Project Title – <u>HR_Analytics</u>

Name: Nikhil Chaudhary
Course: Summer Internship Programme (SIP) Python
Batch: Jun-Jul 2019
Job: Business Analyst Associate (Intern)
Institution: JIMS Engineering Management Technical Campus,Greater Noida.

Submitted to : Mr.Anil Jadon

HR_Analytics

Aim : An employee is an asset to the company. They define the future and present of the company. So, it is obvious that a company invest a huge attention, money and care for its employees to make them not leave. People Analytics is simply the way of giving answer to why employees leave the employers through the data. In this project we try to predict whether on the present conditions if an employee leaves the company or not.

We have given a dataset of IBM 's employee. They contain data of around 5K employees. Based on their behaviour we will have to make a model for prediction in future to stop churning of employees to decrease the chances of the company's losing its profits.

The Dataset contains many independent variables for univariate and Bi-variate Analysis like 'satisfaction_level', 'last_evaluation','number_project','average_montly_hours', 'time_spend_company', 'Work_accident', 'promotion_last_5years', 'department', 'salary'.

We will have to predict the left columns so our Dependent variable will be ‘left’.

Code Analysis :

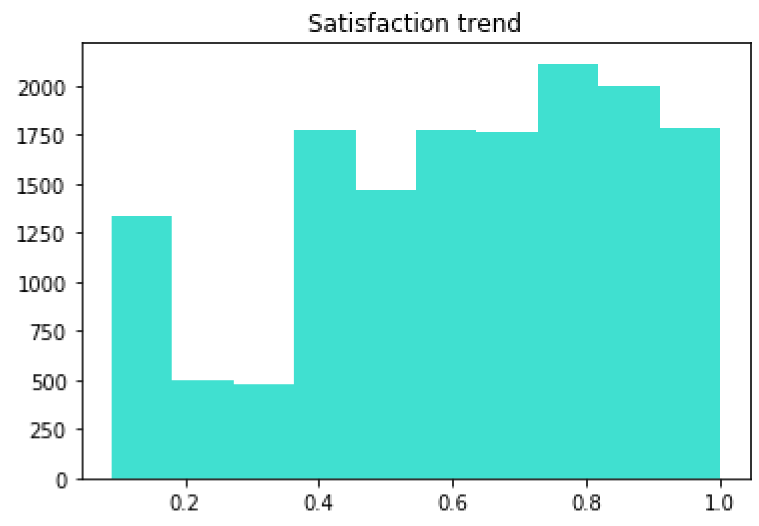
```
#importing necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#reading the excel file and loading the data in hr_data

hr_data = pd.read_excel('HR_data.xlsx')

#viewing the data

hr_data.head()

#viewing the correlation between the columns of the
dataframe

hr_data.corr()

#to view hot_map of corr using seaborn library

matrix = hr_data.corr()
f, ax = plt.subplots(figsize = (10, 10))
```

sns.heatmap(matrix, square = True, center = 0, annot = True, cmap="YlGnBu")

#to get the idea of the datatypes of the columns and their null values :
hr_data.info()

#plotting graph:

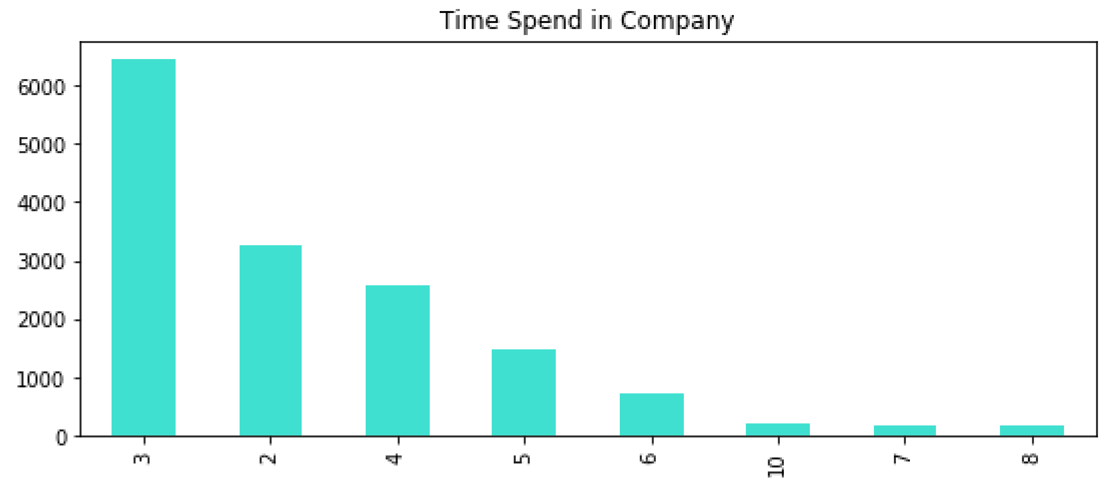
# histogram plot of satisfaction level in the hr_data
1.) plt.subplot(111)
plt.title('Satisfaction trend')
plt.hist(hr_data['satisfaction_level'],color='Turquoise')



#Bar plot of the time_spend_company's value count

2.) plt.subplot(221)

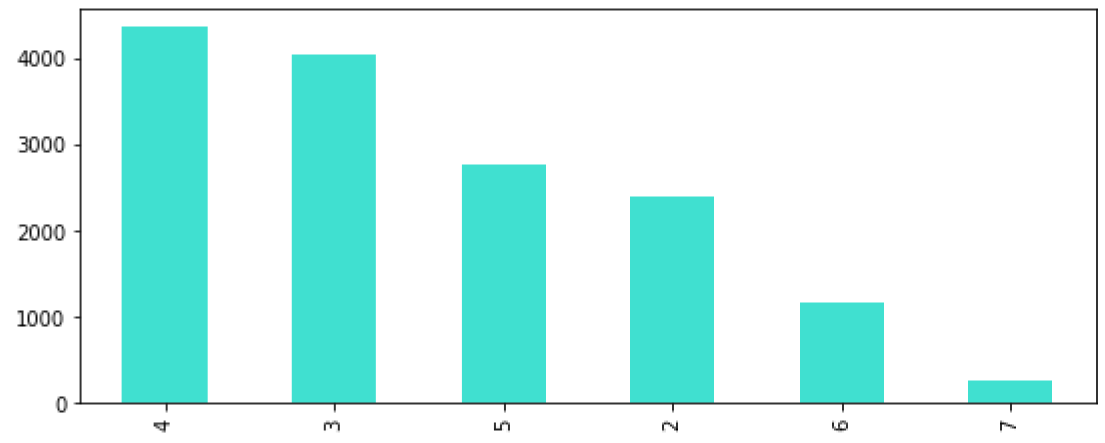
hr_data['time_spend_company'].value_counts().plot(kind = 'bar', figsize = (20,8), title = 'Time Spend in Company',color='Turquoise')



#Bar plot of the number_project's value_Count:

3.) plt.subplot(222)

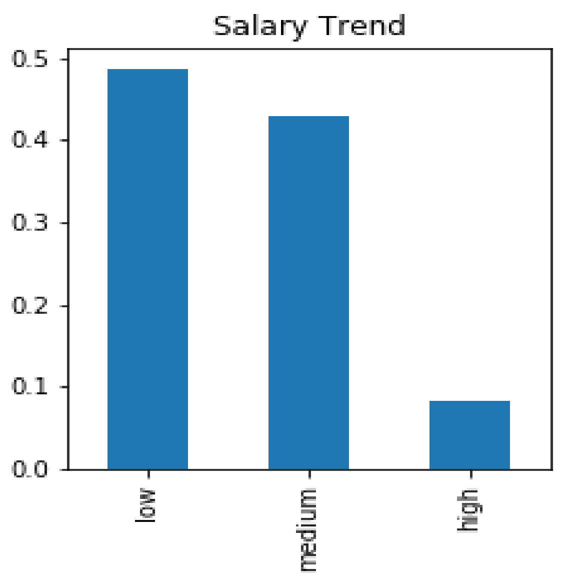
hr_data['number_project'].value_counts().plot(kind = 'bar', figsize = (20,8), title = 'Number of Projects',color = 'Turquoise')



