HR DATA ANALYSIS

Introduction to Data Mining CS-536
Dr. Asim Kareem

GROUP - 14

Muhammad Matloob Altaf	21100164
Muhammad Waseem	21100327
Zahaan Wasif	21100213

Date: May 20, 2020

Contents

Libraries and Data Reading

Following libraries are used to get refined data and this will help us in making graphs and patterns from our dataset. Libraries help us in converting data in meaningful patterns and information which is the core objective of data mining.

```
import pandas as pd
import numpy as np
import pydotplus
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import seaborn as sns
%matplotlib inline
from scipy.stats import zscore
from io import StringIO
from efficient_apriori import apriori
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn import preprocessing as spp
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report,roc_curve,auc
from sklearn.tree import export_graphviz
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from IPython.display import Image
```

Figure 1.1

Now let's discuss the individual working and use of each of these libraries to understand in better way.

Numpy

Numpy helps in doing mathematical and scientific operations and is used extensively to work with multi-dimensional arrays and matrices. Numpy has not been used directly in the project but aids use of further libraries

Pandas

Pandas are used with Numpy library for fast numeric array computations. Its main functionality in our project is to allow the use of dataset attribute values to represent dataset as a Numpy array.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.

Seaborn

Seaborn gives high level interface to draw statistical graphics. It is a complete package for data visualization.

Pydotplus

PyDotPlus is an improved version of the old pydot project that provides a Python Interface to Graphviz's Dot language

Input Dataset:

```
dataset = pd.read_csv('HR_comma_sep.csv',
                      sep=',')
data = dataset.copy(deep=True)
```

Figure 1.2

Above figure shows the command that is used to load the dataset of HR given to us. We are using pandas to load the data into variable dataset and then copying the data into another named data. It is a deep copy so if we can use the data later in assignment.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   promotion_last_5years  14999 non-null  int64
 7   departmennt            14999 non-null  object
 8   salary                 14999 non-null  object
 9   left                   14999 non-null  int64
dtypes: float64(2), int64(6), object(2)
memory usage: 1.0+ MB
```

Figure 1.3

Figure 1.3 shows the column names. Total values and their data types.

Figures 1.4 shows us 1st ten elements from each column of dataset so we can see that the data is loading correctly and the information about whole dataset i.e. the total memory, the number of entries in each column, number of columns, names of columns, and the data type of entries stored in each column.

```
data.head(10)
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years	departmennt	salary
0	0.38	0.53	2	157	3	0	0	sales	low
1	0.80	0.86	5	262	6	0	0	sales	medium
2	0.11	0.88	7	272	4	0	0	sales	medium
3	0.72	0.87	5	223	5	0	0	sales	low
4	0.37	0.52	2	159	3	0	0	sales	low
5	0.41	0.50	2	153	3	0	0	sales	low
6	0.10	0.77	6	247	4	0	0	sales	low
7	0.92	0.85	5	259	5	0	0	sales	low
8	0.89	1.00	5	224	5	0	0	sales	low
9	0.42	0.53	2	142	3	0	0	sales	low

Figure 1.4

In figure 1.5, we describe our data set, which gives us count, mean. Standard deviation, min, max etc. of each feature.

```
data.describe()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years	left	s
count	14999.000000	14999.000000	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	0.021268	0.238083	
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.144281	0.425924	
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000	
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000	
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000	
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000	

Figure 1.5

Dealing with Inconsistencies:

Now the code in following figure 1.6 show how did we remove inconsistencies from the dataset. The department column was misspelled, and we updated the case sensitivity of word as it can be seen from the following figure.

```
#The column "department" is named as "departmennt". so lets rename it to "department".
data.rename(columns={'departmennt': 'department'}, inplace=True)

#Salary has two different names for same category like "low" and "Low", making them same
def salary(row):
    if row['salary'] == 'High' or row['salary'] == 'high':
        return 'high'
    elif row['salary'] == 'Medium' or row['salary'] == 'medium':
        return 'medium'
    else:
        return 'low'
data['salary'] = data.apply(salary,axis=1)
```

Figure 1.6

Figure 1.6 above shows how we used different filling techniques for missing values in each column. Mode fill is to fill the missing values so that the data could not be skewed and we also used backed filled values in those columns where we could not able to predict data for example number of work accidents, we can't say someone had a work accident or how many work accidents someone had based on other workers.

In figure 1.7, we are converting salary to numerical data for better understanding it's relationship with other factors.

```
def salary(row):
    if row['salary'] == 'high':
        return 3
    elif row['salary'] == 'medium':
        return 2
    else:
        return 1

data['salary_numerical'] = data.apply(salary, axis=1)
```

Figure 1.7

Exploratory Data Analysis (EDA):

Normalization for Exploratory Data

Here we normalize the data for better visualization. We normalize the data on mean so we can get useful patterns.

```
#normalizig for better visualization

normalized_average_montly_hours = data['average_montly_hours']/data['average_montly_hours'].mean()
normalized_average_montly_hours = data['average_montly_hours']/data['average_montly_hours'].mean()
normalized_time_spend_company = data['time_spend_company']/data['time_spend_company'].mean()
normalized_number_project = data['number_project']/data['number_project'].mean()
```

Figure 1.8

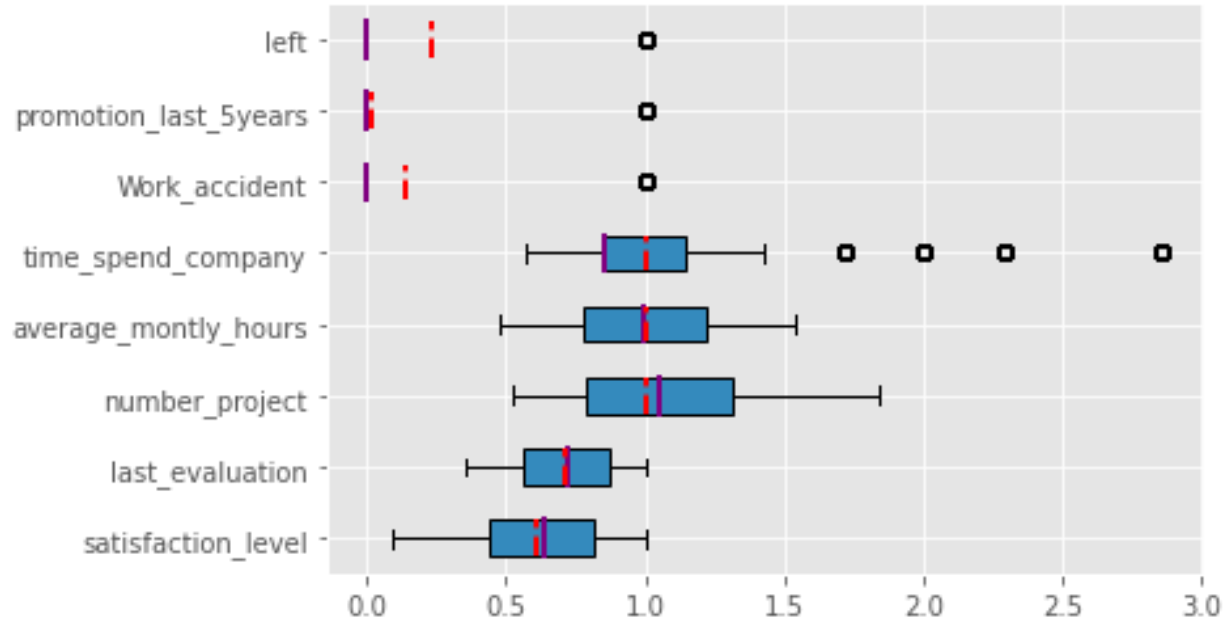

Figure 1.9

In first box the relevant columns are normalized and stored in appropriate variables and then these variables are used in next code where we are using plt.subplots and getting the axis and figure. The above code will generate the following graph.

Correlation in dataset:

We are finding correlation in the dataset using seaborn library and with built in function cor(). To obtain the pattern of this correlation a NumPy array of zeros of dimension corr is created with

data type bool. We are storing the into mask then and making another array by using triu_indices which gives us the indices for lower-triangle array. A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors. The seaborn python package allows the creation of annotated heatmaps which can be tweaked using Matplotlib tools as per our requirement.
The matrix of colors achieved can be viewed below.

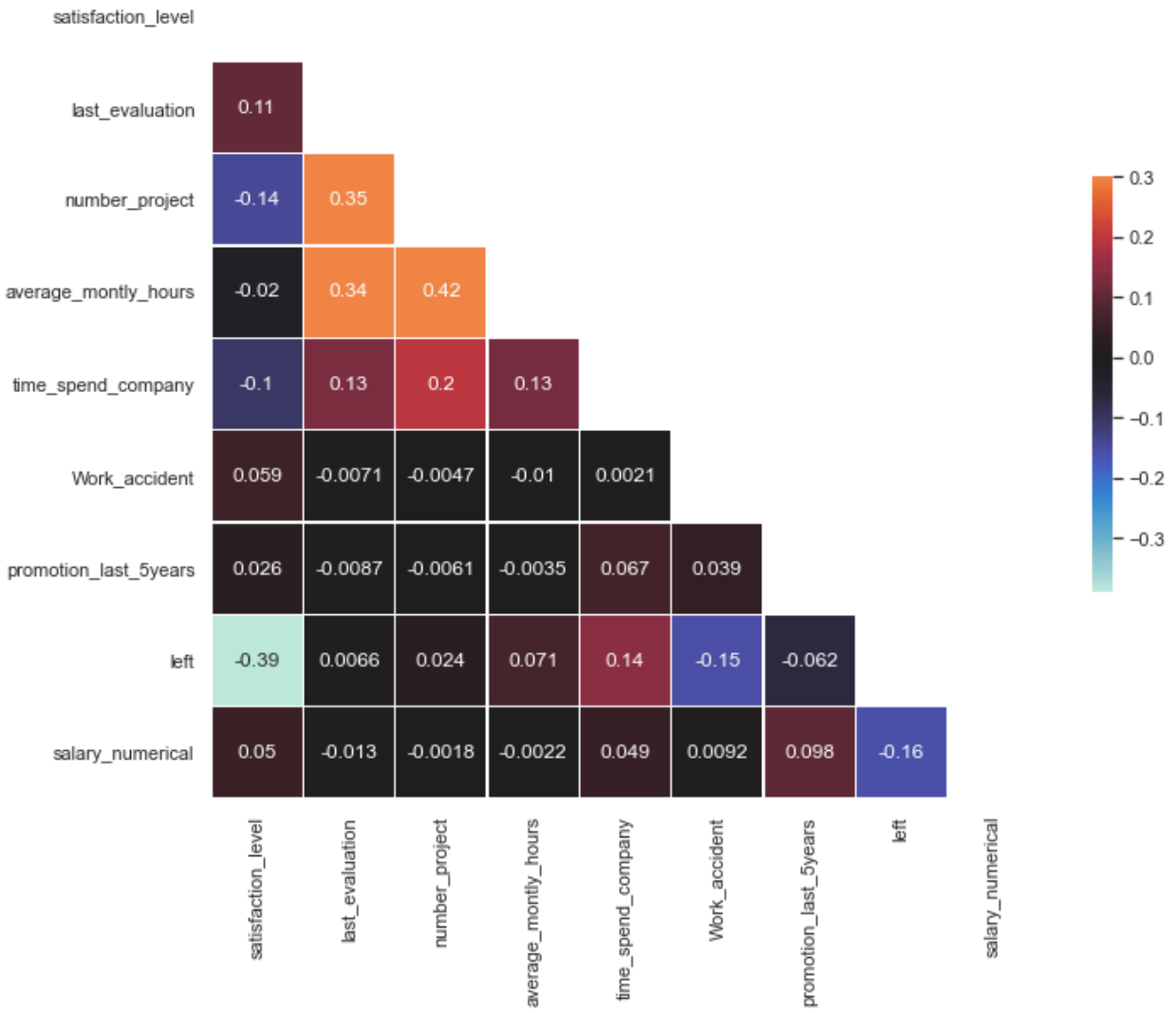

Figure 1.10

There are following finding that we get from the above color matrix

1. This correlation matrix tells us that the higher someone is satisfied in their job the less likely they are to leave.

2. Time spent with the company is positively correlated with leaving,
3. suggesting the longer someone works for this organization the more likely they are too leave.
4. Another relationship that makes sense is that a higher salary appears to be negatively associated with leaving.
5. One correlation in the above matrix doesn't quite make sense.
6. Having had a work accident is negatively related to leaving the company, which doesn't really make intuitive sense. On the surface this doesn't make a whole lot of sense because you would assume that an employee felt unsafe at an organization that they might want to leave.

Exploring People Left in Detail

Count of feature people left:

```
data.left.value_counts()
```

```
0    11428
1     3571
Name: left, dtype: int64
```

Figure 1.11(a)

Here, you can see out of 14,999 3,571 were left, and 11,428 stayed. The number of employee left is approximately 23 % of the total employment.

