Project on Employee Attrition Prediction

Aim - To Predict whether the employee will leave the current company (or will resign from the current company) or not, using several machine learning Models.

Problem Statement - Employee attrition is the gradual reduction in employee numbers. Employee attrition happens when the size of your workforce diminishes over time. This means that employees are leaving faster than they are hired. Employee attrition is defined as employees leaving their organizations for unpredictable or uncontrollable reasons. Many terms make up attrition, the most common being termination, resignation, planned or voluntary retirement, structural changes, long-term illness, layoffs.

Approach in solving the problem: Attrition in a corporate setup is one of the complex challenges that the people, managers and HR must deal with. We will try to visualize the problem and factors contributing to Attrition and will predict Attrition through Machine Learning Models.

Data-

Data Rows -1470 entries

Data columns - 35 entries

The project is sub-divided following sections. These are:

1.Loading necessary libraries

2.Loading Dataset from a CSV file or from a Table.

3.Summarization of Data to understand Dataset.

4. Visualization of Data to understand Dataset (Plots, Graphs etc.)

5. Data pre-processing and Data transformation (Not Required in this case as no categorical data is present in the Dataset)

6. Splitting the data set into independent & dependent sets.

7. Importing the train_test_split model from sklearn. model for splitting data into train & test sets.

8. Importing different kinds of Regression models & then training those models with the help of fit ().

9. Predicting the trained models & then checking their accuracy of the model.

10. Then, trained the test dataset with Tain dataset with the help of better accuracy model.

11. Finally, predicted the Attrition with a new data provided to the Model creating a new data frame using dictionary.

Data Visualization-

```
data['Gender'].value_counts().plot(kind='bar',color=['salmon','lightblue'],title="Count of different gender")
```

```
<Axes: title={'center': 'Count of different gender'}>
```

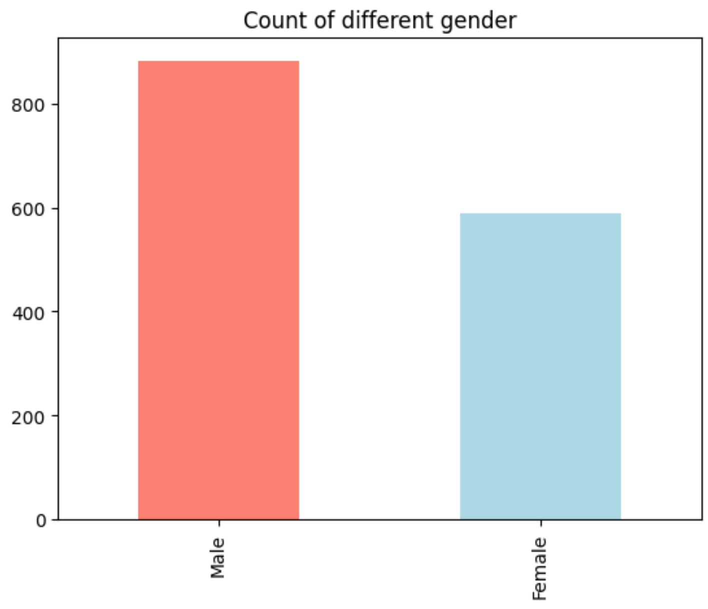


Fig1. Count of different Gender.

Inference: The graph shows the count of different Genders present in the Organization, here number of Male is higher than the Female.

```
pd.crosstab(data['Gender'],data['Attrition']).plot(kind="bar",figsize=(10,6))
plt.title("Attrition vs Gender")
plt.xlabel("Attrition")
plt.ylabel("No of people who left based on gender")
plt.legend(["Yes","No"])
plt.xticks(rotation=0)
```

```
(array([0, 1]), [Text(0, 0, 'Female'), Text(1, 0, 'Male')])
```

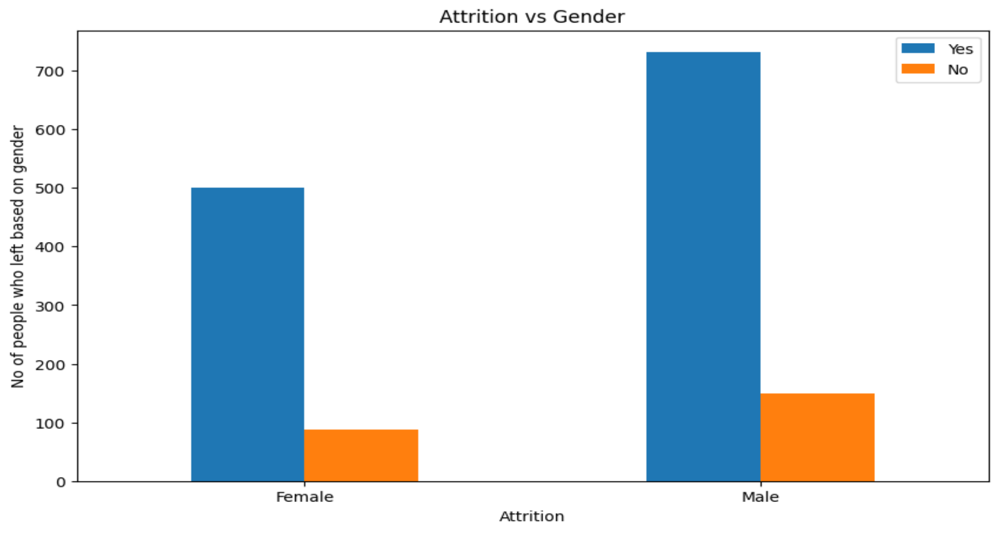


Fig2. Attrition vs Gender

Inference: Here we are trying to figure out that how gender could be the reason for employees to leave the company or to stay in, we can see that Males tend to have higher Attrition rate than Females.

```
pd.crosstab(data['YearsSinceLastPromotion'],data['Attrition']).plot(kind="bar",figsize=(10,6))
plt.title("Attrition vs YearsSinceLastPromotion")
plt.xlabel("Attrition")
plt.ylabel("No. of people who left/stay based on promotion")
plt.legend(["Attrited","NoAttrition"])
plt.xticks(rotation=0)
```

