**HR Analytics using machine Learning**

The key to a success in an organization is to attract talented employee and retain them for long period of time. It is vital for the Human Resource (HR) Department to identify the factors that keep employees and those which prompt them to leave. Organization could do more to prevent the loss of its talented resources.

We know people are good decision maker when unbiased but certain time it gets difficult to take a decision due to their emotions, biases, prejudices etc and this result in inefficient decisions being made.

This can have negative impact in the organization, which lead to low performance in various departments in an organization. Moreover managing and analyzing a vast data manually or semi-manually is not an easy task which can eventually turn out to be a huge mistake.

This is where Machine Language comes in. With the decision-making models, when provided with the data and information, can deliver excellent error free decisions, more insight in the data of the organization which could eventually help the organization to take lot of important decisions.

To better understand the above process, let us go ahead and implement machine learning in the dataset collected from a human resource department.

To start with let’s import python libraries such as Pandas, Numpy, Seaborn, Matplotlib using Jupyter notebook.

**import** **pandas** **as** **pd**

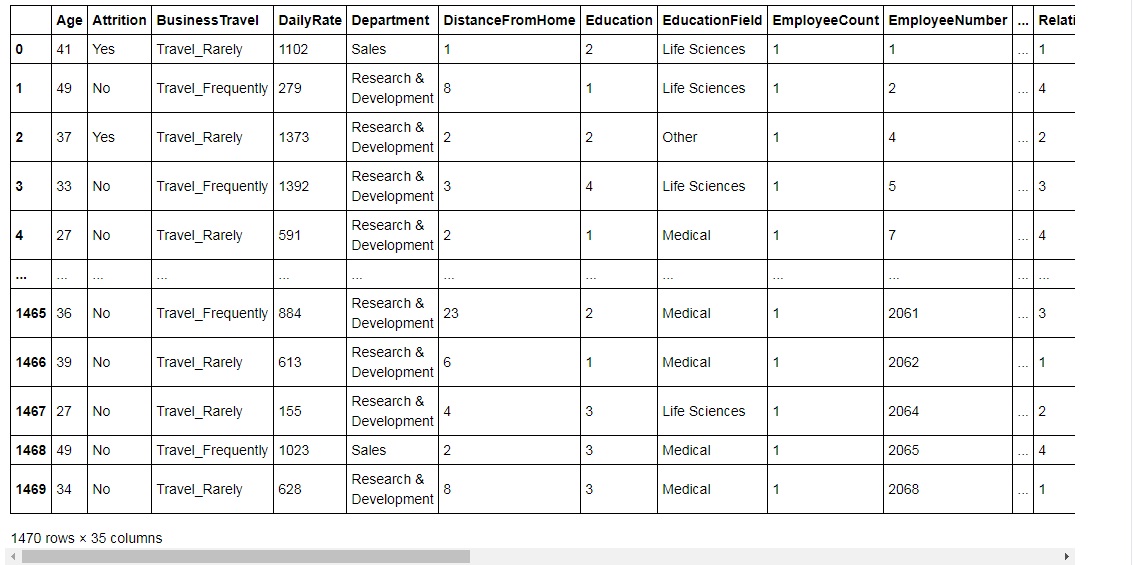
**import** **numpy** **as** **np**

**import** **seaborn** **as** **sns**

**import** **matplotlib.pyplot** **as** **plt**

After the implementation of different python libraries, now let’s import the dataset in the Jupyter notebook using the following codes:-

df=pd.read\_csv('WA\_Fn-UseC\_-HR-Employee-Attrition.csv')



There are 1470 rows and 35 columns in the dataset. With the following code, let’s find out the data type and the missing values in the Dataset:-

df.info()

Output:-

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1470 entries, 0 to 1469

Data columns (total 35 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Age 1470 non-null int64