```
left = data.groupby('left')
left.mean()
```

left	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years	salary_numerical
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251	1.650945
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321	1.414730

Figure 1.11(b)

Here you can interpret, Employees who left the company had low satisfaction level, low promotion rate, low salary, and worked more compare to who stayed in the company.

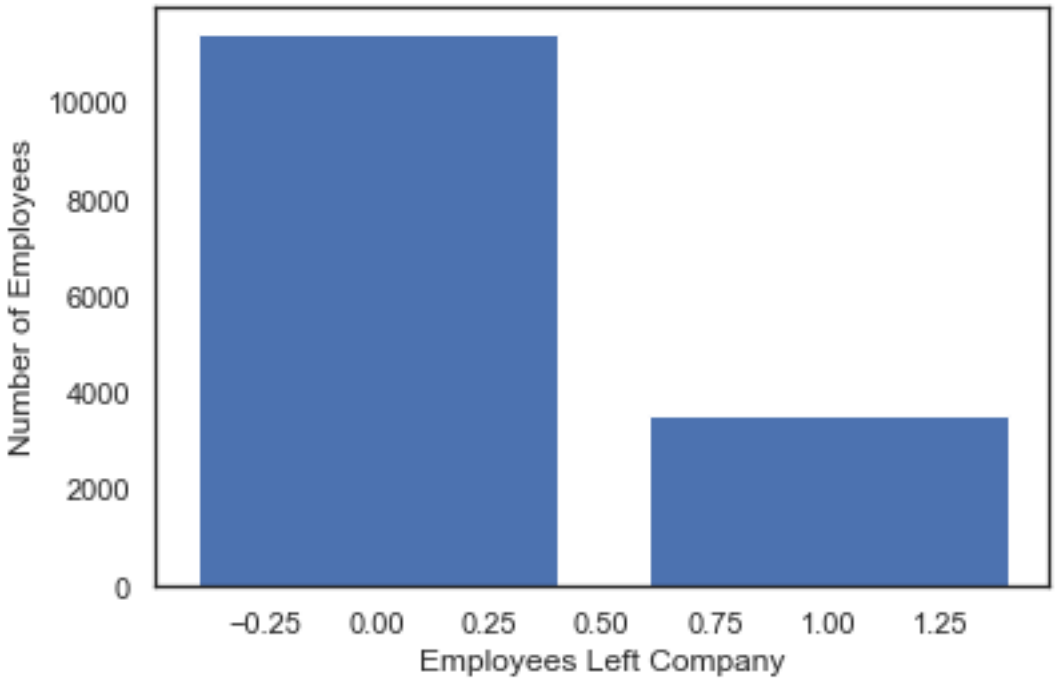

Figure 1.11(c)

Most of the employee are doing the project from 3-5:

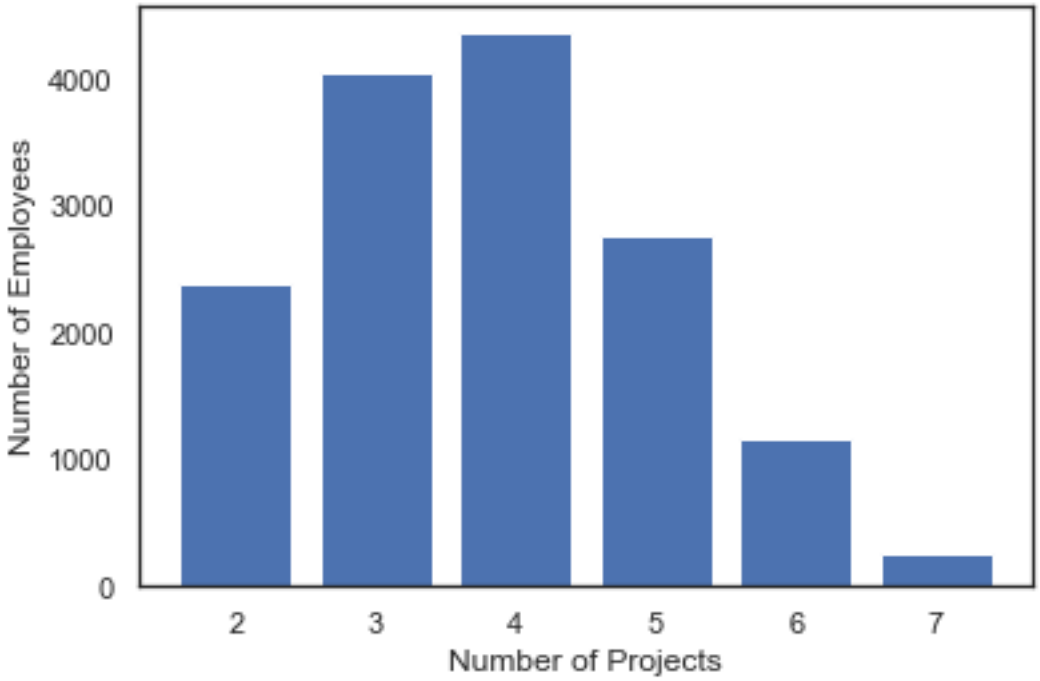

Figure 1.12

Most of the employee experience between 2-4 years. Also, there is a massive gap between 3 years and 4 years experienced employee.

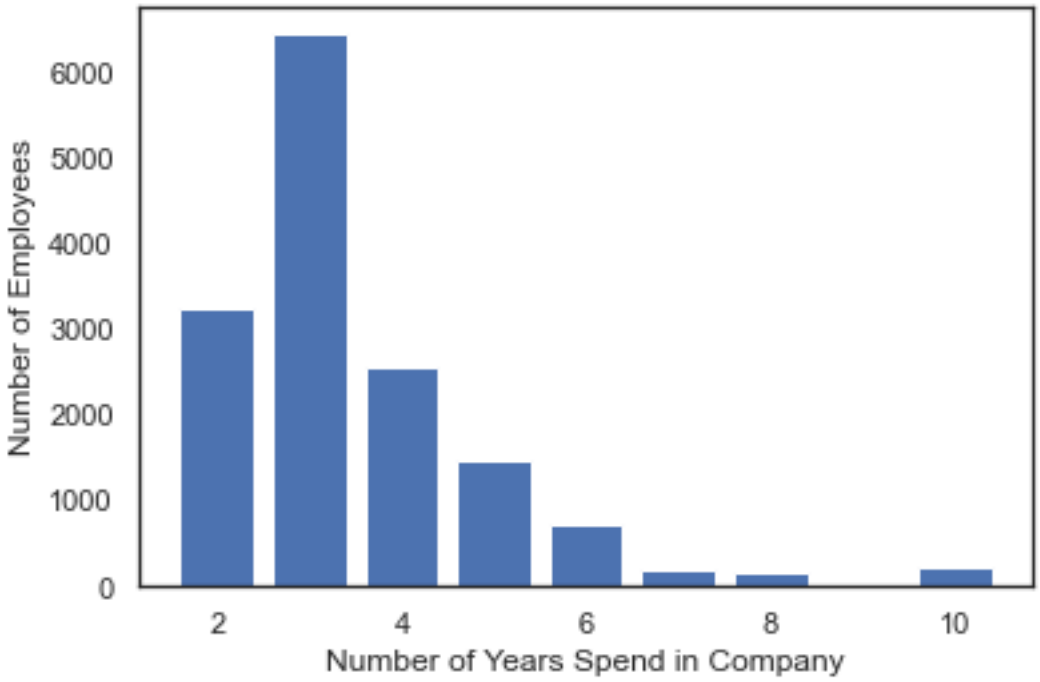

Figure 1.12

You can observe the following points in the below visualization (Figure 1.13):

- Most of the employee is doing the project from 3-5.
- There is a huge drop between 3 years and 4 years experienced employee.
- The no of employee left is 23 % of the total employment.
- A decidedly smaller number of employees get the promotion in the last 5 year.
- The sales department is having maximum no. of employee followed by technical and support
- Most of the employees are getting salary either medium or low.

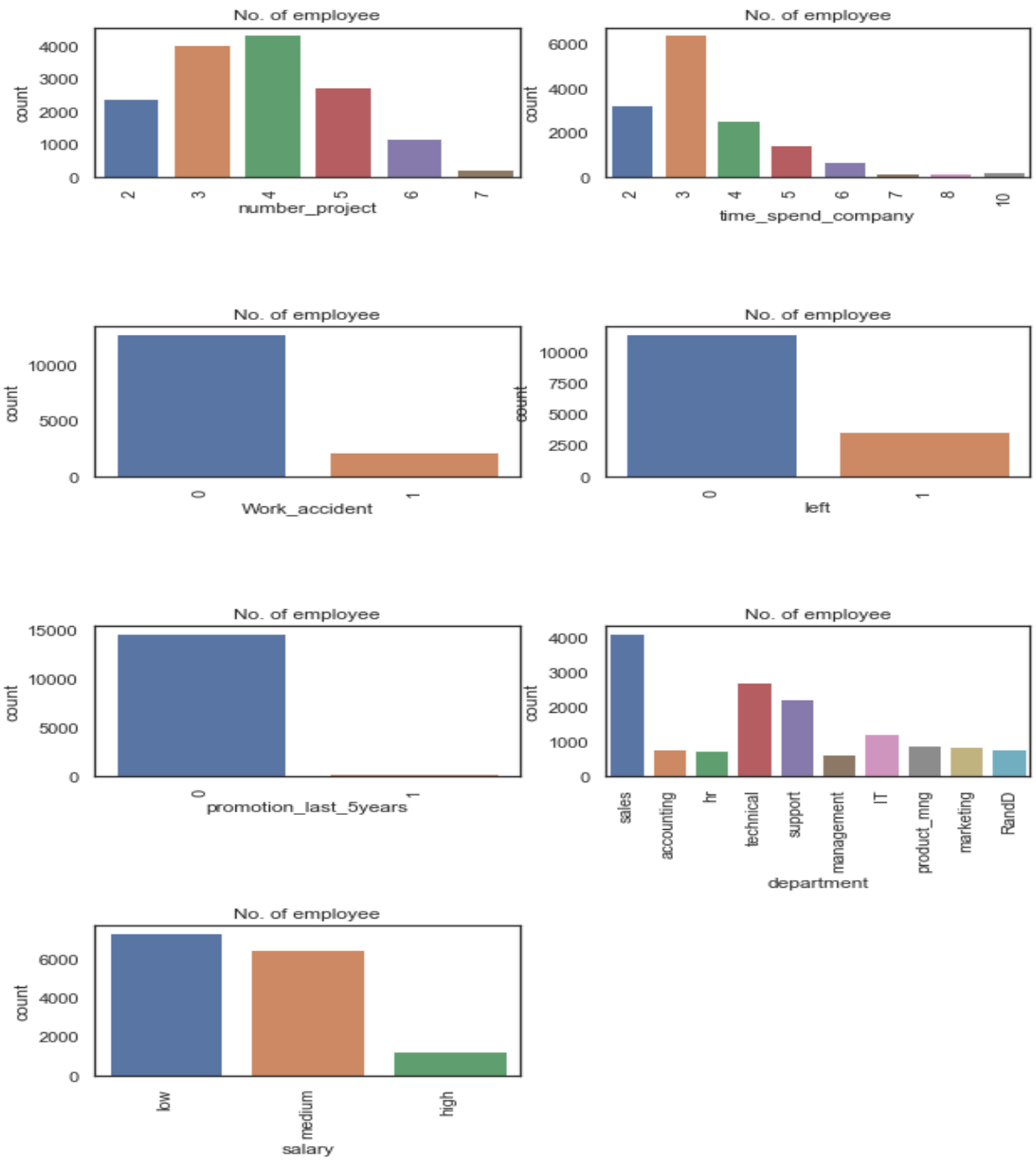

Figure 1.13

You can observe the following points in the below visualization (Figure 1.14):

- Those employees who have the number of projects more than 5 were left the company.
- The employee who had done 6 and 7 projects, left the company it seems to like that they were overloaded with work.