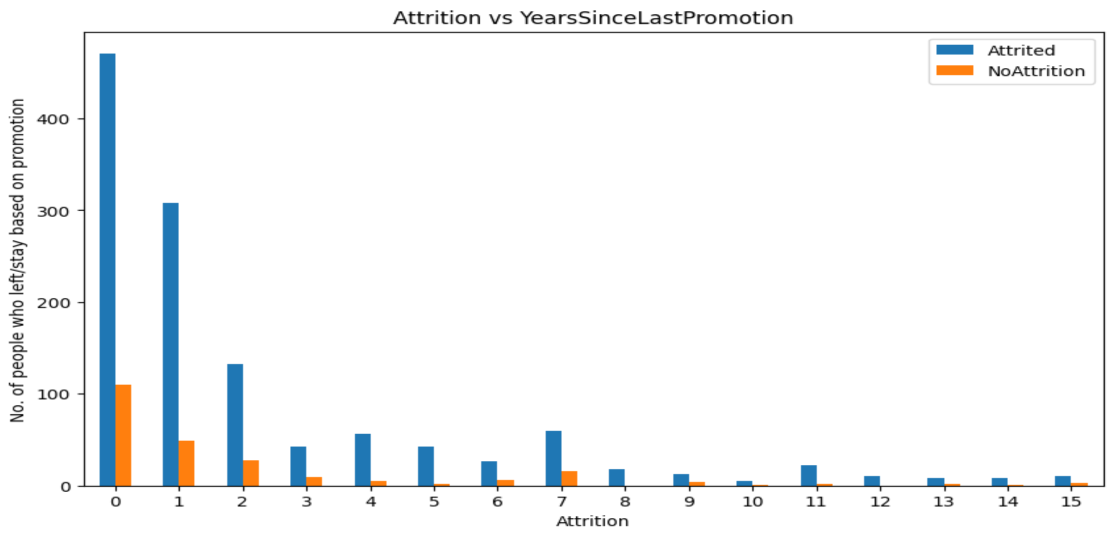


Fig3. Attrition vs YearsSinceLastPromotion

Inference: Here we are figuring out that how promotion could be the reason for employees to leave the company or to stay in, it is observed that Attrition is high during the early years when they are not promoted.

```
#snr.relplot(x='YearsAtCompany',y='YearsInCurrentRole',hue='Attrition',data=data)
snr.relplot(x='MonthlyIncome',y='Age',hue='JobLevel',col='Attrition',data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d5bf5867160>
```

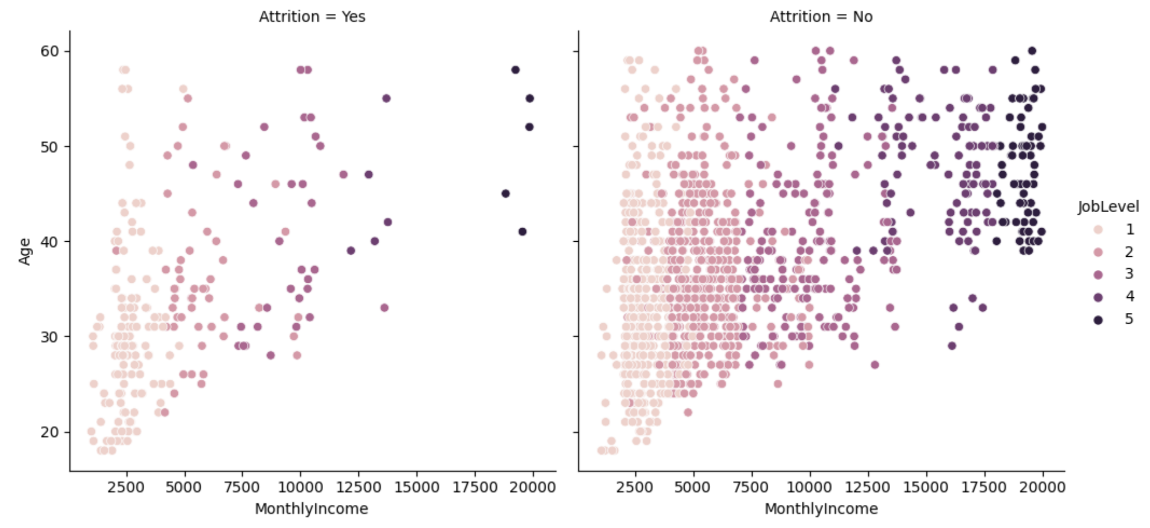


Fig4. Relational Plot showing Corelation between Attributes Monthly Income, Age, Job Level with respect to Attrition.

Inference: This plot shows that Attrition is more from employees who are below 42 years of Age and who have less monthly income and having job level 1,2 and 3.

```
[ ]  snr.relplot(x='Age',y='DistanceFromHome',col='JobLevel',hue='Attrition',data=data)

     <seaborn.axisgrid.FacetGrid at 0x7d5bb40f2650>
```

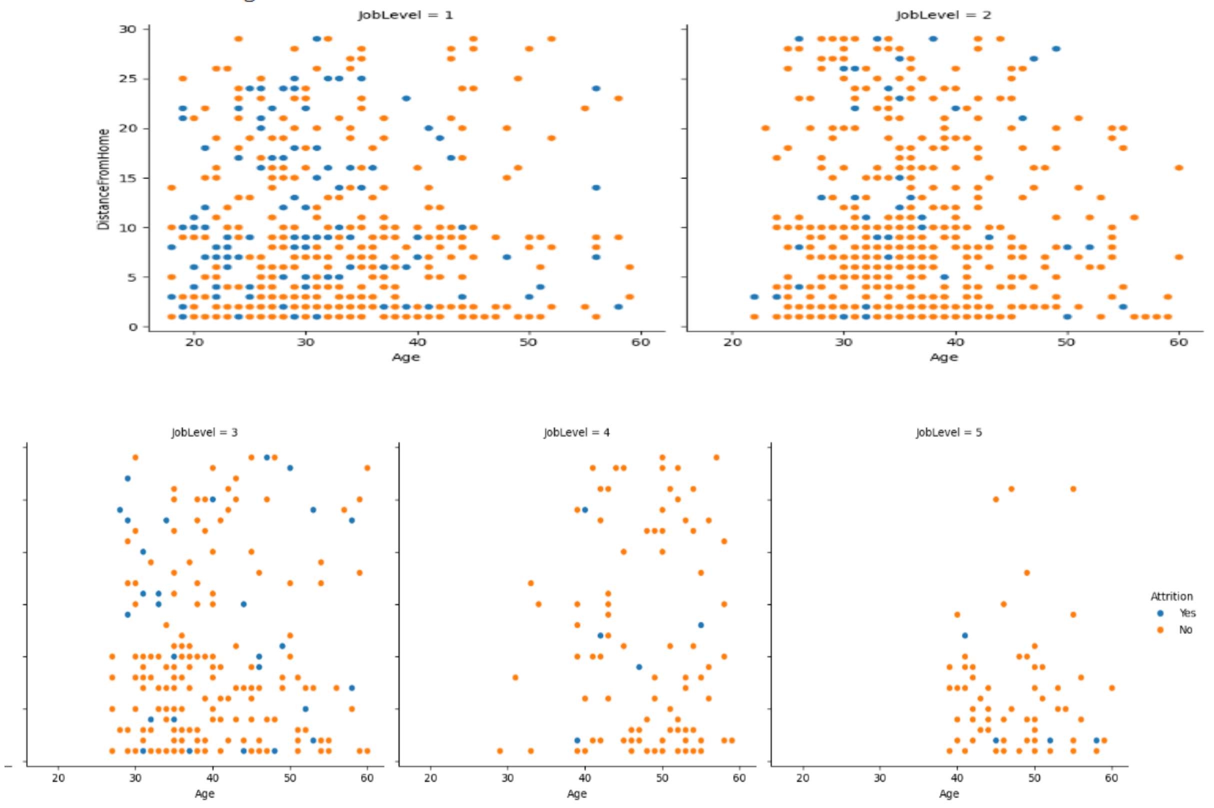


Fig5. **Relational Plot showing Corelation between Attributes Distance from Home, Age, Job Level with respect to Attrition.**