1 Attrition 1470 non-null object

2 BusinessTravel 1470 non-null object

3 DailyRate 1470 non-null int64

4 Department 1470 non-null object

5 DistanceFromHome 1470 non-null int64

6 Education 1470 non-null int64

7 EducationField 1470 non-null object

8 EmployeeCount 1470 non-null int64

9 EmployeeNumber 1470 non-null int64

10 EnvironmentSatisfaction 1470 non-null int64

11 Gender 1470 non-null object

12 HourlyRate 1470 non-null int64

13 JobInvolvement 1470 non-null int64

14 JobLevel 1470 non-null int64

15 JobRole 1470 non-null object

16 JobSatisfaction 1470 non-null int64

17 MaritalStatus 1470 non-null object

18 MonthlyIncome 1470 non-null int64

19 MonthlyRate 1470 non-null int64

20 NumCompaniesWorked 1470 non-null int64

21 Over18 1470 non-null object

22 OverTime 1470 non-null object

23 PercentSalaryHike 1470 non-null int64

24 PerformanceRating 1470 non-null int64

25 RelationshipSatisfaction 1470 non-null int64

26 StandardHours 1470 non-null int64

27 StockOptionLevel 1470 non-null int64

28 TotalWorkingYears 1470 non-null int64

29 TrainingTimesLastYear 1470 non-null int64

30 WorkLifeBalance 1470 non-null int64

31 YearsAtCompany 1470 non-null int64

32 YearsInCurrentRole 1470 non-null int64

33 YearsSinceLastPromotion 1470 non-null int64

34 YearsWithCurrManager 1470 non-null int64

dtypes: int64(26), object(9)

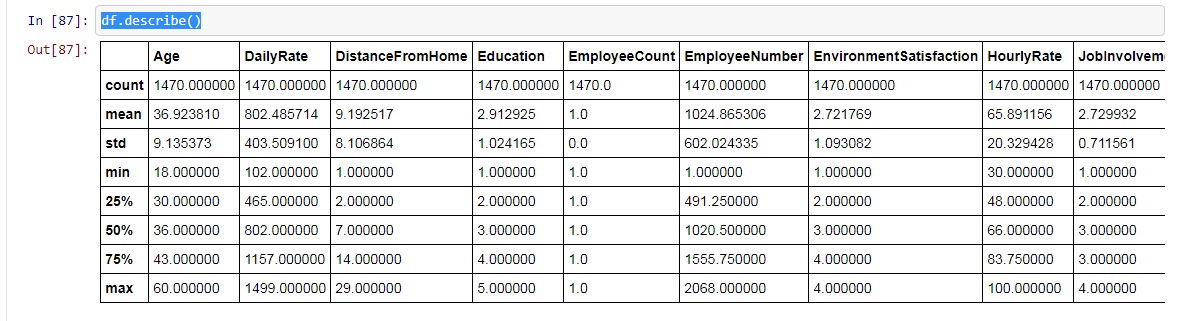
memory usage: 402.1+ KB

After checking the info of the dataset, fortunately there were no as such missing values seen.

Our target column is **Attrition,** which contain categorical values, hence the values needed to be converted to numeric values. Using **dummies,** the categorical values were converted to numeric values. Following codes were used:-

attrition=pd.get\_dummies(df['Attrition'],drop\_first=**True**) df.drop('Attrition',axis=1,inplace=**True**) df=pd.concat([df,attrition],axis=1)

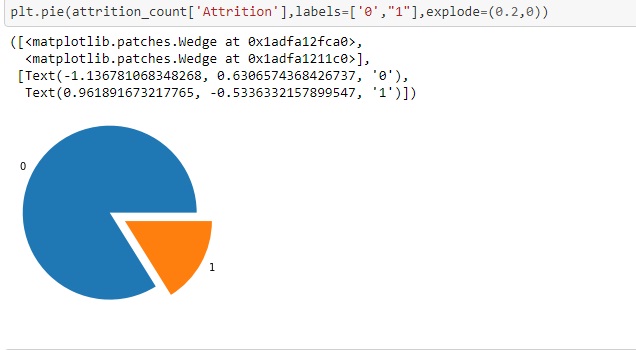
Next we check the statistical data of the dataset using the code df.describe()



