HENRY HARVIN®
EDUCATION

Summer Internship Programme

Henry Harvin Education India LLP
Sector-2, Noida, U.P.-201306

Project Title – **HR ANALYTICS**

Mentor Name: Ms. Pooja Gupta (Designation)

Name: Abhijit Roy
Course: Summer Internship Programme (SIP) Python
Batch: June – July 2019
Job Title: Business Analyst Associate (Intern)
Institution: Lovely Professional University, Punjab

DECLARATION

I here by declare that the project report entitled of "**Car Price Prediction**" submitted by me to **HENRY HARVIN EDUCATION INDIA** is a record of bonafide project work carried out by me under the guidance of MS. POOJA GUPTA (Mentor). This project is an original report with references taken from websites and help from mentors and teachers.

Abhijit Roy

Date: 28 Jul 2019

SIP – Python

Acknowledgements

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time I am utilizing to thank all the people who have been concerned with this project. Primarily I would thank god for being able to complete this project with success. Then I would like to thank my teachers MR. DHIRAJ UPADHYAYA and MR. ANIL JADON whose valuable guidance has been the ones that helped me patch this project and make it full proof success.

Their suggestions and instructions have served as the major contributor towards the completion of the project. I would like to thank my mentor MS. POOJA GUPTA (Mentor) for giving me this golden opportunity.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance has been very helpful in various phases of the completion of the project. Last but not the least I would like to thank my batchmates who have helped me a lot.

Abhijit Roy

SIP-Python

TABLE OF CONTENTS

Abstract

"In the current highly competitive environment, talented people are definitely the most valuable assets. During last years, large investments were put into tools and information systems to manage performance, hiring, compliance and employees' development in order to enhance its capabilities and increase efficiency. Using data produced by these tools and systems typically implemented into enterprise HR departments, most companies are able to provide reports at least at some basic level. Organizations that already launched digital transformation processes do take things one step further by accompanying their reporting with basic analysis of HR metrics or Human Resource metrics".

1. Project Introduction

In a competitive market scenario, it is imperative that an employee's potential be harnessed to the best for organizational success. In such an environment, human resources remain one of the primary distinguishing factors for an organization that can be used for competitive growth in order to create necessary organizational value.

The optimum utilization of the human resources capital that an organization possesses is an on-going process; consistent efforts in the direction will ensure that the human resources of an organization would remain an asset and not a liability. Human Resource Management must be undertaken taking into consideration the needs of the organization as a whole; it can be understood as a domain of study that is oriented towards exploring those practices and approaches, which can be implemented in the context of employees to achieve organizational goals (Armstrong and Taylor, 2014). However, for a Human Resources Management to be appropriately effective and help in making alterations and introductions that yield positive results or have profitable implications, it should be oriented towards gaining a deeper insight into behavioral particularities and characteristics of its employees.

2. Project Data Introduction

This project is based on Predictive Analysis. This is a Python-based Project. This project was created via Spyder 3.3.5. IDE (Integrated Development Environment) using Python 3.7.3 and Ipython Console 7.4.0. The final outcome of this project is saved as a Jupyter Notebook v7.8.0. The libraries of python used in this project are:

1. NumPy
2. Pandas
3. Matplotlib
4. Seaborn
5. Statsmodels
6. Sci-kit Learn

This project is based on a data set provided by the teachers via GITHUB. The data used in the project is continuous, and hence, we are using "LOGISTIC REGRESSION" and "RANDOM FOREST REGRESSION" for predicting our data.

Here, the **target variable** is "LEFT".

Data Set Dictionary:

Name of Column	Description	Type
satisfaction_level	Satisfaction level of the employee	Numeric
last_evaluation	Last evaluation of the employee	Numeric
Number_projects	No of projects completed by the employee	Categorical
Average monthly hours	Average Monthly hours spent by the employee	Categorical

Time spend in company	Time spent by employee in the company	Categorical
Name of Column	**Description**	**Type**
Work accident	Accident happened to employee or not while working	Categorical
Left (target variables)	Employee left or not	Categorical
Promotion last 5 years	Promotion status in the last 5 years	Categorical
Department	Department of the employee	Categorical
Salary	Salary category	Categorical

Data Set Size: 9653 rows, 10 columns

Categorical Variables:[number_projects, work_accident, left, promotion_last5years, department, salary, time_spend_company, average_monthly_hours] = 8 features

Numeric Variables: [satisfaction_level, last_evaluation] = 2 Features

This Data Set is present in the GITHUB Repository as follows:

https://github.com/jitroy160/Final_Projects/blob/master/Final_Projects/HR_Analytics.xlsx

3. Exploratory Data Analysis (EDA)

In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by many to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