#Bar plot of salary trends value_counts

4.)plt.subplot(259)

hr_data['salary'].value_counts(normalize = True).plot(kind = 'bar', figsize = (20,8), title = 'Salary Trend')



#as we know the columns contains two categorical column, so for modeling we need to convert them into their int64 form

#checking unique value in department column and salary column

```
hr_data.department.unique()
hr_data.salary.unique()
```

#now label encoding the two category columns to convert them into int type:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(hr_data['department'])
x=le.transform(hr_data['department'])
hr_data['department'] = x


le = preprocessing.LabelEncoder()
le.fit(hr_data['salary'])
y = le.transform(hr_data['salary'])
hr_data['salary'] = y
```

```
#checking the columns datatype for the final confirmation
hr_data.info()

#printing the unique int values of the category column
print(hr_data.department.unique())
print(hr_data.salary.unique())

#viewing the dataset and columns
hr_data.head()
hr_data.columns

#now selecting the independent variables for model
x =
hr_data[['satisfaction_level','Work_accident',‘promotion_last_5years', 'department', 'salary']]

#now selecting the dependent variable
y = hr_data['left']

#importing train_test_split library for making training and testing models:

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size =
0.3,random_state=0)
```

#now importing the LogisticRegression for implying it:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 0)
model.fit(x_train,y_train) # fitting the model
```

#checking the score of the test and the train dataset:

```
model.score(x_train,y_train)
model.score(x_test,y_test)
```

#since the logistic regression’s result is not more predictive therefore importing the RandomForestClassifier:

```
from sklearn.ensemble import RandomForestClassifier
model_random = RandomForestClassifier(n_estimators=6,
random_state=0,max_depth = 15)
```

#fitting and getting the accuracy score :

```
model_random.fit(x_train,y_train)
model_random.score(x_train,y_train)

model_random.score(x_test,y_test)

ypred = model_random.predict(x_test)
```

#Now making Confusion_matrix:

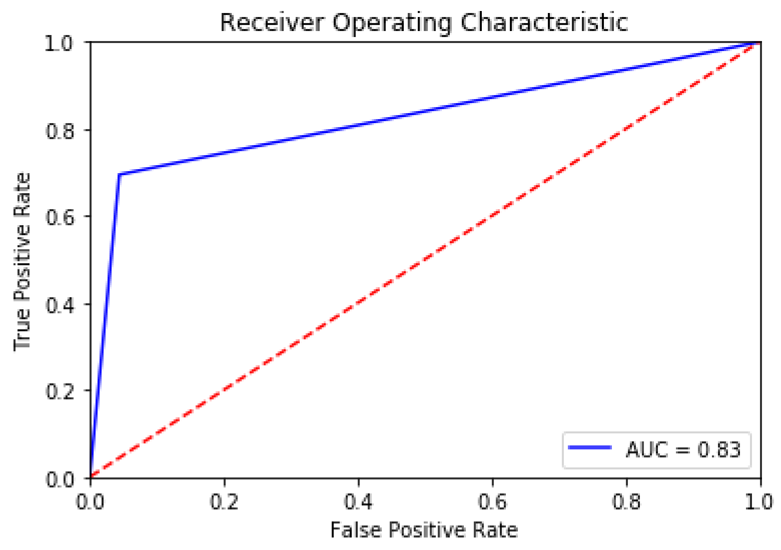
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,ypred)
cm
```

#ROC curve

```
import sklearn.metrics as metrics
```

# calculate the fpr and tpr for all thresholds of the classification:

```
Fpr = False Positive Rate
Tpr = True Positive Rate
fpr, tpr, threshold = metrics.roc_curve(y_test, ypred)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
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()
```



Hence, We have the required model for HR_Analytics.