- The employee with five-year experience is leaving more because of no promotions in last 5 years and more than 6 years' experience are not leaving because of affection with the company.
- Those who promotion in last 5 years they didn't leave, i.e., all those left they didn't get the promotion in the previous 5 years.

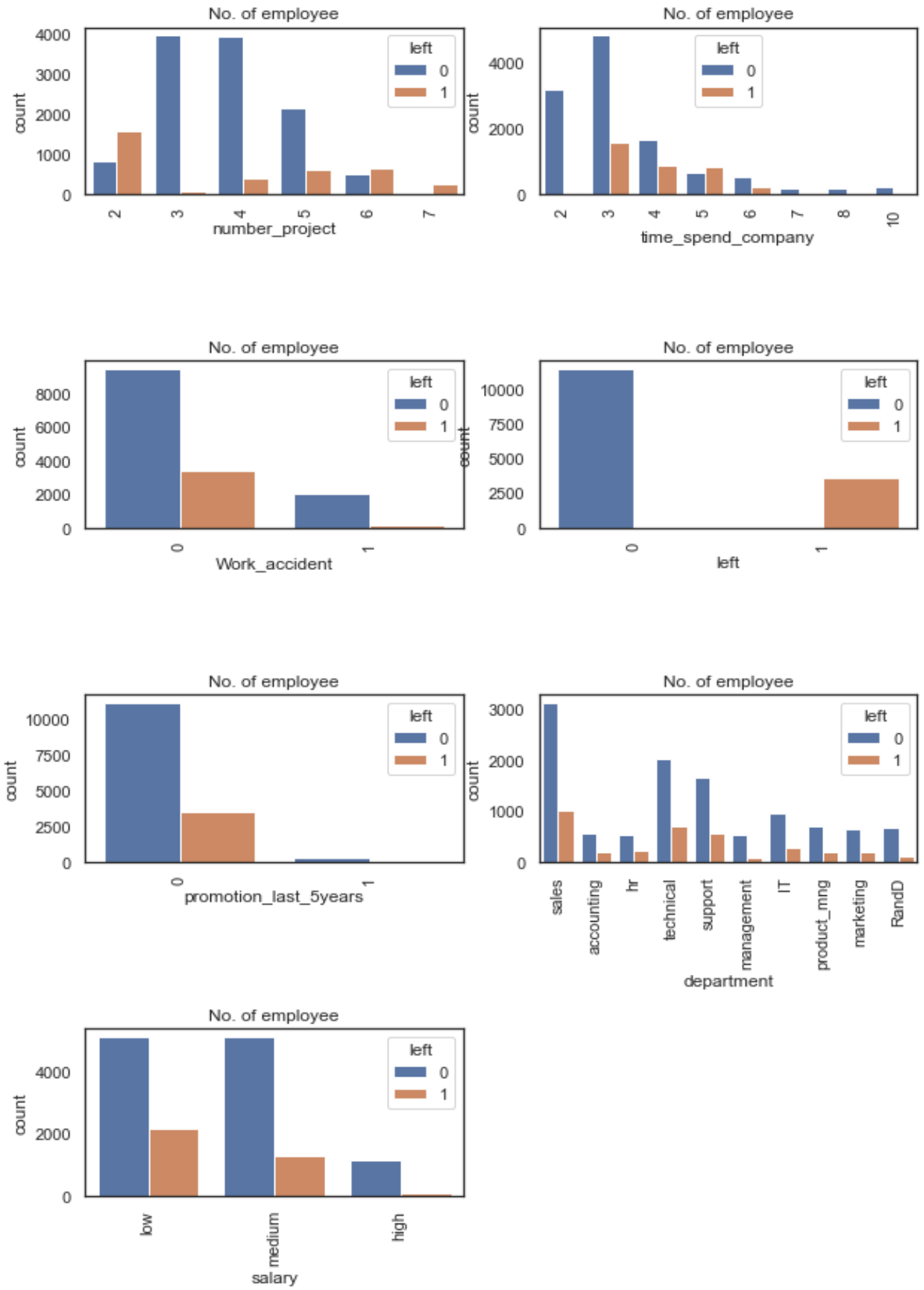
No. of employee
left
0
1
count
number_project
time_spend_company
Work_accident
left
promotion_last_5years
department
sales
accounting
hr
technical
support
management
IT
product_mng
marketing
RandD
salary
low
medium
high

Figure 1.14

What makes people leave company?

Department & People left

In order to see the relation of why people leave the company, we will compare it with certain columns and will compare the result to obtain an informed decision. First, we compare it with department, and for this we took mean of department by group by it and then make a histogram graph.

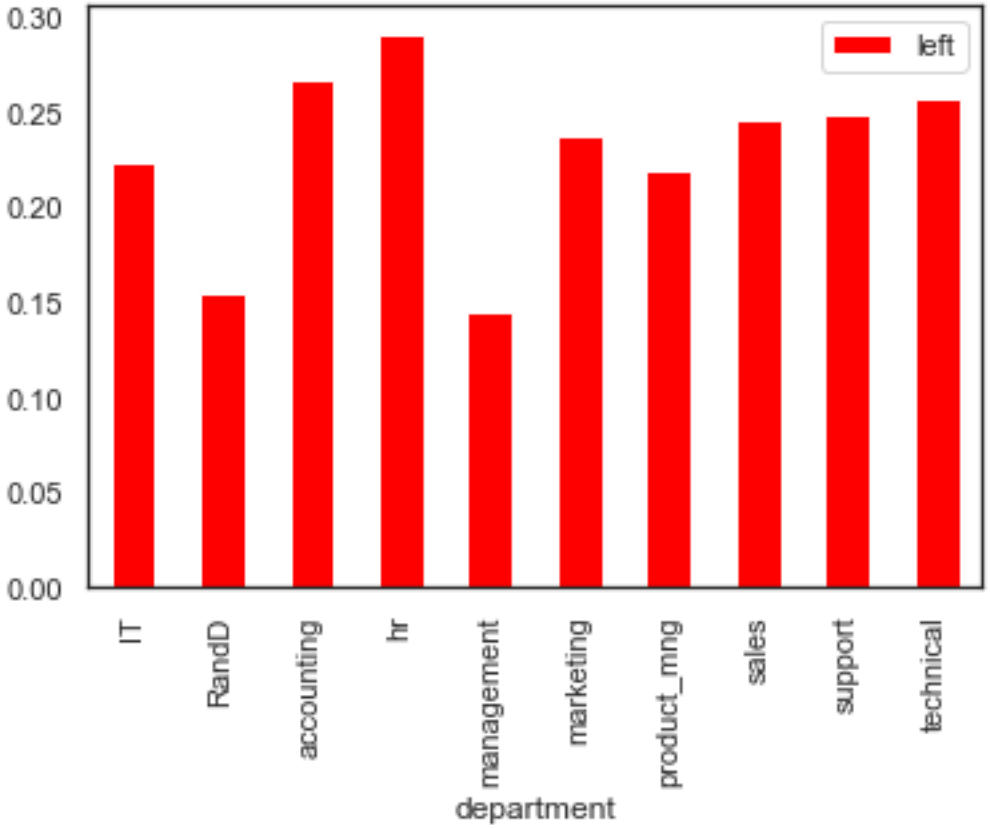

Figure 1.15

Findings:

1. We See that hr., accounting and Technical Departments have more employees who left the company.

2. However, without knowing the total staffing count at the organization, this variable isn't too helpful.

Satisfaction & People Left

Here we made a function satisfaction that will take one argument row will return qualitative ranking i.e. high, low, medium. we have divided the certain ranges into high, low and medium so we can make a better decision from graph. We added a new column and assigned qualitative ranking for three satisfaction level (high, low, medium).