In this project, we used matplotlib, seaborn for EDA using python 3.7.3. It is as follows:

3.1. Data Understanding

At first, I imported all the libraries initially required for EDA. Then, I imported the file saved in the repository link and displayed its data. The source code and output are:

```
In [103]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import statsmodels.api as sm
          import seaborn as sns
          %matplotlib inline

In [104]: hr=pd.read_excel('C://Users//hp//Desktop//Henry Harvin//Assignment #6//HR_Analytics.xlsx')

In [105]: hr.head(5)
```

Out[105]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left
0	0.38	0.53	2	157	3	0	1
1	0.80	0.86	5	262	6	0	1
2	0.11	0.88	7	272	4	0	1
3	0.72	0.87	5	223	5	0	1
4	0.37	0.52	2	159	3	0	1

Then, I used the **columns** and **shape** and **index** function to study the summary of the data(min, max, no of values etc.) The source code and output are the following:

```
In [106]: hr.shape
Out[106]: (14999, 10)

 In [15]: hr.columns
 Out[15]: Index(['satisfaction_level', 'last_evaluation', 'number_project',
                 'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
                 'promotion_last_5years', 'department', 'salary'],
                dtype='object')

 In [16]: hr.index
 Out[16]: RangeIndex(start=0, stop=14999, step=1)
```

Then, I used the describe() function and info() function to study my data set.

```
In [108]: hr.describe()
```

Out[108]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000

```
In [109]: hr.describe(percentiles=[0.25,0.5,0.75,1]).round(2)
```

Out[109]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident
count	14999.00	14999.00	14999.00	14999.00	14999.00	14999.00
mean	0.61	0.72	3.80	201.05	3.50	0.14
std	0.25	0.17	1.23	49.94	1.46	0.35
min	0.09	0.36	2.00	96.00	2.00	0.00
25%	0.44	0.56	3.00	156.00	3.00	0.00
50%	0.64	0.72	4.00	200.00	3.00	0.00
75%	0.82	0.87	5.00	245.00	4.00	0.00
100%	1.00	1.00	7.00	310.00	10.00	1.00
max	1.00	1.00	7.00	310.00	10.00	1.00

3.2. Data Cleaning and Preparation

Data Cleaning, as the name suggests, is to clean the data of any irregularities. By performing this step, we prepare our data for analysis. For this, we check for any spelling errors, empty values and duplicate values. The source code and output are:

```
In [13]: hr.columns
         hr=hr.rename(columns={'average_montly_hours':'average_monthly_hours'}) #renaming incorrect values
         hr.columns=hr.columns.str.lower() #changing the case to lower-case

         #checking for duplicate values
         hr.loc[hr.duplicated()] #checking duplicate values
         hr.drop_duplicates(keep=False, inplace=True) #deleting duplicate values

         hr.columns
         hr.isnull().any() #Hence, no null values

Out[13]: satisfaction_level       False
         last_evaluation          False
         number_project           False
         average_monthly_hours    False
         time_spend_company       False
         work_accident            False
         left                     False
         promotion_last_5years    False
         department               False
         salary                   False
         dtype: bool
```

Now, we have the columns ‘technical’, ‘IT’, ‘support’ that mean the same. Hence, let us combine them to one column - ‘technical’. Check the values after doing it.

```
In [16]: #combining support, technical and IT into one value - technical
         hr['department']=np.where(hr['department'] =='support', 'technical', hr['department'])
         hr['department']=np.where(hr['department'] =='IT', 'technical', hr['department'])

In [17]: hr['department'].unique() #checking the unique values

Out[17]: array(['sales', 'accounting', 'hr', 'technical', 'management',
                'product_mng', 'RandD', 'marketing'], dtype=object)
```

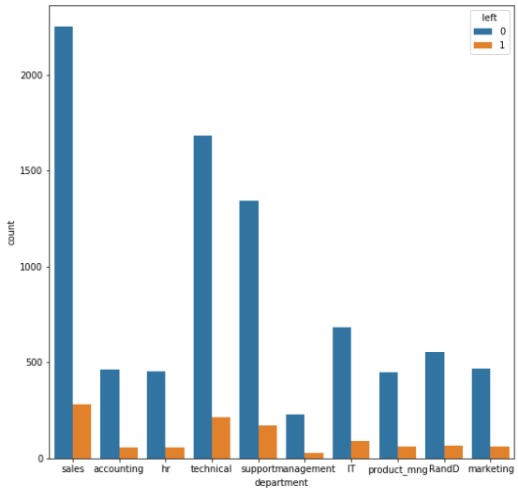
Our data is officially clean. It’s time for the final step of EDA: Visualization.