Inference: This plot shows that Attrition is more from employees who are below 50 years of Age, employees who are having more distance from home and having job level 1 and 2.

```
[ ]  snr.relplot(x='JobSatisfaction', y='JobRole',hue='Attrition',data=data)

     <seaborn.axisgrid.FacetGrid at 0x7d5bb200d330>
```

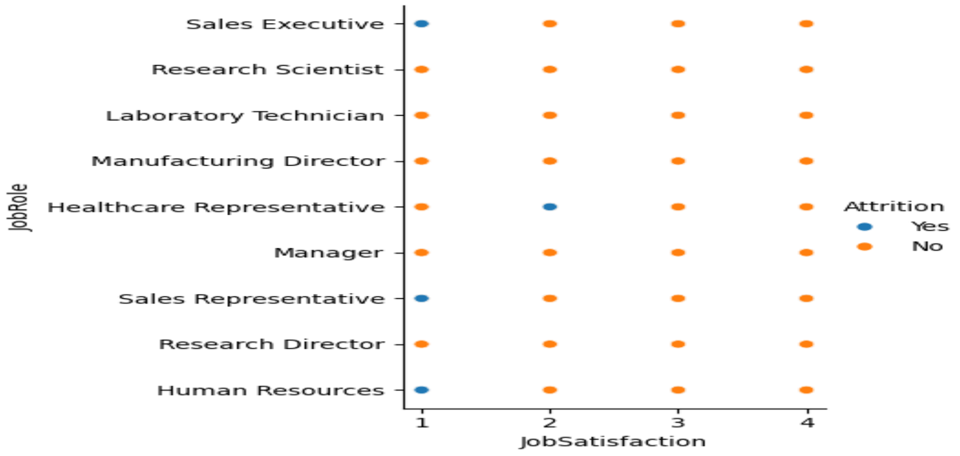


Fig6. Relational plot showing Attrition with respect to Job role and Job satisfaction.

Inference: We can clearly see in the graph that, Attrition is more in employees who are less satisfied with their jobs.

```
[ ]  snr.relplot(y='YearsAtCompany',x='MonthlyIncome',hue='Attrition',data=data)

     <seaborn.axisgrid.FacetGrid at 0x7d5bb0183340>
```

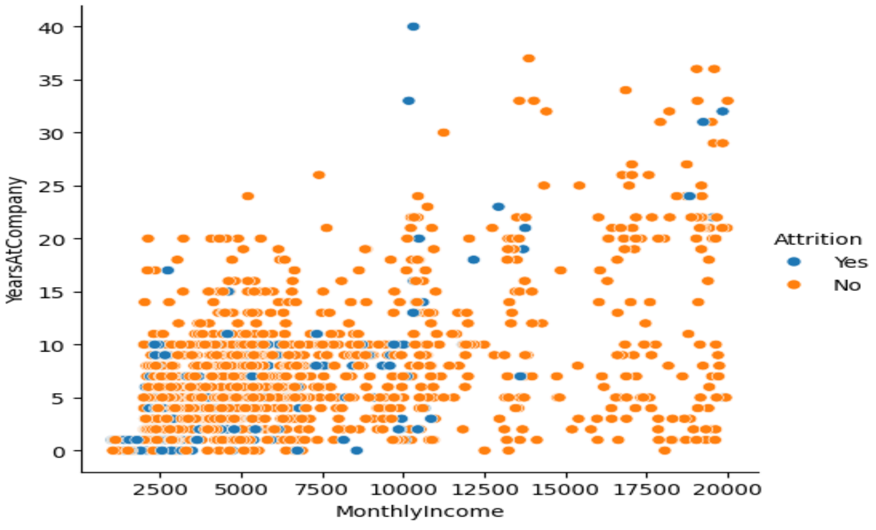


Fig7. Relational plot showing Attrition with respect to Monthly Income and Years at company.

Inference: Here is another interactive plot () that shows Attrition is more in employees with monthly income less than 14000 and who have a shorter span of years with the company.

```
snr.relplot(y='PercentSalaryHike',x='PerformanceRating',hue='Attrition',data=data)

<seaborn.axisgrid.FacetGrid at 0x7d5bb113e980>
```

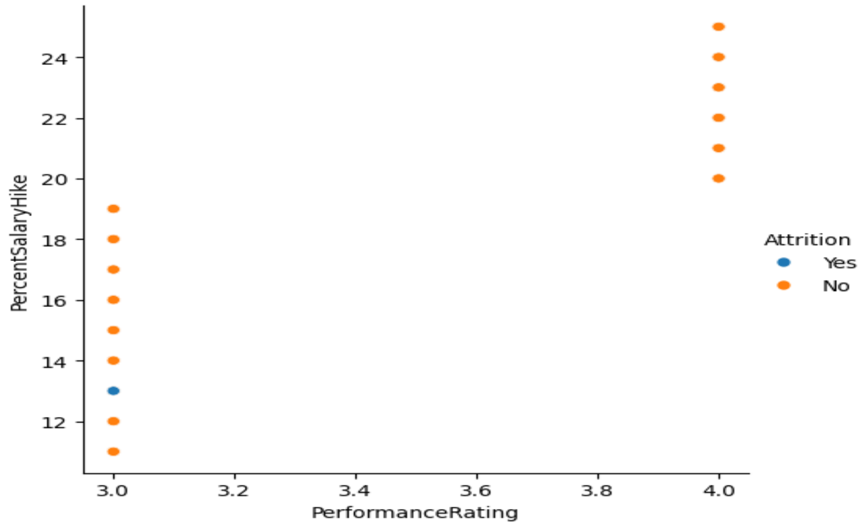


Fig8. Relational plot showing Attrition with respect to percent salary hike and performance rating.