```
#Creating a new column and assigning qualitative rankings for three(3) Satisfaction levels(High, Medium and Low).
#and adding a new column "Satisfaction" to hold the qualitative rankings.

def satisfaction(row):
    if row['satisfaction_level'] >= .5:
        return 'medium'
    elif .3 <= row['satisfaction_level'] < .5:
        return 'high'
    else:
        return 'low'

data['qualitative_satisfaction'] = data.apply(satisfaction, axis=1)

data_satisfaction = data.groupby(by='qualitative_satisfaction',as_index=False).mean()
data_satisfaction.head()
```

	qualitative_satisfaction	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5year
0	high	0.411455	0.573588	2.703469	161.674375	3.271913	0.096979	0.01566
1	low	0.152213	0.781651	5.291272	236.805468	4.288644	0.117245	0.01209
2	medium	0.748778	0.740814	3.814324	204.656394	3.412154	0.161866	0.02438

Figure 1.16(a)

New column holding the ranking values for each of the column in dataset can be seen in above figure 1.16(a).

The graph for the above ranking column is shown by the following figure.

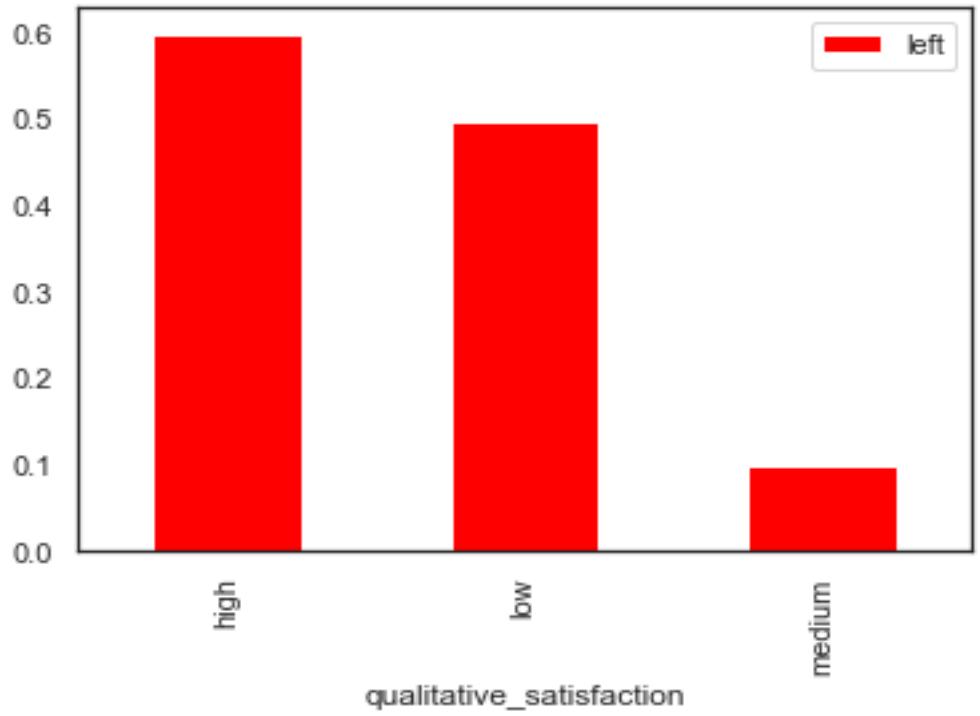

Figure1.16(b)

Findings:

1. It is interesting. Those with high satisfaction levels are more likely to leave based on percentage of those who left, which doesn't make intuitive sense.

2. If we use our three-category breakdown. This chart suggests that those who have medium satisfaction are the least likely to leave.

Salary & People Left:

Here first, we group data by salary and find the count and then plot that group data similarly we have done in above cases.

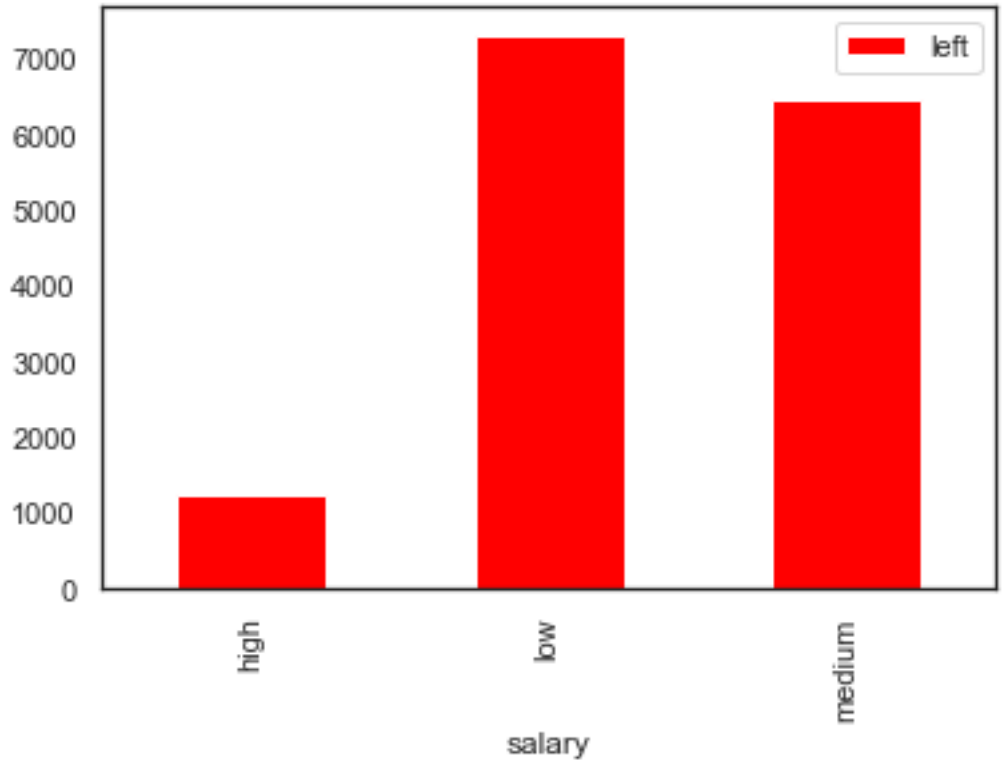

Figure 1.17

Finding:

1. Salary have a linear relationship with leaving, with the more money you earn the less likely you are to leave.

Collective Effect of Salary & Satisfaction level on Company Leaving:

In order to see the collective effect of salary and satisfaction level on why people leave the company, we group by the data with both factors as show below then plot the graph with these both factors taking together.

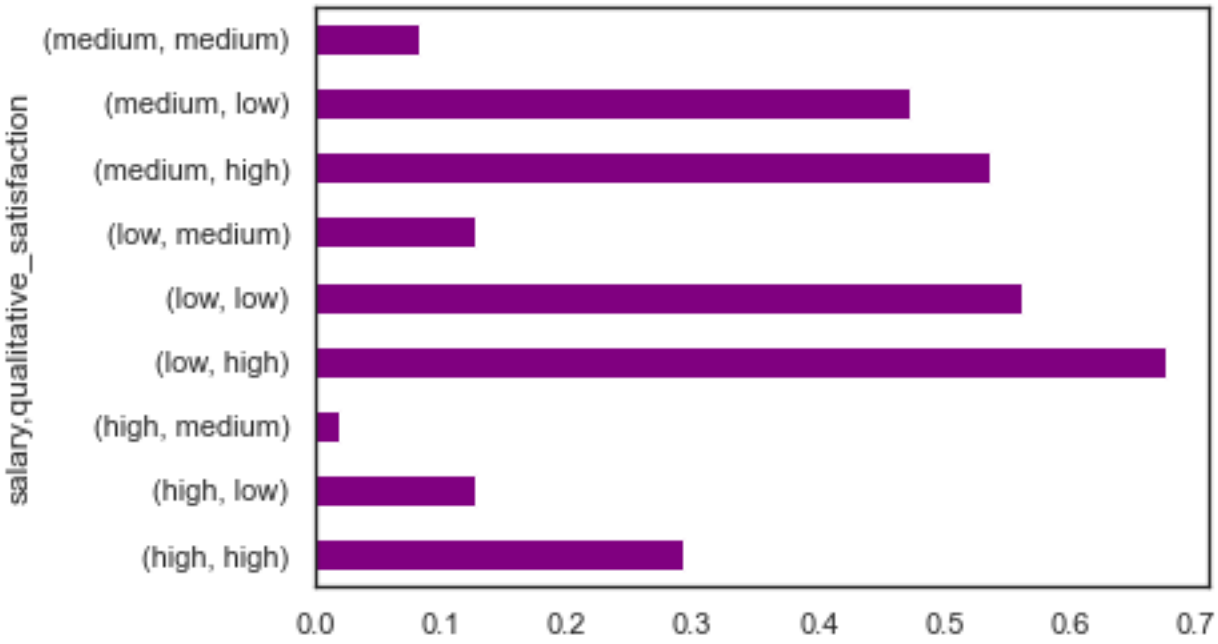

Figure 1.18

Figure 1.18 shows the collective effect on leaving the company and the following findings are obtained from the graph.

Findings:

1. Employees with the highest rate of leaving earns a low salary and reports a medium satisfaction rate.
2. Employees with the lowest rate of leaving earns the highest salary and reports medium satisfaction.

Reasons behind various Satisfaction Level:

If we plot the satisfaction level graph, then we get that the curve is not normally distributed in the dataset. The following figure proves our claim.

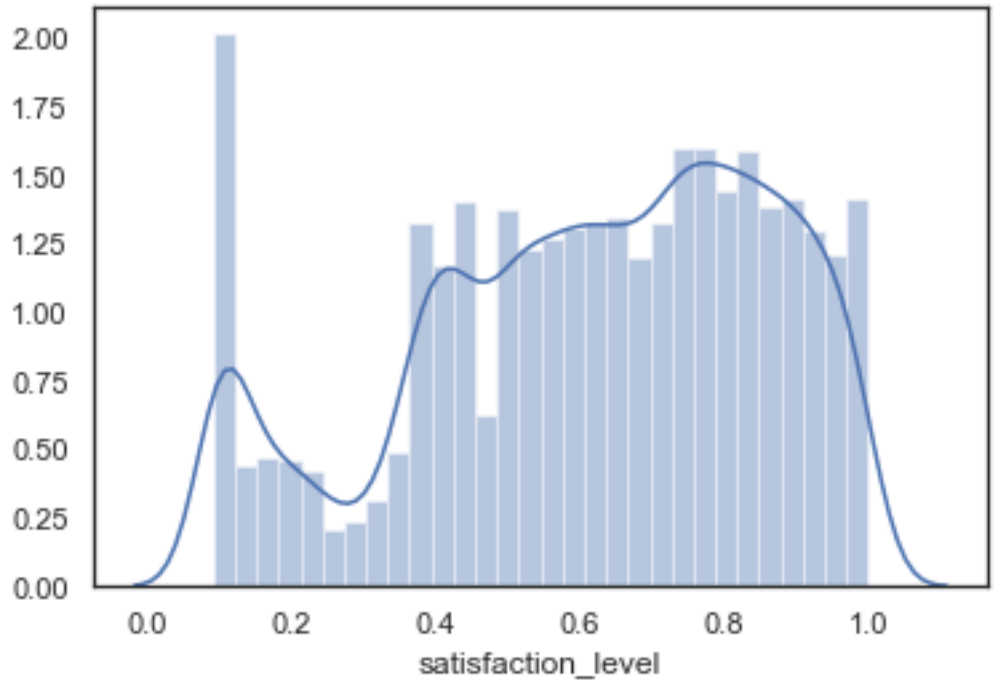

Figure 1.18

To get the reason that we have different satisfaction ranking for different columns we will compare each of the column with satisfaction level and plot their graphs to deduce findings. The dataset is grouped by qualitative satisfaction and the mean is stored in data satisfaction. The plots of given statements are given below in respective manner.

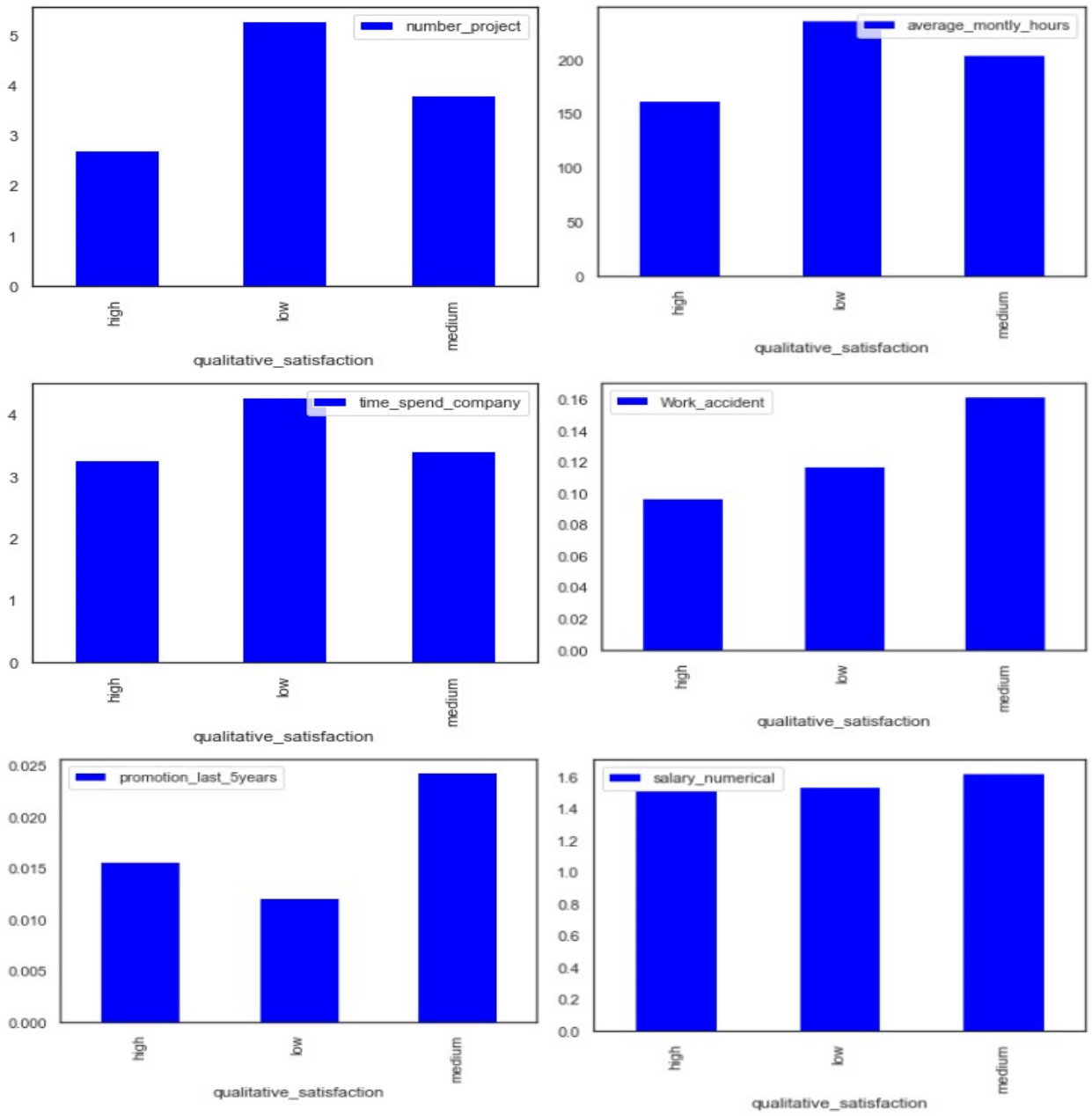

Figure 1.19

Findings:

1. This shows people with lowest number of projects and lowest number of time spend company, monthly work hours have highest satisfaction and people with most number of projects and highest monthly hours tends to loose internal peace.

2. Total time spent in company doesn't seem to have much effect on satisfaction, but people who spend 4 or more years tends to have low satisfaction, i.e. passing time effects their satisfaction.

3. People who have the greatest number of work accidents have medium satisfaction, but a smaller number of work accidents doesn't promise more satisfaction.

4. Similarly, promotion in last five years doesn't have linear effect on satisfaction.

5. Salary doesn't seem to have any effect on satisfaction, as people with high, medium or low satisfaction have same average salary. This is quite as because salary is believed to have a major effect on employees' satisfaction.

Cluster Analysis:

Preprocessing:

Let us find out the number of clusters by k-means should be made and how salary depends upon the department number. The department is a categorical value and we should transform it into numerical values.