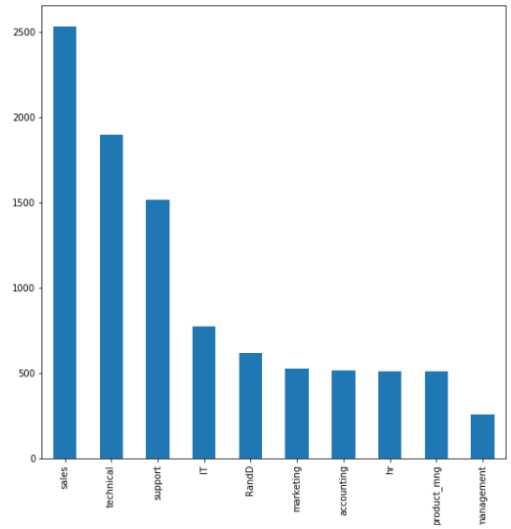
3.3. Visualization

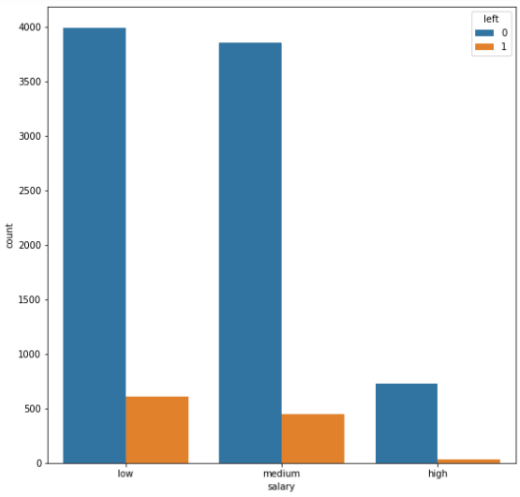
Visualization refers to the term that gives a picture to our information. We can describe our data by drawing graphs and charts to check different parameters that, in the end, might help us choose features for our analysis. In python, we use matplotlib and seaborn for visualization. These two libraries are efficient enough to give us an output that gives us an idea about our data set. The source code and output are :

In [114]:
```
#3. Visualizing the Data

plt.figure(figsize=(20,20))
plt.subplot(2,2,1)
hr.groupby(['department'])['left'].value_counts() #checking how many from each department left and didn't leave
sns.countplot(x = 'department', hue = 'left', data = hr)
plt.subplot(2,2,2)
hr['department'].value_counts().plot('bar') #people in each department
plt.subplot(2,2,3)
sns.countplot(x = 'salary', hue='left', data = hr) #salary of people leaving and not leaving
```

Out[114]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b292f8c128>
```



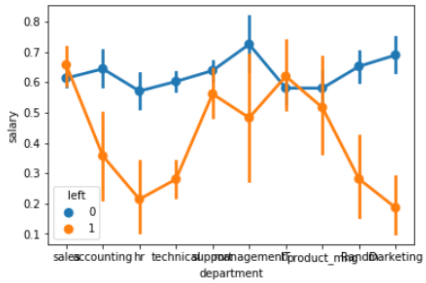




We can see that technical has most of its employees leaving. Also, low and medium salary employees are leaving more than others.

Now, lets check the salary trend by converting the salary category to codes. For that, we will use cat function.

```
In [115]: hr_data=hr.copy()
          hr_data['salary'] = hr_data['salary'].astype('category')
          hr_data['salary'] = hr_data['salary'].cat.reorder_categories(['low', 'medium', 'high'])
          hr_data['salary'] = hr_data['salary'].cat.codes

In [116]: sns.pointplot(x='department', y='salary', hue='left', data=hr_data) #trend

Out[116]: <matplotlib.axes._subplots.AxesSubplot at 0x1b29319cc18>
```



Now, we need to visualize our features to select the most significant of them for our analysis for better results. Hence, let's first visualize the categorical variables.

Categorical Variables: [number_projects, work_accident, left, promotion_last5years, department, salary, time_spend_company, average_monthly_hours] = 8 features

1. Department

First, let us see what the department category has to offer. We can check the turnover frequency for each department. Also, we can check different values for each department.

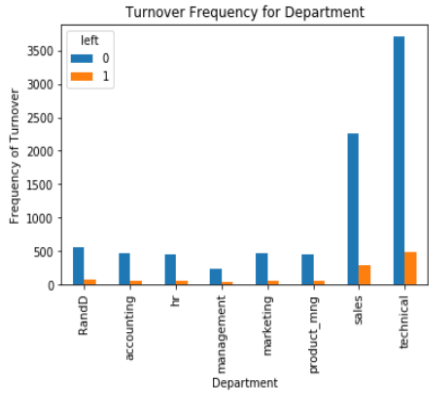
```
dep_=hr_data.groupby('department').mean()
dep_
```