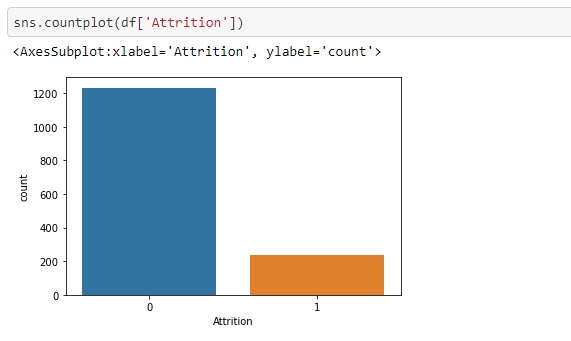
From the above statistical data we conclude that the target column i.e. is **Attrition** is concentrated towards zero and there is high standard deviation in few columns.

Now, let’s check the correlation among different columns with target variables using the code:- df.corr(). After checking out the correlation, now let’s shift to the visualization part, where we can visualize the data to have proper insight of the dataset.

Before getting into the visualization of attrition, let’s count the number of 0s and 1s where 0 represent **NO** and 1 represent **YES.** Now, let’s see the attrition data through pie chart.

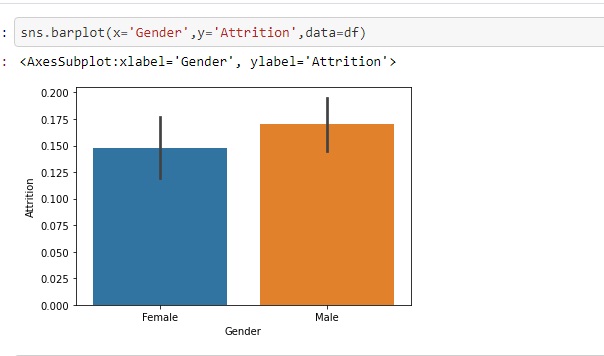


Attrition is also visualized through count plot to see the number of Yes and No.



As mentioned earlier, O represent No and 1 represent Yes hence, it is clear that the number of Yes is quite less than number of No.

In seaborn plot of attrition with respect to gender, it is observed that male attrition is more than female.

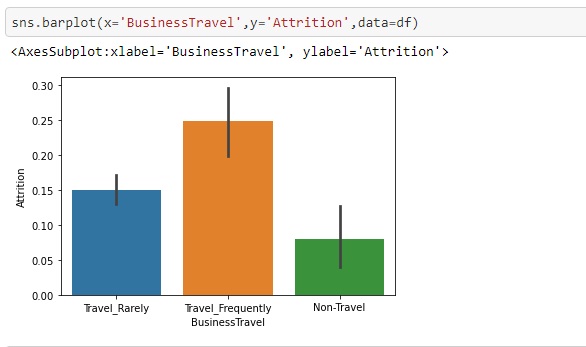


Now, let’s check the attrition with respect to department:



So, it can be easily understood that, Sales department has the highest number of attrition followed by Human Resources and Research & Development department.

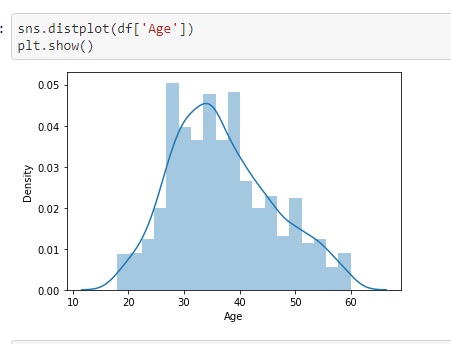
Lastly, let’s see the relation between employee’s business travels with attrition :