```
#data processing
dep_sal_backup = data[['department','salary']]
le = LabelEncoder()
k = le.fit_transform(data['department'])
data['department_numerical'] = k
data.drop([ 'salary', 'department', 'qualitative_satisfaction'], axis=1, inplace=True)
data.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years	left	salary_numerica
0	0.38	0.53	2	157	3	0	0	1	
1	0.80	0.86	5	262	6	0	0	1	
2	0.11	0.88	7	272	4	0	0	1	
3	0.72	0.87	5	223	5	0	0	1	
4	0.37	0.52	2	159	3	0	0	1	

Figure 2.1

Now we have numerical values for department and salary. We can now make clusters of these and study the behavior of these attributes.

K-Means Clustering:

K means is the clustering algorithm in which we make clusters depending upon the average collection of data points in a graph. If the data points are seemed to be scattered e.g. 5 parts, then the clusters made should be 5.

We transform data into a NumPy array.

```
X = data.to_numpy()
# Normalized numpy array, used later in the visualization step
```

Figure 2.2

Elbow Method to Find Optimal Number of Clusters:

So, we decided to use k-means algorithm and make clusters range between 1 to 10 and check what is the best fit for the dataset. This helped us a lot. We could make a graph on clusters numbers vs distortion.

```
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k, random_state = 0)
    kmeanModel.fit(X)
    distortions.append(kmeanModel.inertia_)
```

Figure 2.3(a)

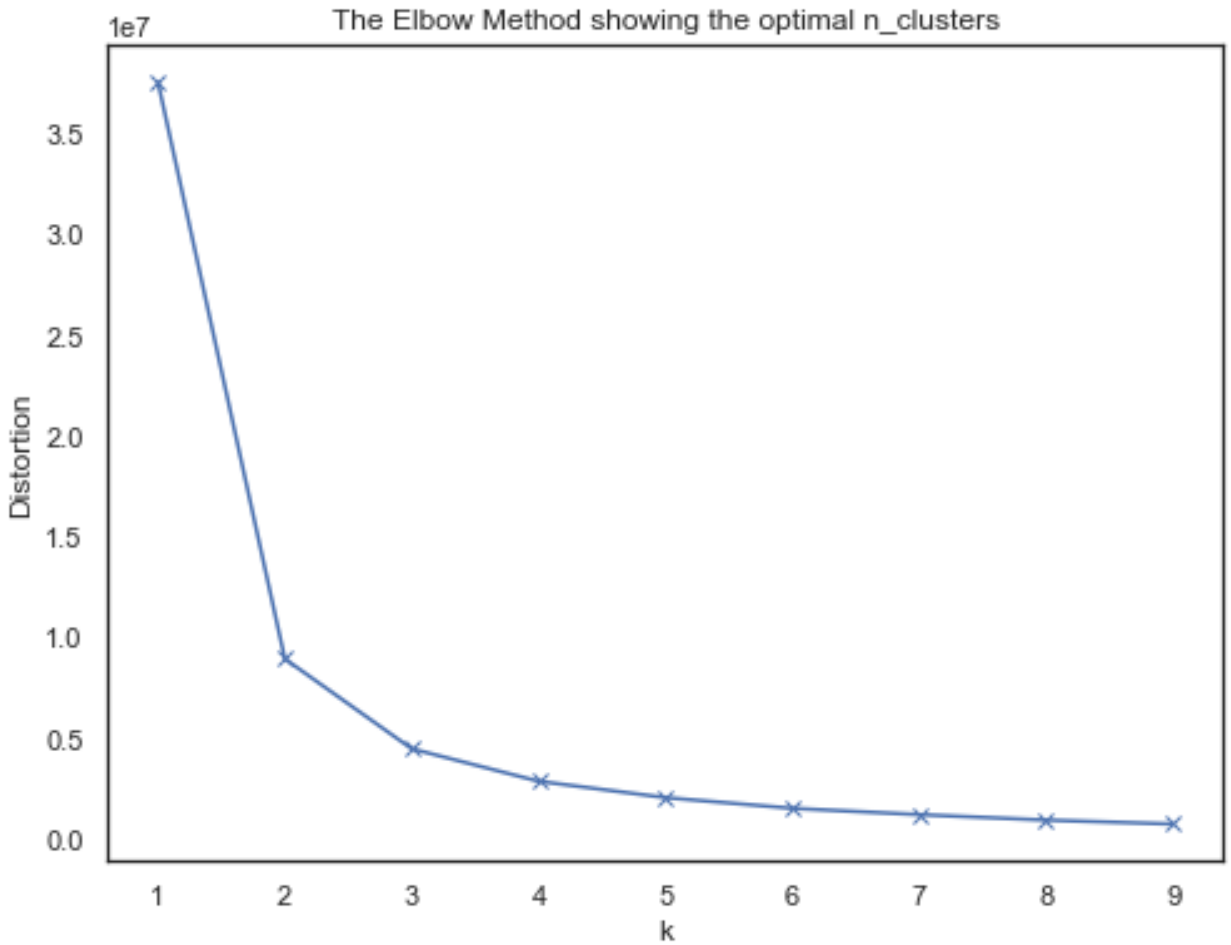

Figure 2.2(b)

We can find out through this that the optimal number of clusters for this dataset should be 3. The distortion is very low here and the clustering algorithm can work fine on this.

Implementation:

We know that the optimal number of clusters for this data set is 3. Let's use k-means and make three clusters. Let us observe the most important factor for the salary amount of any employee and implement k-means algorithm for our dataset.

```
# Initialize
kmeans = KMeans(n_clusters=3, random_state = 0)
# Fit
kmeans.fit(X)
# Print labels
print(kmeans.labels_)
```

```
[2 1 1 ... 2 1 2]
```

Figure 2.3

Here the label number shows the cluster number that data point x belongs to and we are now going to make graphs for the different clusters and study them. We used simple plots to see the relations. (Figure 2.4)

Characteristic Analysis:

1. Lowest rating at Last Evaluation, least the number of projects, Lowest Average Monthly hours
2. Highest rating at Last Evaluation, most the number of projects, Highest Average Monthly Hours, Highest left rate
3. Highest Satisfaction, most the work accidents, Highest promotion, Lowest left rate

All three clusters have almost equal salary, department distribution, time spend company