```
In [19]: dep_=hr.groupby('department').mean()
         dep_

Out[19]:
```

department	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	work_accident	left	promotion_last_5years
RandD	0.631623	0.715114	3.849026	202.089286	3.274351	0.170455	0.103896	0.024351
accounting	0.629052	0.723017	3.837524	200.547389	3.249516	0.154739	0.108317	0.011605
hr	0.640335	0.722185	3.687008	200.159449	3.155512	0.141732	0.110236	0.013780
management	0.645000	0.733333	3.817829	202.856589	3.189922	0.158915	0.112403	0.015504
marketing	0.649163	0.719601	3.747148	199.188213	3.207224	0.153992	0.112167	0.020913
product_mng	0.638071	0.714331	3.787402	198.094488	3.157480	0.173228	0.118110	0.000000
sales	0.644621	0.710877	3.772512	199.416272	3.163507	0.163902	0.110585	0.007109
technical	0.644761	0.719269	3.832378	200.101242	3.198663	0.161652	0.113897	0.007402

Now, let us plot a graph for department turnover frequency.

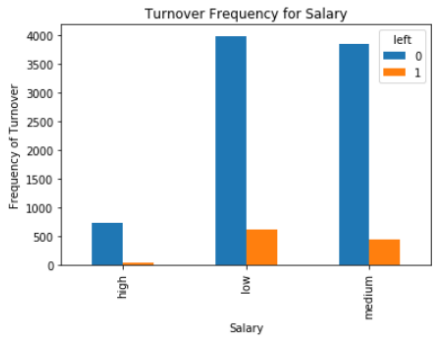
```
In [120]: #plot - employee turnover of the company
          pd.crosstab(hr['department'],hr['left']).plot(kind='bar')
          plt.title('Turnover Frequency for Department')
          plt.xlabel('Department')
          plt.ylabel('Frequency of Turnover')

Out[120]: Text(0, 0.5, 'Frequency of Turnover')
```



2. Salary

Salary is one of the most important features that decide the employee turnover of the company. Let us look at it's data.

```
In [21]: pd.crosstab(hr['salary'],hr['left']).plot(kind='bar')
         plt.title('Turnover Frequency for Salary')
         plt.xlabel('Salary')
         plt.ylabel('Frequency of Turnover')

Out[21]: Text(0, 0.5, 'Frequency of Turnover')
```



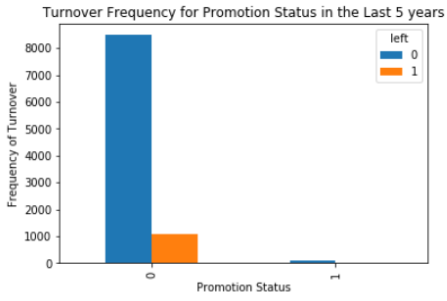
This clearly shows that medium salary and low salary people are leaving more

frequently.

3. Promotion Last 5 Years

This is a very important variable for the analysis. Promotion becomes an essential feature when it comes to employee leaving or staying in the company.

```
In [22]: pd.crosstab(hr['promotion_last_5years'],hr['left']).plot(kind='bar')
         plt.title('Turnover Frequency for Promotion Status in the Last 5 years')
         plt.xlabel('Promotion Status')
         plt.ylabel('Frequency of Turnover')

Out[22]: Text(0, 0.5, 'Frequency of Turnover')
```

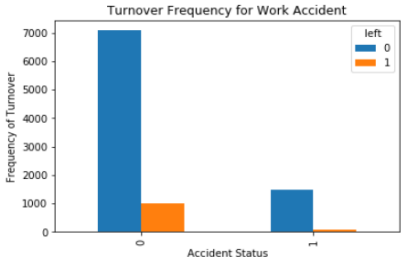


This shows that people who didn't get promotion in the last 5 years were the most to leave.

4. Work Accident

It can be another significant variable in predicting whether the employee will leave or not. Hence, let us check the employee turnover.