Inference: Here in this plot (), we have plotted Percent Salary Hike vs Performance Rating and got the insight that both features have a direct relationship with each other.

```
[ ] snr.relplot(y='JobSatisfaction',x='WorkLifeBalance',hue='Gender',col='Attrition',data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d5bb1001750>
```

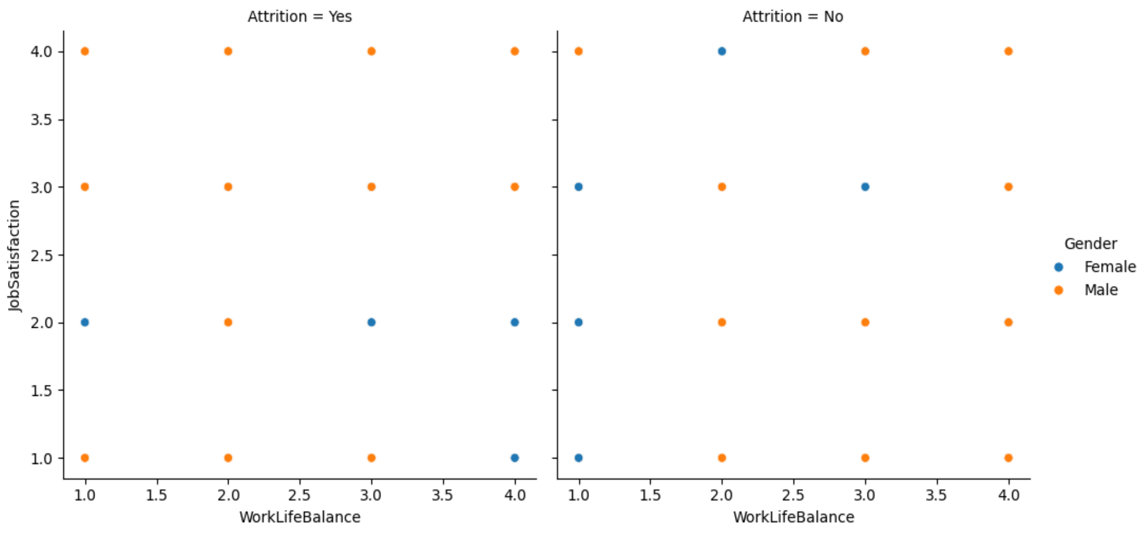


Fig9. Job Satisfaction vs Work life Balance vs Gender.

Inference: Male who are not satisfied with the job tend to have more Attrition despite of having a good work life balance.

```
[ ] snr.relplot(x='MaritalStatus',y='WorkLifeBalance',hue='Attrition',col='Gender',data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d5bb0182e60>
```

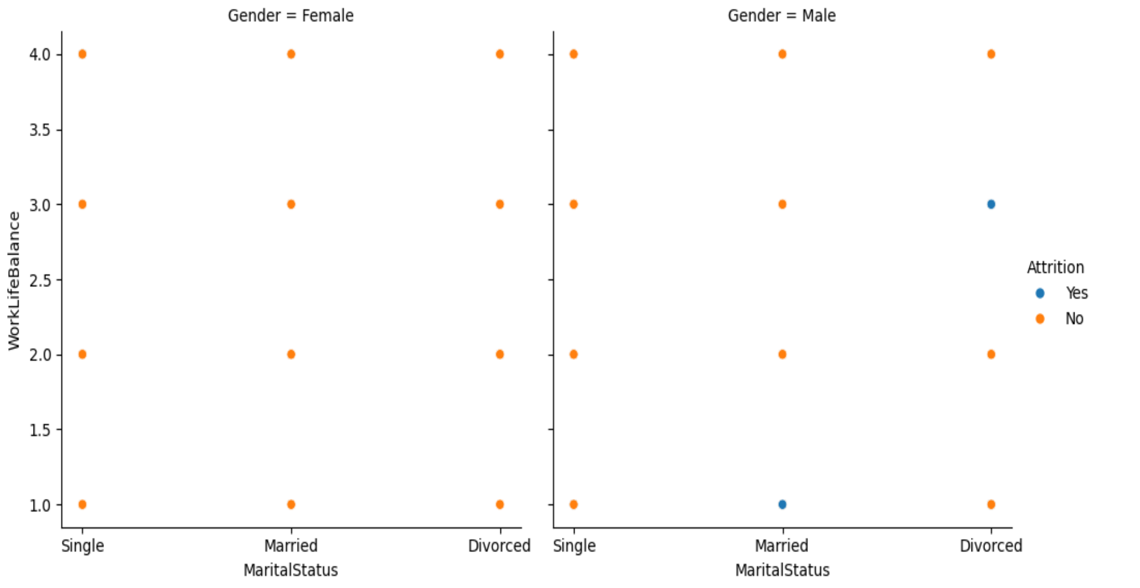


Fig9. Marital Status vs Work life Balance vs Gender.

Inference: Males who are married and divorced with having work life balance less than 3 with the job, tend to have more Attrition as compared to females.

```
[ ]  snr.relplot(y='TotalWorkingYears',x='PercentSalaryHike',hue='PerformanceRating',col='Attrition',data=data)

     <seaborn.axisgrid.FacetGrid at 0x7d5bb120f3a0>
```

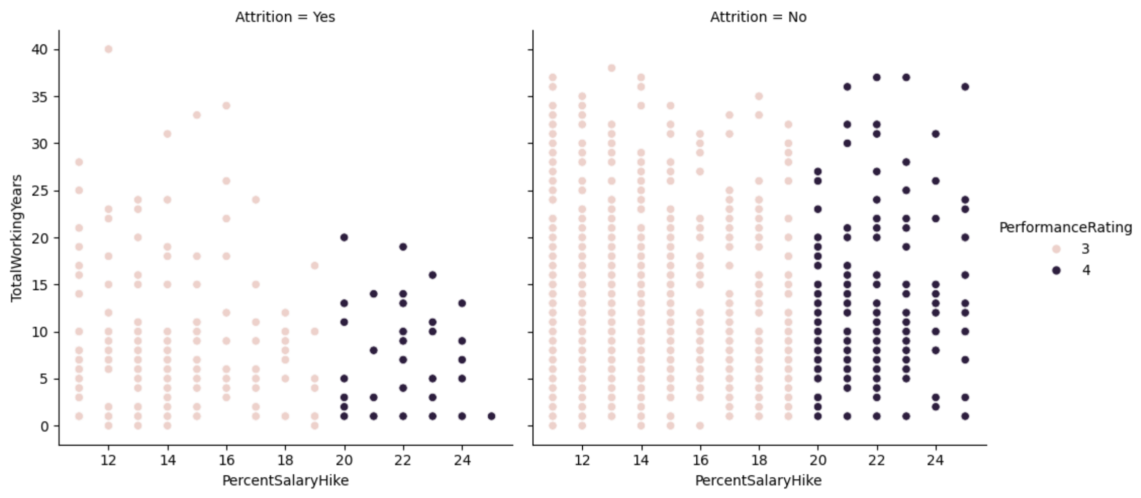


Fig9. Total Working years vs percent salary hike vs performance rating.

Inference: Employees who have performance rating of 3, having work experience of less than 10 years and salary hike less than 20% have more Attrition.

