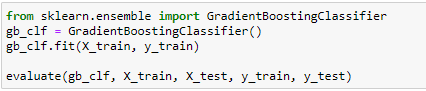
* **Applying machine learning algorithms**

Before we train our model, I’ve created a function called “**evaluate**” which will be used to display the Accuracy score, Cross validation score, Classification report and Confusion matrix.

****

1. **Logistic Regression  
   **

Logistic regression model is giving me 85% accuracy\_score and the cross validation is 82%. LR is working well but I cannot conclude it as good model before looking into multiple models.

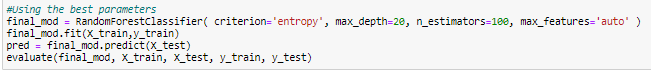
1. **Random Forest Classifier  
   **Random forest model is giving me 91% accuracy\_score and the cross validation is 90% which looks great!
2. **Support Vector Machine  
   **Support vector model is giving me 86% accuracy\_score and the cross validation is 82%.
3. **AdaBoost Classifier  
   **AdaBoost model is giving me 89% accuracy\_score and the cross validation is 84%.
4. **KNeighbors Classifier  
   **KNeighbors model is giving me 86% accuracy\_score and the cross validation is 84%.
5. **Gradient Boosting Classifier  
   **Gradient Boosting model is giving me 92% accuracy\_score and the cross validation is 86%.

**I got the accuracy\_score & cross validation scores of all the models.  
On checking the difference between accuracy\_score & cross\_val\_score, I found that Random Forest Classifier has least difference.   
The model accuracy is 91% which is good but I can improve the model accuracy by tuning it. Hence, I’ll try to improve the model accuracy now.**

* **HYPERPARAMETER TUNING**

For this, I’ve created a dictionary of parameters and found the best parameter from it using GridSearchCV.   
I got these best parameters:   
**{'criterion': 'entropy', 'max\_depth': 20, 'max\_features': 'auto', 'n\_estimators': 100}**

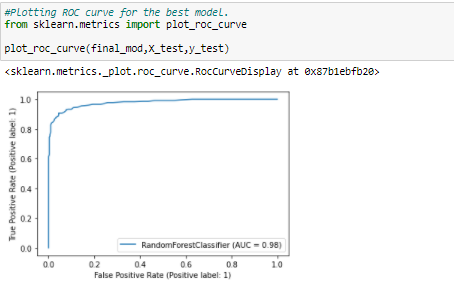
After knowing the above best parameters I have to run for improving model accuracy.

****

**After tuning, the accuracy increased to 92.23% which looks good!**

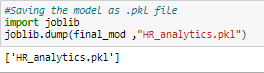
* **ROC-AUC Curve**

As this is a classification problem, I checked AUC ROC curve.  
First we imported the plot and fed the final model to it. We got the area under the curve as 98%.

****

* **SERIALIZATON (Saving model)**

Finally, we imported joblib and saved the model as “**HR\_analytics**” in **.pkl** format.



**Concluding Remarks**

With this project, we got the idea about what type of data we can work with in building a model and what type of data we should avoid.

We also found how balancing the target values in a classification problem play a crucial role.

By analysing the dataset provided in this project, we found that the attrition has a high positive correlation with overtime, as overtime increases chances of attrition also increases.

We also found that attrition has a negative correlation with the job level and monthly income, which meant that with a higher job level and a monthly income, the chance of attrition goes down.

We found how tuning the right parameters increase the performance of the model, and saw role of ROC curve in analysing the performance and finalizing a model in a classification problem.

**Thanks for reading ☺**