```
In [23]: pd.crosstab(hr['work_accident'],hr['left']).plot(kind='bar')
         plt.title('Turnover Frequency for Work Accident')
         plt.xlabel('Accident Status')
         plt.ylabel('Frequency of Turnover')

Out[23]: Text(0, 0.5, 'Frequency of Turnover')
```

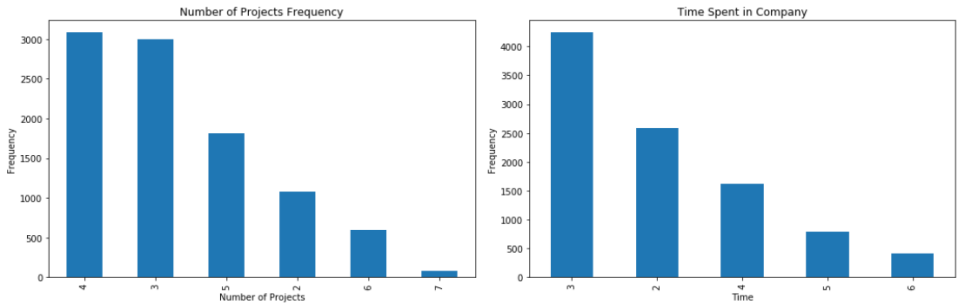


From the graph, we can say that the employees that were not involved in a work accident were leaving more frequently.

5. Number of Projects , Time Spent in Company

Number of Projects is a significant variable in the prediction of employee leaving or not. So is Time spent in company. Let us check.

```
In [24]: plt.figure(figsize=(15,5))
         plt.subplot(1,2,1)
         plt.title('Number of Projects Frequency')
         plt.xlabel('Number of Projects')
         plt.ylabel('Frequency')
         hr['number_project'].value_counts().plot(kind='bar')
         plt.subplot(1,2,2)
         plt.title('Time Spent in Company')
         plt.xlabel('Time')
         plt.ylabel('Frequency')
         hr['time_spend_company'].value_counts().plot(kind='bar')
         plt.tight_layout()
         plt.show()
```



This shows that number of projects 4 were completed most frequently. Also, most employees spent 3 hours in the company.

With this conclusion, the categorical variables visualization is complete. Now, let us make a correlation dataframe for left.

```
In [25]: corr=hr.corr().round(3).loc['left']
         corr=pd.DataFrame(corr)
         corr
         result=[]
```

```
In [26]: for i in corr['left']:
                if (i>-1 and i<-0.4): result.append('strong negative')
                elif (i>-0.4 and i<-0.2): result.append('moderate negative')
                elif (i>-0.2 and i<0): result.append('weak negative')
                elif(i>0 and i<0.2): result.append('weak positive')
                elif(i>0.2 and i<0.5): result.append('moderate positive')
                else : result.append('strong positive')
```

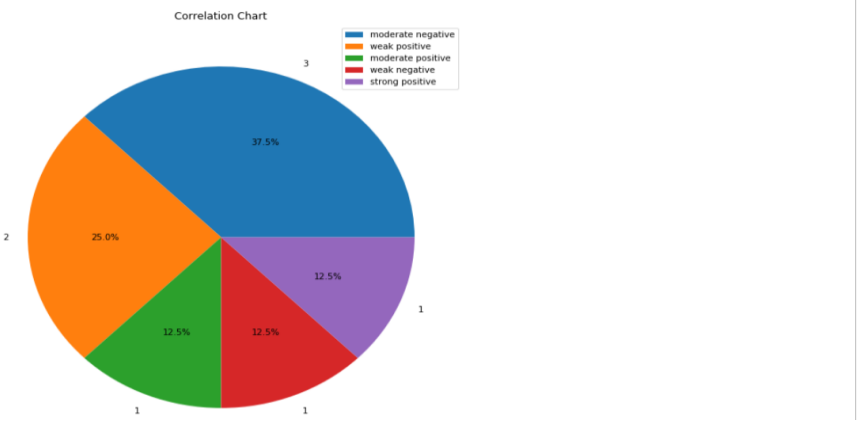
```
In [27]: corr['correlation']=result
```

Now, let us print the unique values of the correlation. Also, display the frequency for each category.

```
In [28]: print(corr['correlation'].unique())
         corr['correlation'].value_counts()

         ['moderate negative' 'weak positive' 'moderate positive' 'weak negative'
          'strong positive']

Out[28]: weak positive        3
         weak negative        2
         moderate positive    1
         moderate negative    1
         strong positive      1
         Name: correlation, dtype: int64
```

Let us create a pie chart for this correlation. It will tell us how much which category is contributing to the data.

```
In [29]: plt.figure(figsize=(10,10))
         plt.title('Correlation Chart')
         labels=corr['correlation'].unique()
         plt15 = corr['correlation'].value_counts().tolist()
         plt.pie(plt15, labels=plt15, autopct='%1.1f%%')
         plt.legend(labels, loc=1)

Out[29]: <matplotlib.legend.Legend at 0x20f1b414a20>
```



This correlation chart shows that most of the categories are moderately positively correlated with the target variable (‘left’).

After the visualization, the variables selected for analysis are:

1. Satisfaction Level
2. Time Spend Company
3. Last Evaluation
4. Number of Projects
5. Work Accident
6. Promotion last 5 years
7. Salary
8. Department

4. Model Building

Now that we have selected which features to be chosen, it is time to make a model for our analysis. First of all, what is a model?

“A statistical model is usually specified as a mathematical relationship between one or more random variables and other non-random variables. As such, a statistical model is "a formal representation of a theory.”

Now, to be precise, we need to create a model. For that, we need a data set that has only values of the features we selected through visualization. Let’s do that. The code and output are:

```
In [31]: hr_data=hr[['left','satisfaction_level','time_spend_company','last_evaluation','number_project','work_accident','promotion_last_
         hr_data=hr_data.reset_index(0)
         hr_data=hr_data.drop(columns=['index'])
         hr_data.head()
```

Out[31]:

	left	satisfaction_level	time_spend_company	last_evaluation	number_project	work_accident	promotion_last_5years	salary	department
0	1	0.36	3	0.46	2	0	0	low	sales
1	1	0.44	3	0.57	2	0	0	low	sales
2	1	0.85	5	0.99	5	0	0	low	sales
3	1	0.78	5	0.93	5	0	0	low	sales
4	1	0.39	3	0.46	2	0	0	low	sales

This is the data set, that we need to work on. Now, moving on further, we need to create dummy variables. What are dummy variables?

“A dummy variable is one that takes the value 0 or 1 to indicate the absence or presence of some categorical effect that may be expected to shift the outcome. Dummy variables are used as devices to sort data into mutually exclusive categories.”

Now, lets create these dummy variables. Remember, we need to create dummy variable for categorical variables only. Lets do this.

In [32]:
```
#Dummy Variables
def dummies(x,df):
    var=pd.get_dummies(df[x], drop_first=True)
    df=pd.concat([df,var], axis=1)
    df.drop([x], axis=1, inplace=True)
    return df
```

In [33]:
```
hr=dummies('department',hr)
hr=dummies('salary',hr)
hr=dummies('number_project',hr)
hr=dummies('promotion_last_5years',hr)
hr=dummies('work_accident',hr)
hr=dummies('time_spend_company',hr)
```

Now, lets create the target and independent variables

```
hr_var=hr.columns.tolist()
y=['left']
x=[var for var in hr_var if var not in y]
```

Now, it is time to build our model. For that, we use RFE (Recursive Feature Engineering) to select ‘n’ no. of variables from our already selected variables. Lets do that.

RFE selects ‘n’ no. of variables for your model on its own. You don’t need to select the variables. However, if you want to, you can do it by not using RFE. I have used it because it selects variables after running some tests on those variables.

Now, lets create our model.

In [38]:

```python
from sklearn.feature_selection import RFE #Recursive Feature Selection for Selecting Features
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
rfe = RFE(model, 10)
rfe = rfe.fit(hr[x], hr[y])
print(rfe.support_)
print(rfe.ranking_)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
[False False False False False False False False False False False False
  True False False  True False  True False  True  True False False False
 False  True  True  True False  True  True False False]
[ 2  3 24 23 22 18 20 17 21 19 11 12  1 10  4  1  7  1 13  1  1 16  6 14
 15  1  1  1  9  1  1  5  8]
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
In [39]: list(zip(hr[x].columns,rfe.support_,rfe.ranking_))
```

```
Out[39]: [('satisfaction_level', False, 2),
          ('last_evaluation', False, 3),
          ('average_monthly_hours', False, 24),
          ('accounting', False, 23),
          ('hr', False, 22),
          ('management', False, 18),
          ('marketing', False, 20),
          ('product_mng', False, 17),
          ('sales', False, 21),
          ('technical', False, 19),
          ('low', False, 11),
          ('medium', False, 12),
          (3, True, 1),
          (3, False, 10),
          (4, False, 4),
          (4, True, 1),
          (5, False, 7),
          (5, True, 1),
          (6, False, 13),
          (6, True, 1),
          (7, True, 1),
          (1, False, 16),
          (1, False, 6),
          (1, False, 14),
          (1, False, 15),
          (3, True, 1),
          (3, True, 1),
          (4, True, 1),
          (4, False, 9),
          (5, True, 1),
          (5, True, 1),
          (6, False, 5),
          (6, False, 8)]
```

```
In [40]: num_vars=hr[x].columns[rfe.support_] #selected features
         num_vars
```

```
Out[40]: Index([3, 4, 5, 6, 7, 3, 3, 4, 5, 5], dtype='object')
```

```
In [41]: x=hr[num_vars]
         y=hr['left']
```

The rfe.support_ function gives the value ‘True’ to each feature selected. The rfe.ranking_ gives the value 1 to each feature selected. Then, we are overwriting the response variable(s), i.e. x to the dataframe with the features selected and the target variable i.e. y to the left column of the dataframe.

Now comes the most important part. We will be splitting our data set into training set and test set. What are these sets? Why do we do this? Split the data set?

First of all, if you run the model on the whole data set and predict from the same, you will get accuracy too high, which would be invalid because your dependent variable will be included in the data set in which you are predicting your values.

Secondly, if the model fails, the data set has to be re-loaded from the beginning. Hence, we first train our model with the training set, and when our model runs perfectly, we use it on our test set to predict values. Remember, training set should always be greater than test set. The more you train your model, the better it will predict. Let’s do this…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
```

Here, we will use logistic regression to predict values. Why logistic regression? What is logistic regression?