Data Cleaning:

Deleting some unwanted columns from the data.

```
[ ]  df=data.drop(['JobInvolvement','EmployeeCount','EmployeeNumber','Over18','StandardHours'],axis=1) #axis=1 will delete the entire column

     df.tail(10)
```

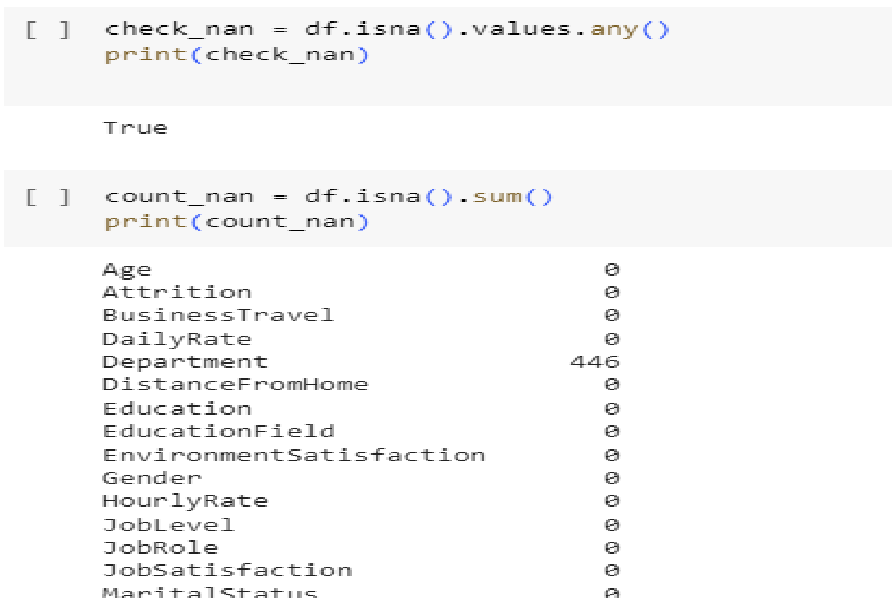
One Hot Coding to replace categorical data with numeric values.

```
df['Attrition']=df['Attrition'].apply({'Yes':1,'No':0}.get)
df['BusinessTravel']=df['BusinessTravel'].apply({'Travel_Rarely':1,'Travel_Frequently':2,'Non-Travel':0}.get)
df['Department']=df['Department'].apply({'Human Resources':1,'Research & Development':2,'Non-Sales':3}.get)
df['EducationField']=df['EducationField'].apply({'Human Resources':1,'Life Sciences':2,'Marketing':3,'Medical':4,'Other':5,'Technical Degree':6}.get)
df['Gender']=df['Gender'].apply({'Male':1,'Female':0}.get)
df['MaritalStatus']=df['MaritalStatus'].apply({'Single':2,'Married':1,'Divorced':0}.get)
df['JobRole']=df['JobRole'].apply({'Sales Executive':1,'Research Scientist':2,'Laboratory Technician':3,'Manufacturing Director':4,'Healthcare Representativ
df['OverTime']=df['OverTime'].apply({'Yes':1,'No':0}.get)
```

By using the to_numeric() function with the errors=‘coerce’ parameter, we convert the column to floats, and non-numeric values are replaced with NaN**.**

```
[ ]  df['BusinessTravel'] = pd.to_numeric(df['BusinessTravel'], errors='coerce')
     df['Department'] = pd.to_numeric(df['Department'],errors='coerce')
     df['EducationField'] = pd.to_numeric(df['EducationField'], errors='coerce')
     df['Gender'] = pd.to_numeric(df['Gender'], errors='coerce')
     df['JobRole'] = pd.to_numeric(df['JobRole'], errors='coerce')
     df['MaritalStatus'] = pd.to_numeric(df['MaritalStatus'], errors='coerce')
     df['OverTime'] = pd.to_numeric(df['OverTime'], errors='coerce')
     df['Attrition'] = pd.to_numeric(df['Attrition'], errors='coerce')
```

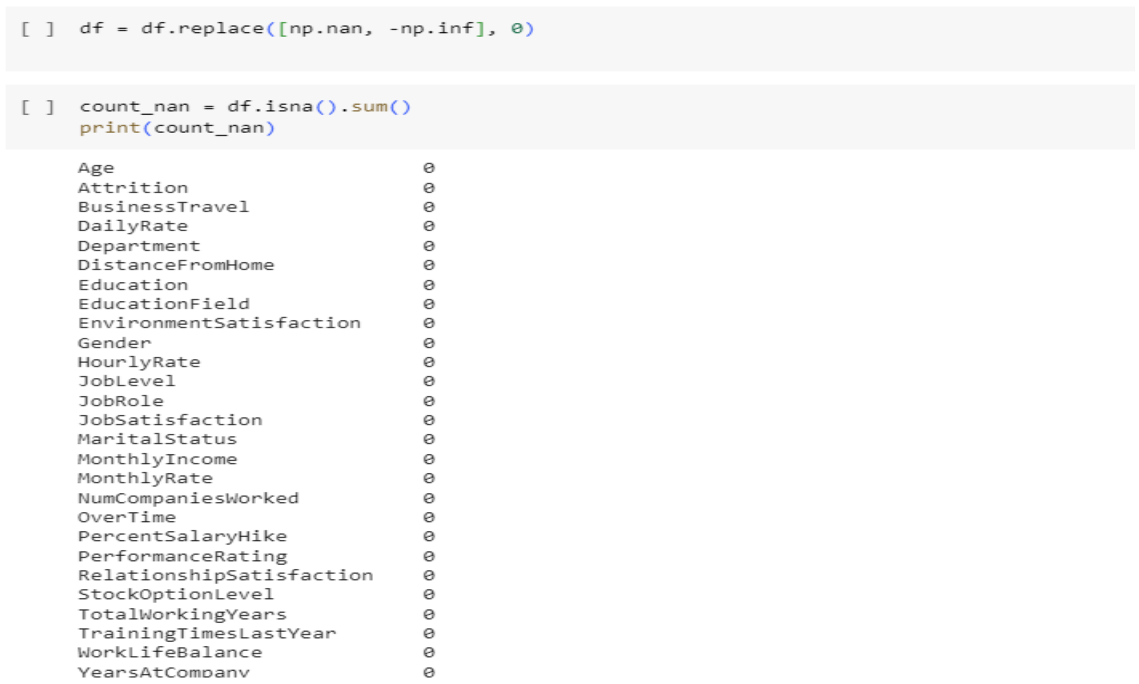
Checking for the Nan values present in the data.



```
[ ]  check_nan = df.isna().values.any()
     print(check_nan)

     True
```

```
[ ]  count_nan = df.isna().sum()
     print(count_nan)

     Age                           0
     Attrition                     0
     BusinessTravel                0
     DailyRate                     0
     Department                  446
     DistanceFromHome              0
     Education                     0
     EducationField                0
     EnvironmentSatisfaction       0
     Gender                        0
     HourlyRate                    0
     JobLevel                      0
     JobRole                       0
     JobSatisfaction               0
     MaritalStatus                 0
```