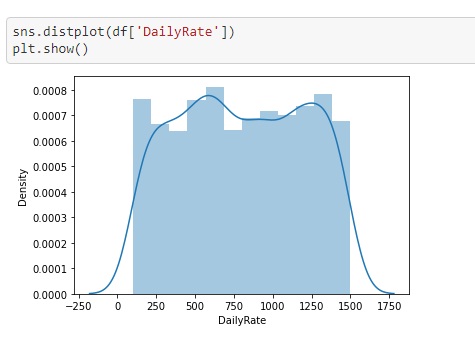


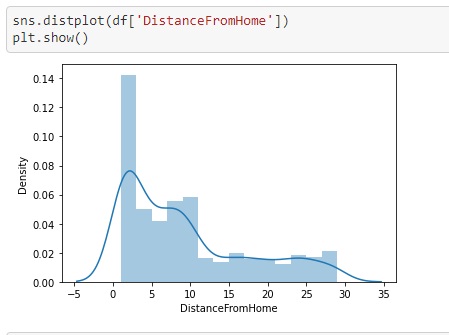
From the above visualization we can conclude that people those who travel frequently are having highest number of attrition followed by travel rarely and Non-travel.

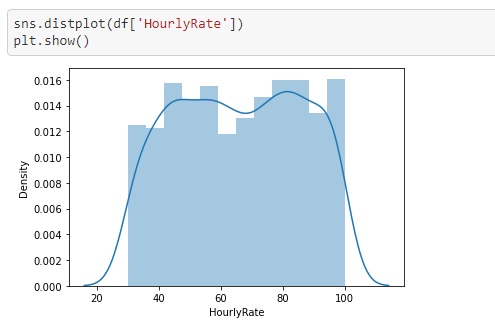
Through these above charts and count plots, we can have a clear picture of relationship between the target column and the other columns.

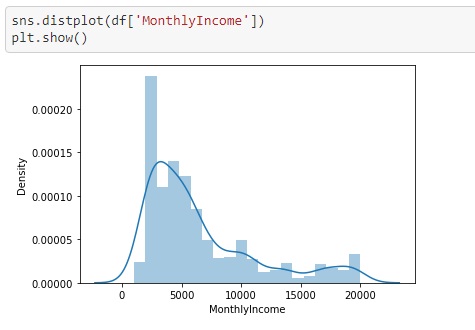
Next, let’s visualize different columns using **distplot**:

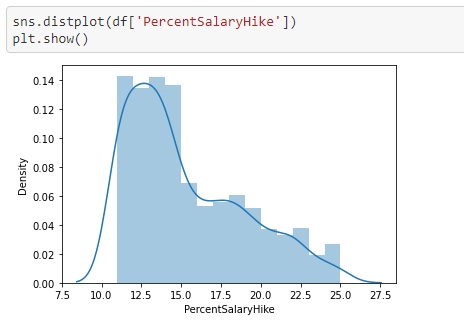












From the above **distplot** of various data of the dataset, the distribution of data along with skewness are visualized.

Now let’s convert all the categorical data into numeric data. To do so, Label Encoder is been imported from Scikit learn. Using Label Encoder, all the columns containing categorical data are converted to numeric data for further processing, using the following code:-

**from** **sklearn.preprocessing** **import** LabelEncoder

le=LabelEncoder()

**for** columns **in** df.columns:

**if** df[columns].dtype==np.number:

**continue**

**else**:

df[columns]=le.fit\_transform(df[columns])

After converting into numeric data, now let’s check for skewness using the code:- df.skew().