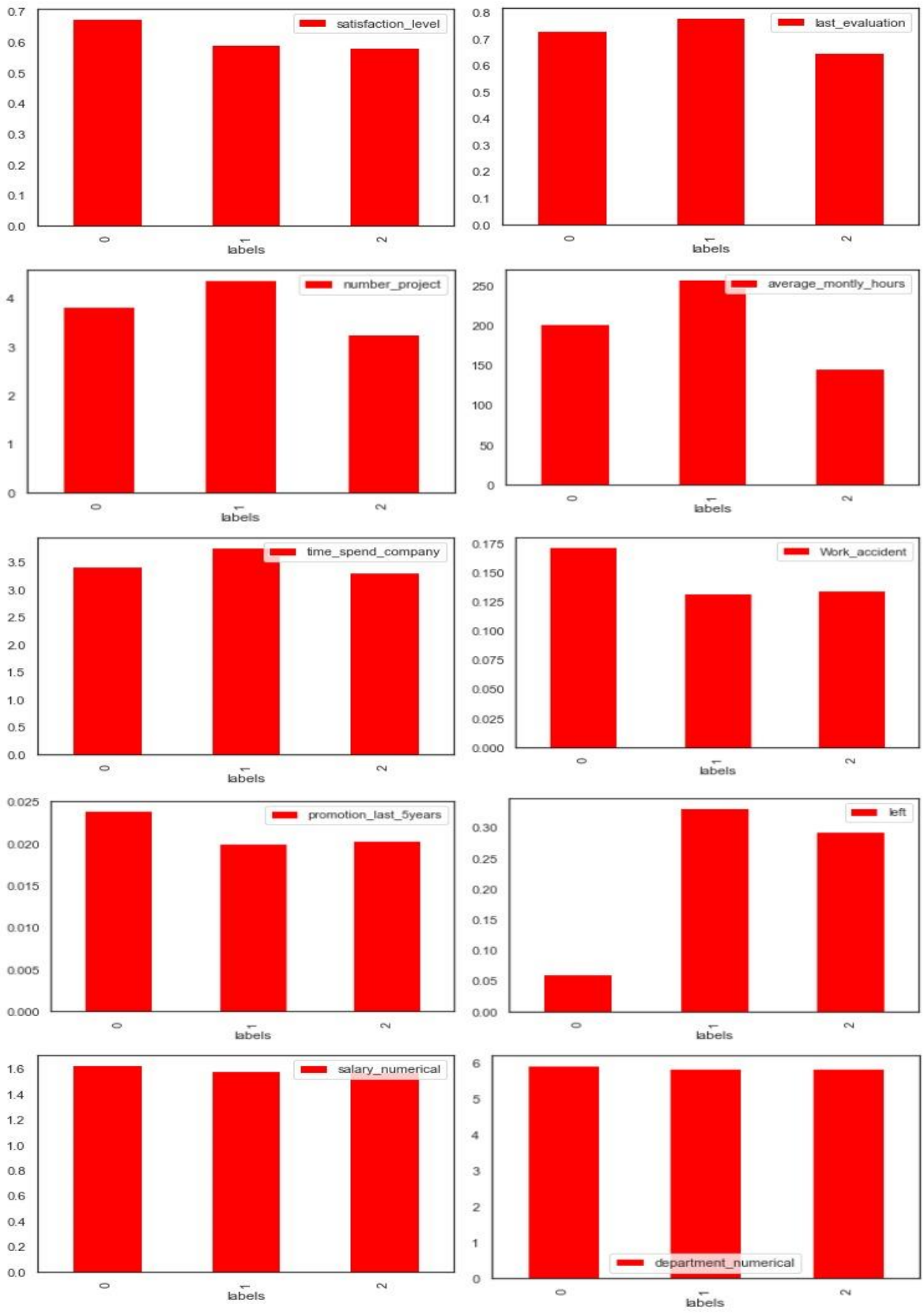
satisfaction_level
last_evaluation
number_project
average_montly_hours
time_spend_company
Work_accident
promotion_last_5years
left
salary_numerical
department_numerical
labels

Figure 2.4

Outlier Detection:

Outlier detection can be done on “satisfaction_level”, “last_evaluation”, “number_project”, “average_montly_hours”, “time_spend_company” because the rest of the attributes are Boolean or Categorical. So, we going to drop other columns and start our outlier analysis on the remaining columns. We have dropped the attributes we do not need.

```
outlier_data = data.drop([ 'Work_accident', 'promotion_last_5years', 'left', 'salary_numerical', 'department_numerical'], axis
='columns')
outlier_data.head()
```

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

Figure 3.1

Boxplot:

We first normalized the values so that they can be graphed easily with other attributes. The different columns have different outlier value which goes beyond the standard deviation value of these values.

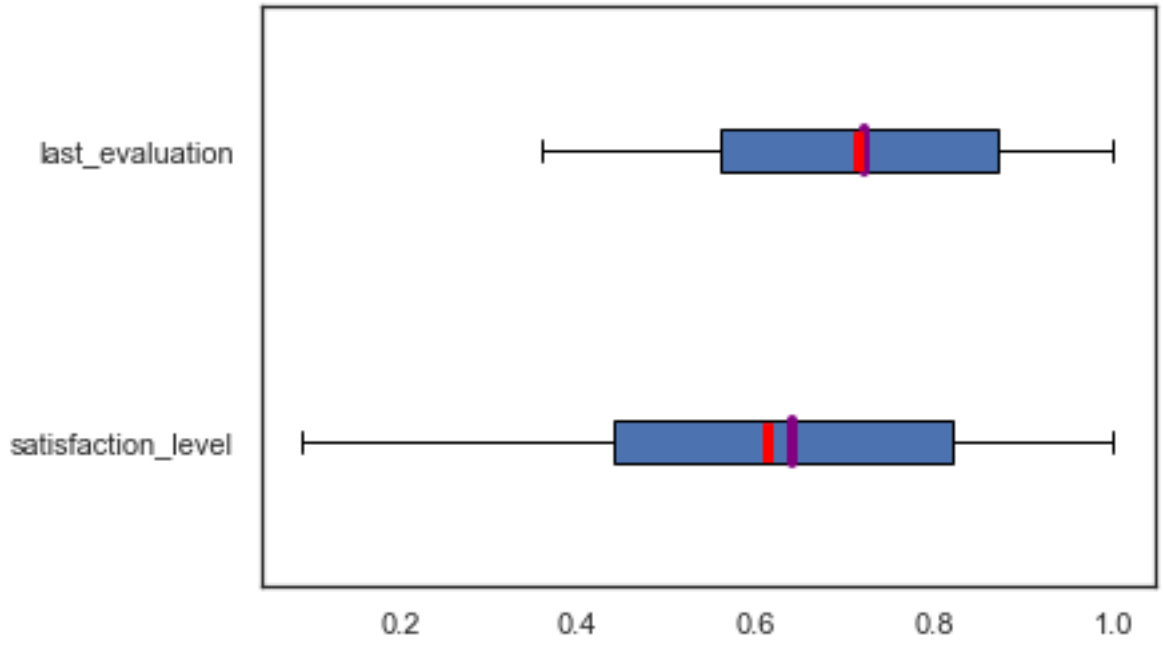

Figure 3.2(a)

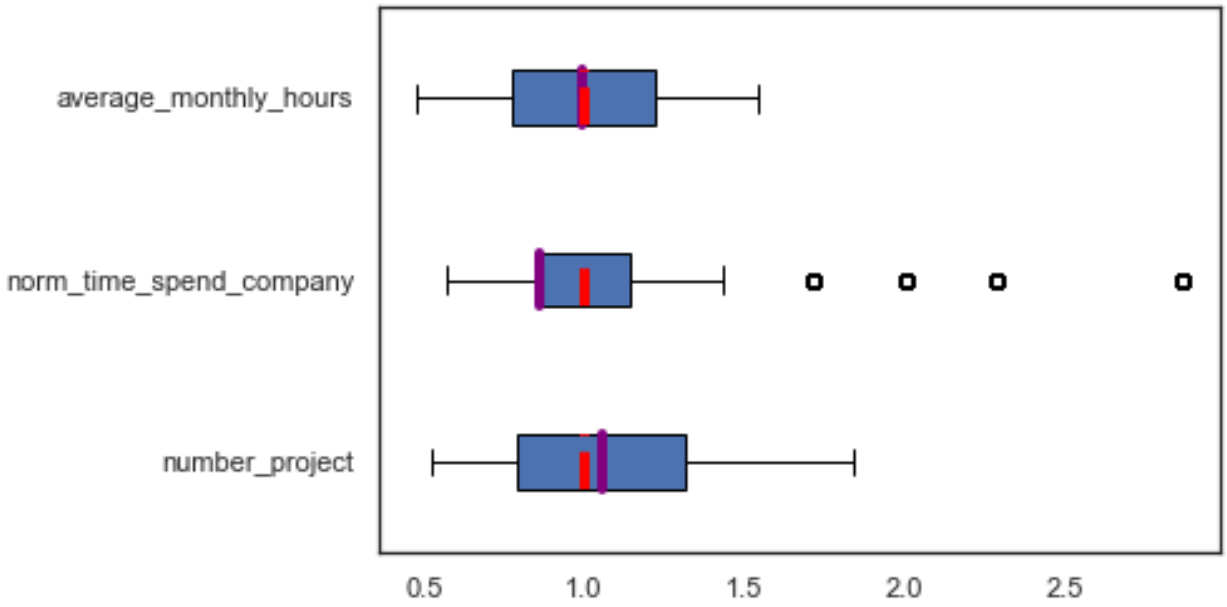

Figure 3.2(b)

Findings:

We can see that there are many values in the column “time_spend_company” which cause our dataset have outliers and there might be different outlier values in different attributes as well which can cause outliers in our clustering model.

ZScore Analysis:

Z-score outlier analysis is the famous algorithm to find the outliers in the dataset. If a point in our dataset goes out of the threshold defined by us, it would consider as an outlier. The implementation of Z-score outlier detection is given below:

```
score = np.abs(zscore(zscore_data))
threshold = 3
zoutliers = np.where(score > threshold)
# type(outliers)
zoutliers_in_dataset = pd.DataFrame(zoutliers).transpose()
zoutliers_in_dataset
```

	0	1
0	11007	4
1	11008	4
2	11009	4
3	11010	4
4	11011	4
...	...	...
371	14205	4
372	14206	4
373	14207	4
374	14208	4
375	14209	4

376 rows × 2 columns

Figure 3.3

This is the basic implementation of z-score outlier detection where we defined out threshold to be 3. The reason of this threshold is because we wanted to take out the most extreme values out of dataset. Each row references to a datapoint in **zscore_data**. The first element contains the row number and second element has column number, which mean zscore_data [11007][4] have a Z-score higher than 3.

Findings:

ZScore Analysis suggest the similar results to boxplot. We see that only 4th column i.e. “time_spend_company” has the outliers.There are total 376 outliers.

IQR Analysis

Interquartile Rule is another algorithm of finding the outliers in the dataset. In this algorithm, we normalize the datapoints and then take values between 25th and 75th percentile. We then take out the values that lie between these values find the outlier by using a constant k which is the degree of how much the outlier is at the extreme position.

```
for col in iqr_data.columns:
    Q1, Q3 = np.percentile(iqr_data[col], 25), np.percentile(iqr_data[col], 75)
    IQR, k = (Q3 - Q1), 3
    lower_Outlier, higher_Outlier = Q1 - (k * IQR), Q3 + (k * IQR)
    # identify outliers
    iqr_outliers = [j for j in iqr_data[col] if j < lower_Outlier or j > higher_Outlier]
    print("\n" + col + "\n")
    print(iqr_outliers)
```

```
satisfaction_level

[]

last_evaluation

[]

number_project

[]

average_montly_hours

[]

time_spend_company

time_spend_company

[8, 8, 8, 8, 8, 10, 8, 8, 8, 8, 8, 10, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 1
0, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 1
0, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8,
8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10,
8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10, 8, 8, 8, 8,
8, 8, 8, 8, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 8, 8, 8, 8, 8, 10, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8,
8, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8,
8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 1
0, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8,
8, 10, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 10, 10, 10, 8, 8, 8, 10, 10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 8, 8, 8,
10, 10, 10, 10, 10, 10, 8, 8, 8, 8, 8, 8, 8, 10, 10, 10, 10, 10]
```

Figure 3.4

Findings:

IQR shows the same behavior as Boxplot and ZScore. These lists show the outliers that exist in each attribute. According to IQR, there are no outliers in attributes that are “satisfaction_level”, “last_evaluation”, “number_project” and “average_monthly_hours”. But for the case of time spend in the company there are many outliers that exist. Each outlier is listed.