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

We are using logistic regression because our target variable is categorical or binary (0,1) and is not continuous.

5. Prediction and Evaluation

We are at the final stage of our project. Time to predict values!

```
In [44]: y_pred=logreg.predict(x_test)
         from sklearn.metrics import accuracy_score
         print('Logistic regression accuracy: ',(accuracy_score(y_test,y_pred )*100).round(3),'%') #accuracy score of the model

         Logistic regression accuracy:  88.607 %
```

Our model accuracy is 88.607%. It's perfect for logistic regression. Now, let us check the actual and predicted values.

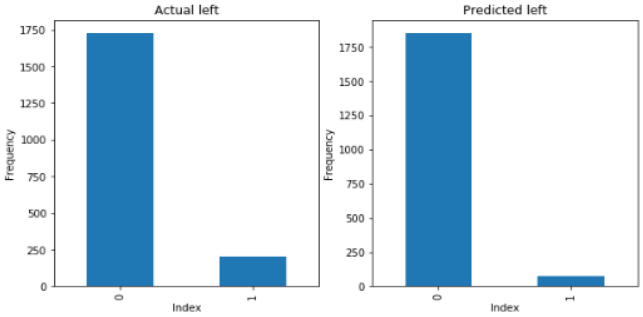
```
In [47]: y_pred=pd.DataFrame(y_pred)

         y_test=pd.DataFrame(y_test)
         y_test=y_test.reset_index(drop=True)

         p=pd.concat([y_test, y_pred], axis=1)
         p=p.rename(columns={0:'pred_left'})
         p.head()
```

Out[47]:

	left	pred_left
0	1	0
1	0	0
2	0	0
3	0	0
4	0	0

Time to plot this. Remember, these two need to be plot seperately. Let us try this and check how accurate our accuracy is.

```
In [48]: plt.figure(figsize=(10,5))
         plt.subplot(1,2,1)
         plt.title('Actual left')
         plt.xlabel('Index')
         plt.ylabel('Frequency')
         p['left'].value_counts().plot('bar')
         plt.subplot(1,2,2)
         plt.title('Predicted left')
         plt.xlabel('Index')
         plt.ylabel('Frequency')
         p['pred_left'].value_counts().plot('bar')
         plt.show()
```



Let us check the random forest classifier for this model. What is Random Forest?

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Now, let us use Random Forest Classifier for our analysis and predict values using random forest classifier and check the accuracy score.

The code and output are:

```
In [49]: #Random forest
         from sklearn.ensemble import RandomForestClassifier
         rf=RandomForestClassifier()
         rf.fit(x_train,y_train)

         C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators wil
         l change from 10 in version 0.20 to 100 in 0.22.
           "10 in version 0.20 to 100 in 0.22.", FutureWarning)

Out[49]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)

In [50]: rf_pred=pd.DataFrame(rf.predict(x_test))
         acc=accuracy_score(y_test,rf_pred)*100

In [51]: print('Random Forest accuracy: ',acc.round(3),'%') #accuracy score of the model

         Random Forest accuracy:  92.025 %

In [52]: rf_pred[0].value_counts()
         rf_pred.index

Out[52]: RangeIndex(start=0, stop=1931, step=1)
```

The random forest accuracy is 92.025%. This certainly means that random forest is a better method for this data. Let us validate our analysis using confusion matrix and k-fold cross validation.

Confusion Matrix: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Now, let us create a confusion matrix for the same.

```
In [53]: #validation techniques - confusion matrix for logistic regression

         from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test,y_pred)
         cm

Out[53]: array([[1682,   48],
                [ 172,   29]], dtype=int64)

In [54]: cm_rf=confusion_matrix(y_test,rf_pred)
         cm_rf

Out[54]: array([[1625,  105],
                [  49,  152]], dtype=int64)
```

K-Fold Cross Validation: Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model