Following is the output :-

Age 0.413286

BusinessTravel -1.439006

DailyRate 0.000930

Department 0.172231

DistanceFromHome 0.958118

Education -0.289681

EducationField 0.550371

EmployeeCount 0.000000

EmployeeNumber 0.000000

EnvironmentSatisfaction -0.321654

Gender -0.408665

HourlyRate -0.032311

JobInvolvement -0.498419

JobLevel 1.025401

JobRole -0.357270

JobSatisfaction -0.329672

MaritalStatus -0.152175

MonthlyIncome 0.060816

MonthlyRate 0.012315

NumCompaniesWorked 1.026471

Over18 0.000000

OverTime 0.964489

PercentSalaryHike 0.821128

PerformanceRating 1.921883

RelationshipSatisfaction -0.302828

StandardHours 0.000000

StockOptionLevel 0.968980

TotalWorkingYears 1.112899

TrainingTimesLastYear 0.553124

WorkLifeBalance -0.552480

YearsAtCompany 1.676650

YearsInCurrentRole 0.917363

YearsSinceLastPromotion 1.984290

YearsWithCurrManager 0.833451

Attrition 1.844366

dtype: float64

There are skewnesses present in the dataset which can be removed using the log transformation method. As Attrition is the target column, hence it is dropped and skewness is removed from rest of the data. Using the following code we remove the skewness from the dataset.

**for** col **in** new.columns:

**if** new[col].skew()>0.55:

new[col]=np.log1p(new[col])

Next up, lets divide the dataset into X and Y ,where X is the predictor variables and Y is the target variable.

Before train-test-split, let’s scale the data of X using Standard Scaler, hence we import it from Scikit learn. Following code is been used to do so:-

**from** **sklearn.preprocessing** **import** StandardScaler

scaler=StandardScaler()

x=scaler.fit\_transform(x)

After Scaling is completed, now let’s import train-test-split using Scikit learn.

**from** **sklearn.model\_selection** **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,train\_size=.80)

With above code, the dataset is been split for training and testing the model where 80% data is for training and 20% for testing.

Now, we have come to the most important part, were we need to decide the algorithms which will best suit our dataset. As we have seen that our target variable is a categorical type, hence we have decided to take Logistic Regression, Random Forest Classifier, GaussianNB. We also need to check the accuracy score and error of our model hence we choose Accuracy score and classification report and confusion matrix respectively. Following are the code used :

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn.ensemble** **import** RandomForestClassifier

**from** **sklearn.naive\_bayes** **import** GaussianNB

**from** **sklearn.metrics** **import** accuracy\_score

**from** **sklearn.metrics** **import** classification\_report,confusion\_matrix

model=[LogisticRegression(),GaussianNB(),RandomForestClassifier()]

**for** m **in** model:

m.fit(x\_train,y\_train)

m.score(x\_train,y\_train)

predm=m.predict(x\_test)

print('Accuracy score of',m,'is:')

print(accuracy\_score(y\_test,predm))

print(confusion\_matrix(y\_test,predm))

print(classification\_report(y\_test,predm))

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print('**\n**')

After applying the above code, we get the following output:-

Accuracy score of LogisticRegression() is:

0.8639455782312925

[[240 9]

[ 31 14]]

precision recall f1-score support

0 0.89 0.96 0.92 249

1 0.61 0.31 0.41 45

accuracy 0.86 294

macro avg 0.75 0.64 0.67 294

weighted avg 0.84 0.86 0.84 294

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Accuracy score of GaussianNB() is:

0.7925170068027211

[[217 32]

[ 29 16]]

precision recall f1-score support

0 0.88 0.87 0.88 249

1 0.33 0.36 0.34 45

accuracy 0.79 294

macro avg 0.61 0.61 0.61 294

weighted avg 0.80 0.79 0.80 294

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Accuracy score of RandomForestClassifier() is:

0.8571428571428571

[[245 4]

[ 38 7]]

precision recall f1-score support

0 0.87 0.98 0.92 249

1 0.64 0.16 0.25 45

accuracy 0.86 294

macro avg 0.75 0.57 0.59 294

weighted avg 0.83 0.86 0.82 294

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**OBSERVATION:**

Logistic Regression has the highest accuracy score.

**Conclusion:**

The employees having the highest performance rating has greatest environment satisfaction and vice versa. The employee with highest performance has high percentage of salary hike. Employees working in sales department have the highest number of attrition followed by Human resource department. Moreover, it can also be seen that employees those who are travelling frequently have tendency of attrition.

HR Analytics is gaining traction in organisations that embrace digital transformation. The scope has expanded from analytics of employee work performance to providing insights so that decisive improvements can be made to organisational processes. While some level of attrition is inevitable, it should be kept at the minimal possible level.