Prediction Algorithms:

Now here, we are using some predictions algorithms on our dataset to predict which employ will leave the company. For this first we are loading the train and test data and we used the built-in function train_test_split of python library sklearn as it is shown in the below picture:

```
In [42]: X=data.drop(['left'],axis=1)
         Y=data['left']

         X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,)

In [43]: features = list(X_test.columns)
         X_test
```

Out[43]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years	salary_numerical
11678	0.27	0.42	6	173	7	0	0	2
6876	0.79	0.84	4	144	2	1	0	2
1139	0.42	0.46	2	132	3	0	0	1
2557	0.61	0.55	5	260	3	0	0	2
607	0.11	0.88	5	250	4	0	0	2
...	...	...	...	...	...	...	...	...
384	0.36	0.48	2	156	3	0	0	3
6065	0.70	0.73	2	139	2	0	0	2
1345	0.38	1.00	5	137	4	0	0	1
8084	0.67	0.39	2	235	6	0	0	1
6999	0.90	0.69	3	231	4	0	0	2

4500 rows × 9 columns

Figure 4.1

This library will differentiate the matrix into train and test subsets.

Logical Regression

It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

```
lr = LogisticRegression(solver = 'lbfgs', max_iter = 400)
lr.fit(X_train, Y_train)
predict = lr.predict(X_train)
print("Logistic Regression Training Score is",lr.score(X_train,Y_train))
print("Logistic Regression Testing Score is",lr.score(X_test,Y_test))
```

```
Logistic Regression Training Score is 0.7841699209448519
Logistic Regression Testing Score is 0.7926666666666666
```

Figure 4.2(a)

Accuracy:

In figure 4.2(b), we can see the accuracy of logistical regression is 79.267% on test data.

```
accuracy_lr = lr.score(X_test,Y_test)*100
print("Accuracy of Logistic Regression is ",accuracy_lr)
```

```
Accuracy of Logistic Regression is  79.26666666666667
```

Figure 4.2(b)

Estimates:

The table below shows the main outputs from the logistic regression. The coefficients appear in the column against all features are called Estimate. The negative sign with the estimate of "satisfaction_level" suggests that people with higher satisfaction level has the low tendency to leave the company, and the positive sign with the "last_evaluation" means people who had higher score at last evaluation are more likely to leave the company.

```
for col,value in zip(data[features].columns,lr.coef_[0]):
    print(col,"*",value,"+")
print(lr.intercept_[0])
```

```
satisfaction_level * -4.005144518651388 +
last_evaluation * 0.6788459868846111 +
number_project * -0.2881423550512125 +
average_montly_hours * 0.004506505161526785 +
time_spend_company * 0.25413583426841696 +
Work_accident * -1.5512901712901614 +
promotion_last_5years * -0.7792496574541395 +
salary_numerical * -0.6745161956259008 +
department_numerical * 0.025534438547443895 +
0.8766782547996771
```

Figure 4.3

Confusion Matrix and Classification Report:

Confusion matrix describes the individual results of each outcome. As we have Left as a binary variable, we have 2 by 2 matrix, it can be read in this way:

Total = 4490	Predicted Not Left	Predicted Left	
Actual Not Left	True Negative = 3198	False Positive = 217	217+3198 = 3415
Actual Left	False Negative = 716	True Positive = 369	716+369 = 1075

Accuracy can be calculated as (True Negative + True Positive)/Total = 79.44 %.

```
print("Confusion Matrix for logistic Regression")
print(confusion_matrix(Y_test, lr.predict(X_test)))
```

```
Confusion Matrix for logistic Regression
[[3198  217]
 [ 716  369]]
```

Figure 4.4

Classification Report can be viewed in Figure 4.5:

```
print("Classification Report for Logistic Regression")
print(classification_report(Y_test, lr.predict(X_test)))
```

```
Classification Report for Logistic Regression
              precision    recall  f1-score   support

           0       0.82      0.94      0.87      3415
           1       0.63      0.34      0.44      1085

    accuracy                           0.79      4500
   macro avg       0.72      0.64      0.66      4500
weighted avg       0.77      0.79      0.77      4500
```

Figure 4.5

AUC-ROC curve:

Now we'll use AUC-ROC curve to measure performance of the classification problem at various

thresholds settings. ROC is a probability curve and AUC represents degree or measurement of the classification problem. Higher the AUC, better the model at predicting.

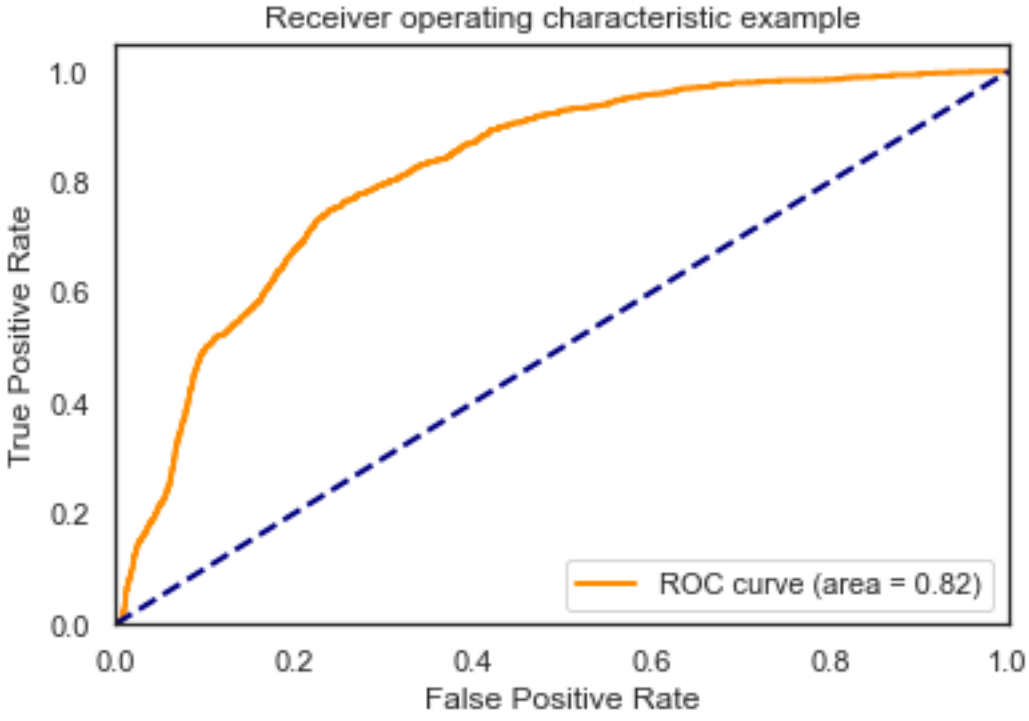

Figure 4.6

As we can see our model is pretty good at predicting ROC curve covers 0.82 area (closer to 1, means higher accuracy). But this model has a lower specificity. So, we'll implement a few more algorithms to achieve better accuracy and specificity.

Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes).

```
dt=DecisionTreeClassifier(criterion = 'entropy')

dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, 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, presort='deprecated',
                       random_state=None, splitter='best')
```

Figure 4.7

Accuracy**:**

On test data we achieved the accuracy of 97.93 %.

```
prediction_dt=dt.predict(X_test)
accuracy_dt=accuracy_score(Y_test,prediction_dt)*100
print('Accuracy = ' + str(accuracy_dt))
prediction_dt
```

```
Accuracy = 97.93333333333332

array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

Figure 4.8

Confusion Matrix and Classification Report:

We get the confusion matrix and classification report for the decision tree as follows:

```
print("Confusion Matrix for Decision Tree")
print(confusion_matrix(Y_test, prediction_dt))
```

```
Confusion Matrix for Decision Tree
[[3365   50]
 [  43 1042]]
```

Figure 4.9(a)

```
print("Classification Report for Decision Tree")
print(classification_report(Y_test, prediction_dt))
```

```
Classification Report for Decision Tree
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3415
           1       0.95      0.96      0.96      1085

    accuracy                           0.98      4500
   macro avg       0.97      0.97      0.97      4500
weighted avg       0.98      0.98      0.98      4500
```

Figure 4.9(b)

From Figure 4.9(a), we can see we achieved slightly better results than logistical regression. Now we'll print the decision tree (A tree.dot file is also attached with the submission).

```
dot_data = StringIO()
export_graphviz(dt, feature_names = features, out_file=dot_data,
                filled = True, rounded = True,
                special_characters= True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

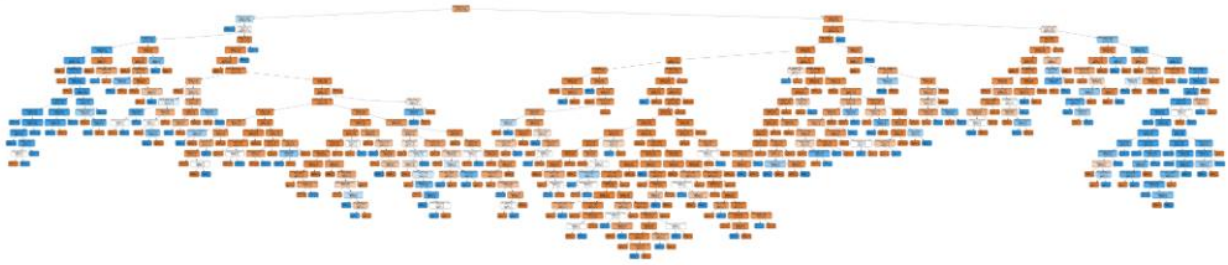

Figure 4.10

This tree helps us to visualize test on each node, threshold values used, and consequences of each test.

Feature Importance:

According to decision tree we can see the importance of each feature in the below table (Figure 4.11). It's clear that the satisfaction level has the highest effect on whether a person leaves the company or not. Alternatively, we can say the promotion, work accident, and salary has the lowest effect on the retention of the employee. This is quite interesting as the salary is supposed to have a major impact on a person's decision to leave the company.

```
feature_importance=pd.DataFrame(dt.feature_importances_,index=X_train.columns,columns=['Importance']).sort_values('Importance',a
scending=False)
feature_importance
```

	Importance
satisfaction_level	0.361205
time_spend_company	0.191177
number_project	0.180883
last_evaluation	0.130857
average_montly_hours	0.108374
department_numerical	0.016579
salary_numerical	0.006237
Work_accident	0.003955
promotion_last_5years	0.000734

Figure 4.11

Now we test our model on a custom-made employee who has high satisfaction, moderate number of hours worked weekly, medium salary and other attributes. We see that our model predicts that employee will stay.

Test:

```
#checkin our model for anyother data
catagory = ['Employe will stay', 'Employ will leave']
custom_dt = [[10,200,4,0,0, 0.80, 0.96, 2, 4]]
catagory[int(dt.predict(custom_dt))]
```

```
'Employe will stay'
```

Figure 4.12

K-Nearest Neighbors Algorithm (KNN)

The k-nearest neighbors (KNN) algorithm supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in proximity. In other words, similar things are near to each other. “Birds of a feather flock together.”

```
sc=StandardScaler().fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
```

Figure 4.13

Optimal Number of k:

Now we’ll run the KNN for different values of k and choose k which gives us the highest accuracy.

```
k_range=range(1,26)
scores={}
scores_list=[]


for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_std,Y_train)
    prediction_knn=knn.predict(X_test_std)
    scores[k]=accuracy_score(Y_test,prediction_knn)*100
    scores_list.append(accuracy_score(Y_test,prediction_knn))
```

Figure 4.14(a)

Results from above data are plotted in the below graph (Figure 4.14(b)). We find that k = 1 gives us the highest accuracy.