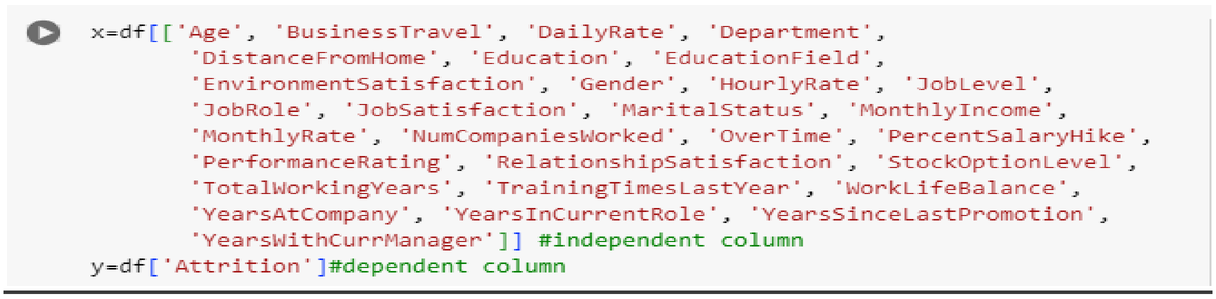
Replacing the Nan values.



```
[ ]  df = df.replace([np.nan, -np.inf], 0)
```

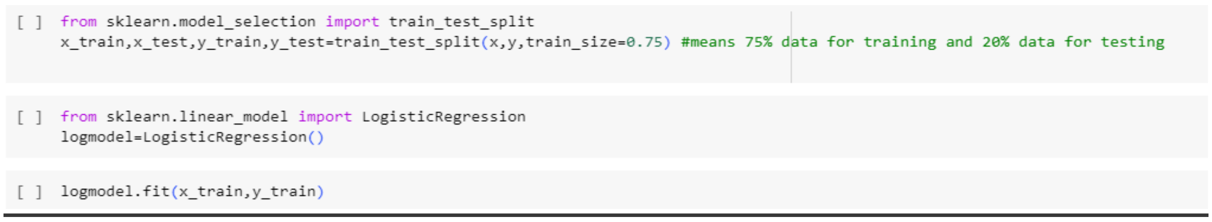
```
[ ]  count_nan = df.isna().sum()
     print(count_nan)

     Age                         0
     Attrition                   0
     BusinessTravel              0
     DailyRate                   0
     Department                  0
     DistanceFromHome            0
     Education                   0
     EducationField              0
     EnvironmentSatisfaction     0
     Gender                      0
     HourlyRate                  0
     JobLevel                    0
     JobRole                     0
     JobSatisfaction             0
     MaritalStatus               0
     MonthlyIncome               0
     MonthlyRate                 0
     NumCompaniesWorked          0
     OverTime                    0
     PercentSalaryHike           0
     PerformanceRating           0
     RelationshipSatisfaction    0
     StockOptionLevel            0
     TotalWorkingYears           0
     TrainingTimesLastYear       0
     WorkLifeBalance             0
     YearsAtCompany              0
```

Splitting Data in to Independent and Dependent Columns.

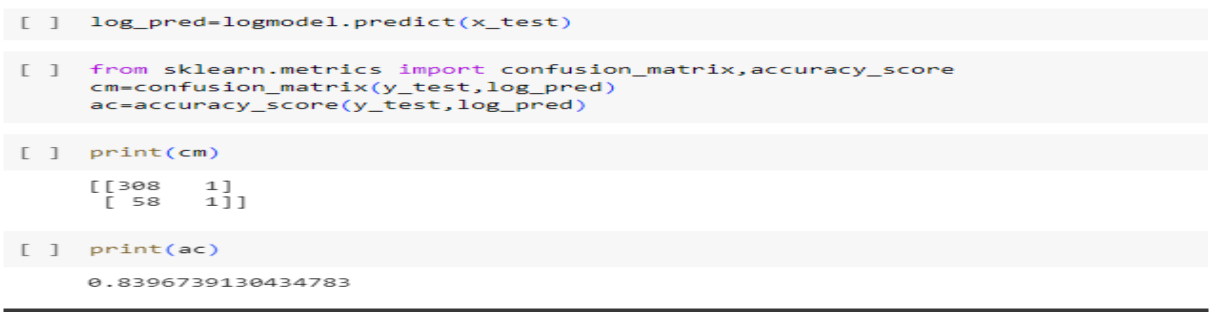


```
x=df[['Age', 'BusinessTravel', 'DailyRate', 'Department',
        'DistanceFromHome', 'Education', 'EducationField',
        'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobLevel',
        'JobRole', 'JobSatisfaction', 'MaritalStatus', 'MonthlyIncome',
        'MonthlyRate', 'NumCompaniesWorked', 'OverTime', 'PercentSalaryHike',
        'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel',
        'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
        'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
        'YearsWithCurrManager']] #independent column
y=df['Attrition']#dependent column
```

Creating Machine Learning Model with Logistic Regression Classification.



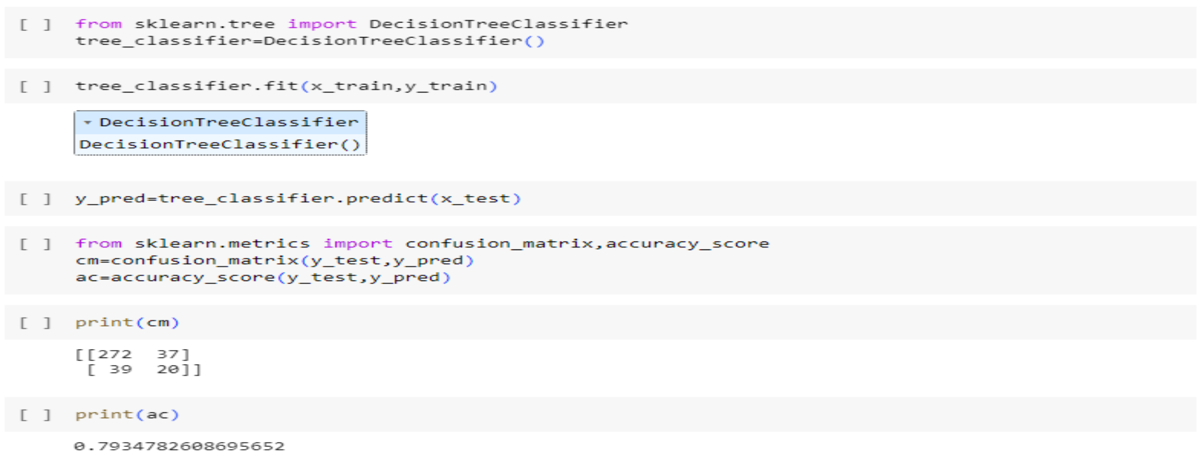
```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.75) #means 75% data for training and 20% data for testing

[ ] from sklearn.linear_model import LogisticRegression
    logmodel=LogisticRegression()

[ ] logmodel.fit(x_train,y_train)
```



```
[ ] log_pred=logmodel.predict(x_test)

[ ] from sklearn.metrics import confusion_matrix,accuracy_score
    cm=confusion_matrix(y_test,log_pred)
    ac=accuracy_score(y_test,log_pred)

[ ] print(cm)

    [[308    1]
     [ 58    1]]

[ ] print(ac)

    0.8396739130434783
```

Here we can see that Logistic Regression ML Model is giving an Accuracy of 83%.