5. Summarize the skill of the model using the sample of model evaluation scores

Now, let us use K-fold Cross-Validation to check overfitting of model. The code and output are:

```
In [56]: #validation techniques - K-fold for random forest
         from sklearn import model_selection
         from sklearn.model_selection import cross_val_score,KFold
         kfold = KFold(n_splits=15, random_state=100)
         modelCV = RandomForestClassifier()
         scoring = 'accuracy'
         results = model_selection.cross_val_score(modelCV, x_test, y_test, cv=kfold, scoring=scoring)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators wil
l change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:528: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

```
In [146]: print("15-fold cross validation average accuracy: ",((results.mean())*100).round(3),'%')

          15-fold cross validation average accuracy:  97.619 %
```

We can see that our cross validation accuracy is **97.619%**, which is almost as much as our random forest accuracy. Hence, we can say that there is negligible overfitting of our model.

Now, let us see a classification report to check the generalization of our data. This will tell if our analysis is generalized to the whole data set or not.

```
In [58]: #classification report to check generalisation of data
         from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred)) #logistic regression report
```

	precision	recall	f1-score	support
0	0.91	0.97	0.94	1730
1	0.38	0.14	0.21	201
micro avg	0.89	0.89	0.89	1931
macro avg	0.64	0.56	0.57	1931
weighted avg	0.85	0.89	0.86	1931

```
In [59]: from sklearn.metrics import classification_report
         print(classification_report(y_test, rf_pred)) #random forest report
```

	precision	recall	f1-score	support
0	0.97	0.94	0.95	1730
1	0.59	0.76	0.66	201
micro avg	0.92	0.92	0.92	1931
macro avg	0.78	0.85	0.81	1931
weighted avg	0.93	0.92	0.92	1931

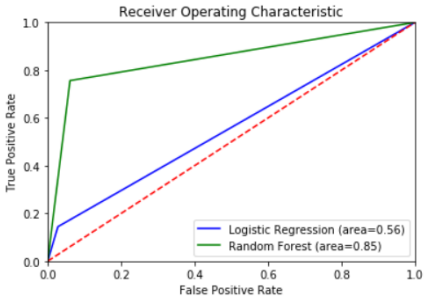
According to our report, the recall value for logistic regression is 0.97, which means it is more generalized than not, whereas the recall value for random forest is 0.94 which means it is 94% generalized. Hence, we can conclude that both our models are pretty generalized.

Let us check the predicted probability for both the cases- 0 for did not leave and 1 for left. The code and output are:

```
In [60]: a=rf.predict_proba(x_test) #predicted probability
         a=pd.DataFrame(a)

In [64]: a.loc[:,0].head(5) #probability for 0 - didn't leave

Out[64]: 0    0.345931
         1    0.990875
         2    0.996339
         3    0.352228
         4    0.968249
         Name: 0, dtype: float64

In [65]: a.loc[:,1].head(5) #probability for 1 - left

Out[65]: 0    0.654069
         1    0.009125
         2    0.003661
         3    0.647772
         4    0.031751
         Name: 1, dtype: float64
```

Now, let us make an ROC (Receiver Operating Characteristic) Curve to understand which method is better for this data set – logistic regression or random forest.

```
In [66]: #roc curve
         from sklearn.metrics import roc_curve, roc_auc_score
         fpr,tpr,threshold=roc_curve(y_test,y_pred)
         fpr_rf,tpr_rf,threshold_rf=roc_curve(y_test,rf_pred)
         roc_auc=roc_auc_score(y_test,y_pred)
         roc_auc_rf=roc_auc_score(y_test,rf_pred)

In [67]: plt.title('Receiver Operating Characteristic')
         plt.plot(fpr,tpr,'b',label='Logistic Regression (area=%0.2f)'%roc_auc)
         plt.plot(fpr_rf,tpr_rf,'g',label='Random Forest (area=%0.2f)'%roc_auc_rf)
         plt.legend(loc='lower right')
         plt.plot([0,1],[0,1],'r--')
         plt.xlim([0,1])
         plt.ylim([0,1])
         plt.ylabel('True Positive Rate')
         plt.xlabel('False Positive Rate')
         plt.show()
```



Here, we can infer that Random Forest is better than Logistic Regression in this case.

References :

- https://www.academia.edu/39249197/HR_ANALYTICS_A_MODERN_TOOL_IN_HR_FOR_PREDICTIVE_DECISION_MAKING
- https://github.com/akjadon/Finalprojects_DS/tree/master/HR_Analytics
- https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
- https://machinelearningmastery.com/k-fold-cross-validation/
- https://www.cgi.com/sites/default/files/hr_whitepaper.pdf
- https://en.wikipedia.org/wiki/Logistic_regression

------This Project is available to be seen on GITHUB follow the below link------

⇩ ⇩

https://github.com/jitroy160/Final_Projects/blob/master/Final_Projects/Final_Project_CarPrice_Prediction.ipynb