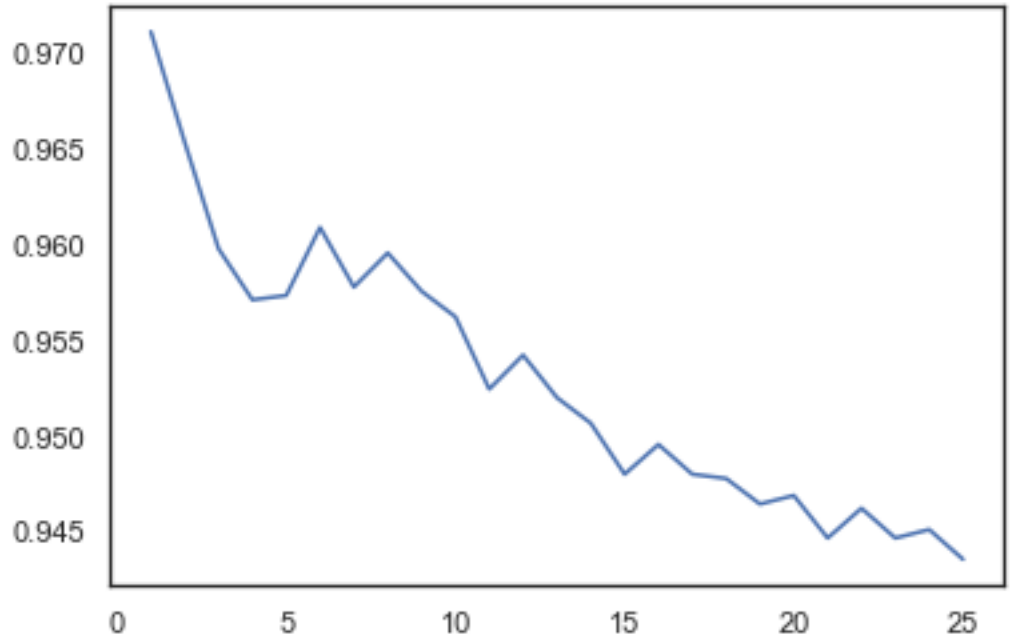

Figure 4.14(b)

Accuracy:

Achieved accuracy is 97.11 %, which is slightly less than decision tree.

```
#using n_neighbors = 1, as it has the highest accuracy
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_std,Y_train)
prediction_knn=knn.predict(X_test_std)
accuracy_knn=accuracy_score(Y_test,prediction_knn)*100
print('Accuracy = ' + str(accuracy_knn))
prediction_knn
```

```
Accuracy = 97.11111111111111
```

Figure 4.15

Random Forest:

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

```
rf = RandomForestClassifier(n_estimators=35,criterion="entropy")
rf.fit(X_train,Y_train)
pred_rf = rf.predict(X_test)
print("Training accuracy for Random Forest is",rf.score(X_train,Y_train)*100)
print("Testing accuracy for Random Forest is",rf.score(X_test,Y_test)*100)
accuracy_rf = rf.score(X_test,Y_test)*100
```

```
Training accuracy for Random Forest is 99.97142585008096
Testing accuracy for Random Forest is 98.82222222222222
```

Figure 4.16

Accuracy:

Accuracy of the random forest algorithm on train data is 99.97% and on test data it is 98.82%.

Confusion Matrix and Classification Report:

```
print("Confusion Matrix of Random Forest \n",confusion_matrix(Y_test,pred_rf))
```

```
Confusion Matrix of Random Forest
 [[3408    7]
 [  46 1039]]
```

```
print("Classification report for Random Forest")
print(classification_report(Y_test,pred_rf))
```

```
Classification report for Random Forest
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3415
           1       0.99      0.96      0.98      1085

    accuracy                           0.99      4500
   macro avg       0.99      0.98      0.98      4500
weighted avg       0.99      0.99      0.99      4500
```

Figure 4.17

We can see that classification report suggests that random forest has the highest accuracy of predicting True Positive and False Positive.

Feature Importance:

```
feature_importances = pd.DataFrame(rf.feature_importances_,
                                   index = features,
                                    columns=['importance']).sort_values('importance',ascending=False)
feature_importances
```

	importance
satisfaction_level	0.343768
time_spend_company	0.189323
number_project	0.152265
average_montly_hours	0.147704
last_evaluation	0.127266
department_numerical	0.019465
salary_numerical	0.011173
Work_accident	0.007819
promotion_last_5years	0.001218

Figure 4.18

Clearly some features like “satisfaction_level”, “time_spend_company” has a higher impact on the retention of the employees. People having a higher satisfaction are more likely to stay in the organization.

Accuracy Comparison of Prediction Algorithms

Random forest has the highest accuracy of all. It is because it makes multiple random decision trees and picks the one with highest mode.

Figure 4.19 shows the accuracy comparison of logistical regression, decision tree, k-nearest neighbors algorithm, random forest algorithm.

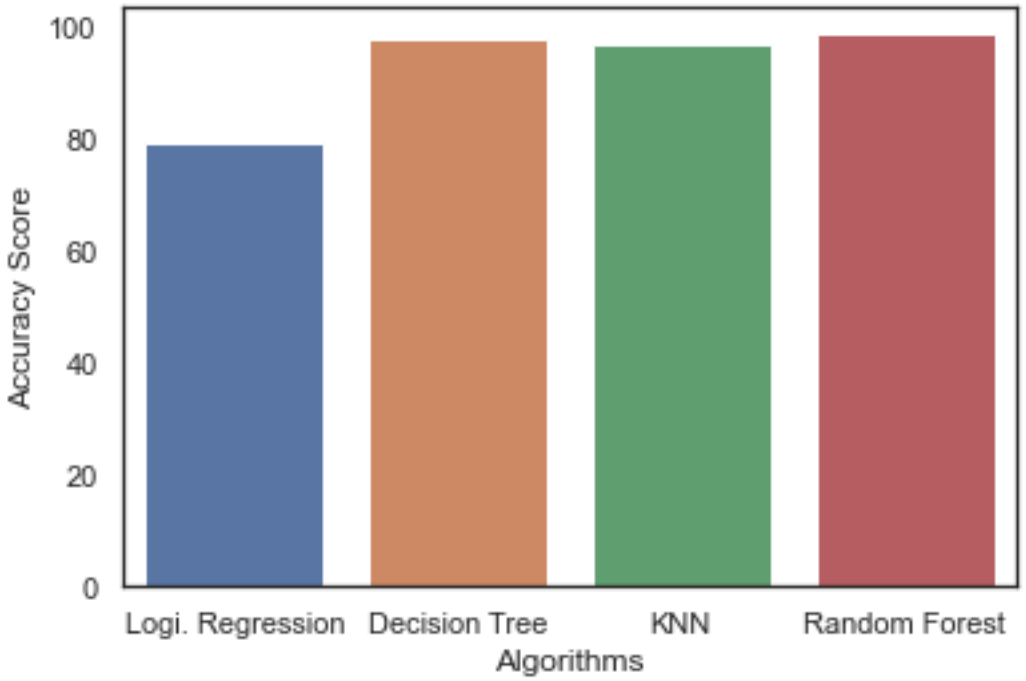

Figure 4.19

PCA Analysis

Principal Component Analysis (PCA) is used to explain the variance-covariance structure of a set of variables through linear combinations. It is often used as a dimensionality-reduction technique. It is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

```
pca = PCA(n_components = 2)
```

```
pca.fit(data)
```

```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

Figure 5.1

Variance and Covariance:

```
pca.explained_variance_ratio_
```

```
array([0.99505374, 0.00328353])
```

```
pca.explained_variance_
```

```
array([2494.61707963,    8.23187396])
```

```
pca.get_covariance
```

```
<bound method _BasePCA.get_covariance of PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)>
```

Figure 5.2

Mean and Score Sample:

```
pca.mean_
```

```
array([6.12833522e-01, 7.16101740e-01, 3.80305354e+00, 2.01050337e+02,
       3.49823322e+00, 1.44609641e-01, 2.12680845e-02, 2.38082539e-01,
       1.59470631e+00, 5.87052470e+00])
```

```
pca.score_samples(data)
```

```
array([-14.88019135, -18.41685095, -19.42760601, ..., -14.9821359 ,
       -16.0049539 , -15.08192061])
```

Naïve Bayes Classifier

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of feature in a class is unrelated to the presence of any other feature. The model was trained using the training datasets generated previously i.e., X_train and Y_train with an accuracy of 79.89% (approximately) while accuracy at testing was 80.13% (approximately). Gaussian is used in classification and assumes that feature follow a normal distribution due to which the accuracy is not at par with other methods. It can be observed from the figure that 2111 points were mislabeled out of 104099 which makes up 14.07% (approximately) of the dataset.

```
gnb = GaussianNB()
```

```
gnb.fit(X_train, Y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Figure 6.1

Mislabeled Points:

Total mislabeled points are 2111, which is 14.07% of the data.

```
gnb.predict_proba(X_test)
```

```
array([[2.91205525e-01, 7.08794475e-01],
       [9.99977719e-01, 2.22813990e-05],
       [1.23530231e-01, 8.76469769e-01],
       ...,
       [1.02947532e-01, 8.97052468e-01],
       [4.03142988e-01, 5.96857012e-01],
       [7.81885015e-01, 2.18114985e-01]])
```

```
print("Number of mislabeled points out of total %d points : %d" %(X_train.shape[0], (Y_train != pred_gnb).sum()))
```

```
Number of mislabeled points out of total 10499 points : 2111
```

Figure 6.2

Accuracy:

Accuracy of the classifier is 80.13 %.

```
print("Accuracy = ", accuracy_gnb*100)
```

```
Accuracy =  80.13333333333334
```

Figure 6.3

Suggestions and Recommendations to HR

1. In Figure 4.18, we see that the most effecting factor to employee retention are satisfaction level, time spend company, number of projects they are working on, average monthly hours they are working.

2. On the other hand, same feature importance suggests that promotion in last 5 years, work accidents, salary, and department doesn't really affect a person's decision to leave the company. (Figure 4.18)

3. In the section Reasons Behind Higher Satisfaction (Figure 1.19), We found that people with lowest number of projects and lowest time spend company, and average monthly work hours have highest satisfaction and people with the greatest number of projects and highest monthly hours tends to lose internal peace.

4. If we look closely in exploratory data analysis, we find that there's a huge difference between people working for 3 years and working for 4 years, meaning most people leave after 3 years. This be a general behavior as people mostly work for 2-3 years in a company and as the experience increases, we people go for more promising opportunities (NY Times).

5. In exploratory data analysis, we can see that people working on 5 or more projects leave the company, it can be said that people who are working for a longer period of time have done the more number of projects.

6. In outlier detection, we see that some people have 1, 8, 9, and 10 projects, these people are termed as outliers. Which means number of projects are not equally divided. HR should divide projects moderately.

7. Promotion in last five years doesn't have linear effect on satisfaction. So instead of promoting people, HR should work on ways to increase employee satisfaction.

8. Salary doesn't seem to have any effect on satisfaction, as people with high, medium or low satisfaction have same average salary. This is quite interesting as salary is believed to have a major effect on employees' satisfaction, and consequently on employee retention.