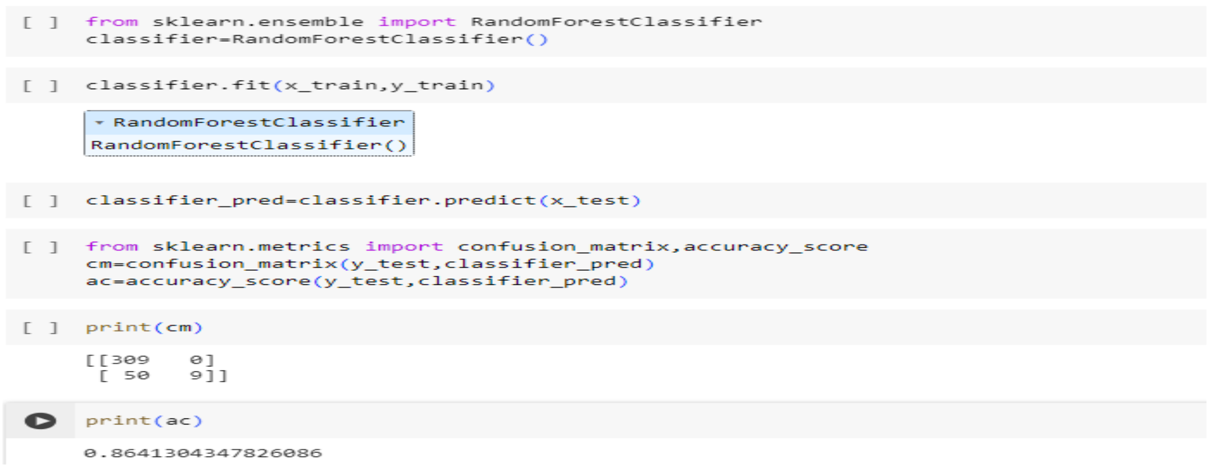
Creating Machine Learning Model with Decision Tree Regression Classification.



```
[ ] from sklearn.tree import DecisionTreeClassifier
    tree_classifier=DecisionTreeClassifier()

[ ] tree_classifier.fit(x_train,y_train)

    ▾ DecisionTreeClassifier
    DecisionTreeClassifier()

[ ] y_pred=tree_classifier.predict(x_test)

[ ] from sklearn.metrics import confusion_matrix,accuracy_score
    cm=confusion_matrix(y_test,y_pred)
    ac=accuracy_score(y_test,y_pred)

[ ] print(cm)

    [[272  37]
     [ 39  20]]

[ ] print(ac)

    0.7934782608695652
```

Decision Tree Regression ML Model is giving an Accuracy of 79%.

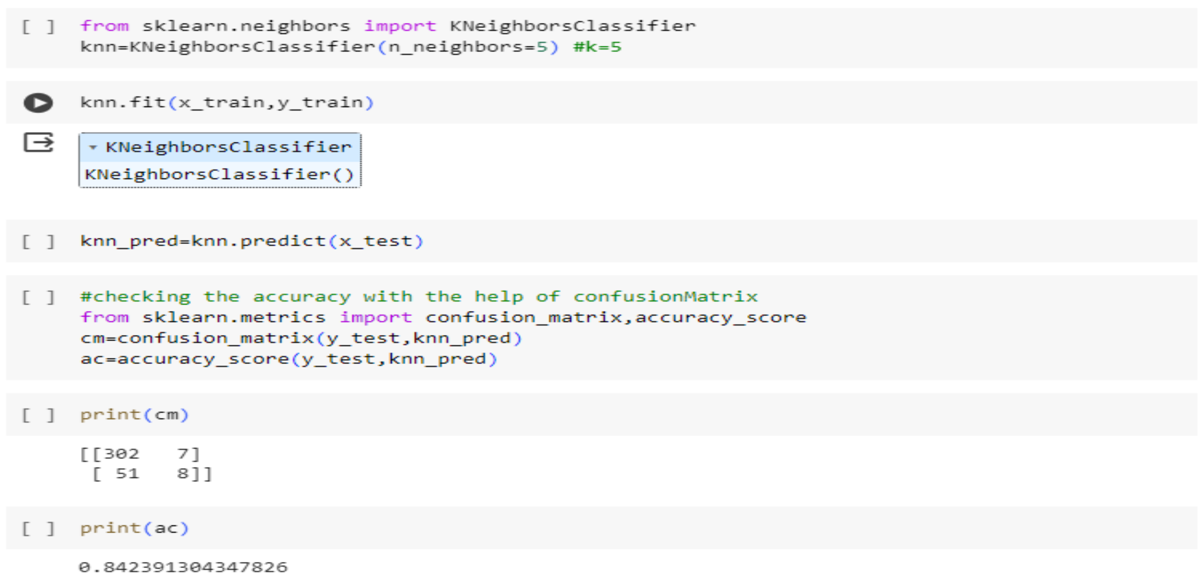
Creating Machine Learning Model with Random Forest Classifier.



```
[ ] from sklearn.ensemble import RandomForestClassifier
    classifier=RandomForestClassifier()

[ ] classifier.fit(x_train,y_train)

    ▾ RandomForestClassifier
    RandomForestClassifier()

[ ] classifier_pred=classifier.predict(x_test)

[ ] from sklearn.metrics import confusion_matrix,accuracy_score
    cm=confusion_matrix(y_test,classifier_pred)
    ac=accuracy_score(y_test,classifier_pred)

[ ] print(cm)

    [[309    0]
     [ 50    9]]

    print(ac)

    0.8641304347826086
```

Random Forest Regression ML Model is giving an Accuracy of 86%.

Creating Machine Learning Model with K Nearest Neighbor Classification.



```
[ ] from sklearn.neighbors import KNeighborsClassifier
    knn=KNeighborsClassifier(n_neighbors=5) #k=5
```

```
knn.fit(x_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

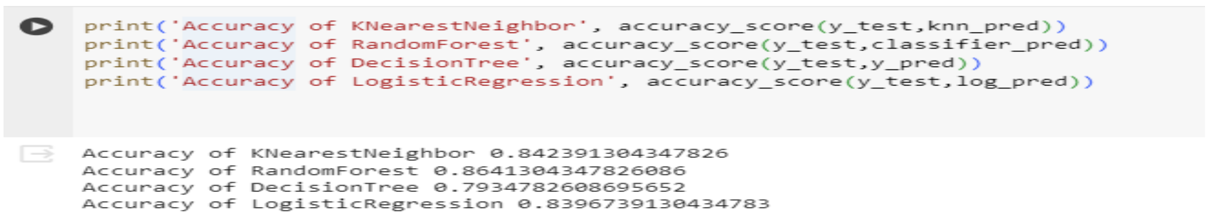
```
[ ] knn_pred=knn.predict(x_test)
```

```
[ ] #checking the accuracy with the help of confusionMatrix
    from sklearn.metrics import confusion_matrix,accuracy_score
    cm=confusion_matrix(y_test,knn_pred)
    ac=accuracy_score(y_test,knn_pred)
```

```
[ ] print(cm)

    [[302   7]
     [ 51   8]]
```

```
[ ] print(ac)

    0.842391304347826
```

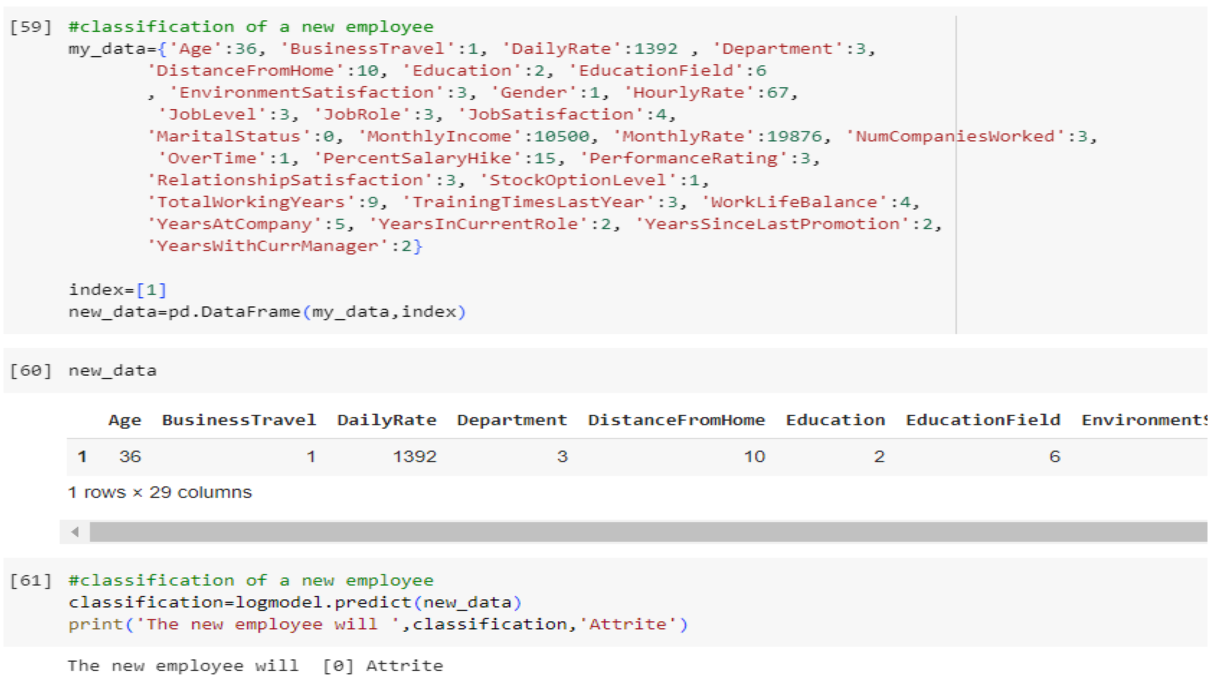
K Nearest Neighbor Regression ML Model is giving an Accuracy of 84%.



```
print('Accuracy of KNearestNeighbor', accuracy_score(y_test,knn_pred))
print('Accuracy of RandomForest', accuracy_score(y_test,classifier_pred))
print('Accuracy of DecisionTree', accuracy_score(y_test,y_pred))
print('Accuracy of LogisticRegression', accuracy_score(y_test,log_pred))

Accuracy of KNearestNeighbor 0.842391304347826
Accuracy of RandomForest 0.8641304347826086
Accuracy of DecisionTree 0.7934782608695652
Accuracy of LogisticRegression 0.8396739130434783
```

Comparing the Accuracies of all the 4 models, Random Forest Classifier is giving the best accuracy, that is of 86%.

Now we will classify a new employee and predict that the employee will Attrite or not using Random Forest classifier.



```
[59] #classification of a new employee
     my_data={'Age':36, 'BusinessTravel':1, 'DailyRate':1392 , 'Department':3,
             'DistanceFromHome':10, 'Education':2, 'EducationField':6
             , 'EnvironmentSatisfaction':3, 'Gender':1, 'HourlyRate':67,
              'JobLevel':3, 'JobRole':3, 'JobSatisfaction':4,
             'MaritalStatus':0, 'MonthlyIncome':10500, 'MonthlyRate':19876, 'NumCompaniesWorked':3,
              'OverTime':1, 'PercentSalaryHike':15, 'PerformanceRating':3,
             'RelationshipSatisfaction':3, 'StockOptionLevel':1,
             'TotalWorkingYears':9, 'TrainingTimesLastYear':3, 'WorkLifeBalance':4,
             'YearsAtCompany':5, 'YearsInCurrentRole':2, 'YearsSinceLastPromotion':2,
             'YearsWithCurrManager':2}

     index=[1]
     new_data=pd.DataFrame(my_data,index)
```

```
[60] new_data
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Environments
1	36	1	1392	3	10	2	6	

1 rows × 29 columns

```
[61] #classification of a new employee
     classification=logmodel.predict(new_data)
     print('The new employee will ',classification,'Attrite')

     The new employee will  [0] Attrite
```

Output: The new Employee will not Attrite.

Performance Analysis of Models:

Model Name	**Accuracy**
Logistic Regression	83%
Decision Tree	79%
Random Forest	86%
K Nearest Neighbor	84%

Conclusion: The main objective of this research was to predict Employee Attrition based on the Dataset provided. Here we have done various visualizations on different attributes. Based on those visualizations we analysed that the main factors for contributing to increase in the Attrition Rate are Job satisfaction and Monthly Income.

Job satisfaction is the most important factor in predicting attrition. Employees who are satisfied with their jobs are less likely to leave. Based on the data used, focusing on these top 3 features such as Employee job satisfaction, Salary hike and work life balance, the company can reduce employee attrition and improve its bottom line.

Here 4 Regression based Algorithms are used, out of which Random Forest is giving the highest accuracy which is 86%, that's why we have used the same for predicting the Employee Attrition with the new dataset.