# Untitled1

June 21, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

[4]:
```
df = pd.read_csv("HR_comma_sep[1].csv")
```

[5]:
```
df
```

[5]:
```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
…                     …                …              …
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2

       average_montly_hours  time_spend_company  Work_accident  left  \
0                       157                   3              0     1
1                       262                   6              0     1
2                       272                   4              0     1
3                       223                   5              0     1
4                       159                   3              0     1
…                        …                   …             …    …
14994                   151                   3              0     1
14995                   160                   3              0     1
14996                   143                   3              0     1
14997                   280                   4              0     1
14998                   158                   3              0     1

       promotion_last_5years    sales  salary
```

```
0                          0    sales     low
1                          0    sales  medium
2                          0    sales  medium
3                          0    sales     low
4                          0    sales     low
…                        …      …     …
14994                      0  support     low
14995                      0  support     low
14996                      0  support     low
14997                      0  support     low
14998                      0  support     low

[14999 rows x 10 columns]
```

[6]:
```
df.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   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   sales                  14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

[7]:
```
df.isna().sum()
```

[7]:
```
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
sales                    0
salary                   0
dtype: int64
```

[8]:
```python
df["left"].unique()
```

[8]:
```
array([1, 0])
```

[9]:
```python
df["promotion_last_5years"].unique()
```

[9]:
```
array([0, 1])
```

[10]:
```python
df["number_project"].unique()
```

[10]:
```
array([2, 5, 7, 6, 4, 3])
```

[11]:
```python
df.satisfaction_level.unique()
```

[11]:
```
array([0.38, 0.8 , 0.11, 0.72, 0.37, 0.41, 0.1 , 0.92, 0.89, 0.42, 0.45,
       0.84, 0.36, 0.78, 0.76, 0.09, 0.46, 0.4 , 0.82, 0.87, 0.57, 0.43,
       0.13, 0.44, 0.39, 0.85, 0.81, 0.9 , 0.74, 0.79, 0.17, 0.24, 0.91,
       0.71, 0.86, 0.14, 0.75, 0.7 , 0.31, 0.73, 0.83, 0.32, 0.54, 0.27,
       0.77, 0.88, 0.48, 0.19, 0.6 , 0.12, 0.61, 0.33, 0.56, 0.47, 0.28,
       0.55, 0.53, 0.59, 0.66, 0.25, 0.34, 0.58, 0.51, 0.35, 0.64, 0.5 ,
       0.23, 0.15, 0.49, 0.3 , 0.63, 0.21, 0.62, 0.29, 0.2 , 0.16, 0.65,
       0.68, 0.67, 0.22, 0.26, 0.99, 0.98, 1.  , 0.52, 0.93, 0.97, 0.69,
       0.94, 0.96, 0.18, 0.95])
```

[12]:
```python
df.last_evaluation.unique()
```

[12]:
```
array([0.53, 0.86, 0.88, 0.87, 0.52, 0.5 , 0.77, 0.85, 1.  , 0.54, 0.81,
       0.92, 0.55, 0.56, 0.47, 0.99, 0.51, 0.89, 0.83, 0.95, 0.57, 0.49,
       0.46, 0.62, 0.94, 0.48, 0.8 , 0.74, 0.7 , 0.78, 0.91, 0.93, 0.98,
       0.97, 0.79, 0.59, 0.84, 0.45, 0.96, 0.68, 0.82, 0.9 , 0.71, 0.6 ,
       0.65, 0.58, 0.72, 0.67, 0.75, 0.73, 0.63, 0.61, 0.76, 0.66, 0.69,
       0.37, 0.64, 0.39, 0.41, 0.43, 0.44, 0.36, 0.38, 0.4 , 0.42])
```

[13]:
```python
df.time_spend_company.unique()
```

[13]:
```
array([ 3,  6,  4,  5,  2,  8, 10,  7])
```

[14]:
```python
df.Work_accident.unique()
```

[14]:
```
array([0, 1])
```

[15]:
```python
df.sales.unique()
```

[15]:
```
array(['sales', 'accounting', 'hr', 'technical', 'support', 'management',
       'IT', 'product_mng', 'marketing', 'RandD'], dtype=object)
```

[16]:
```python
df.salary.unique()
```

[16]:
```
array(['low', 'medium', 'high'], dtype=object)
```

[17]:
```
df.corr()
```

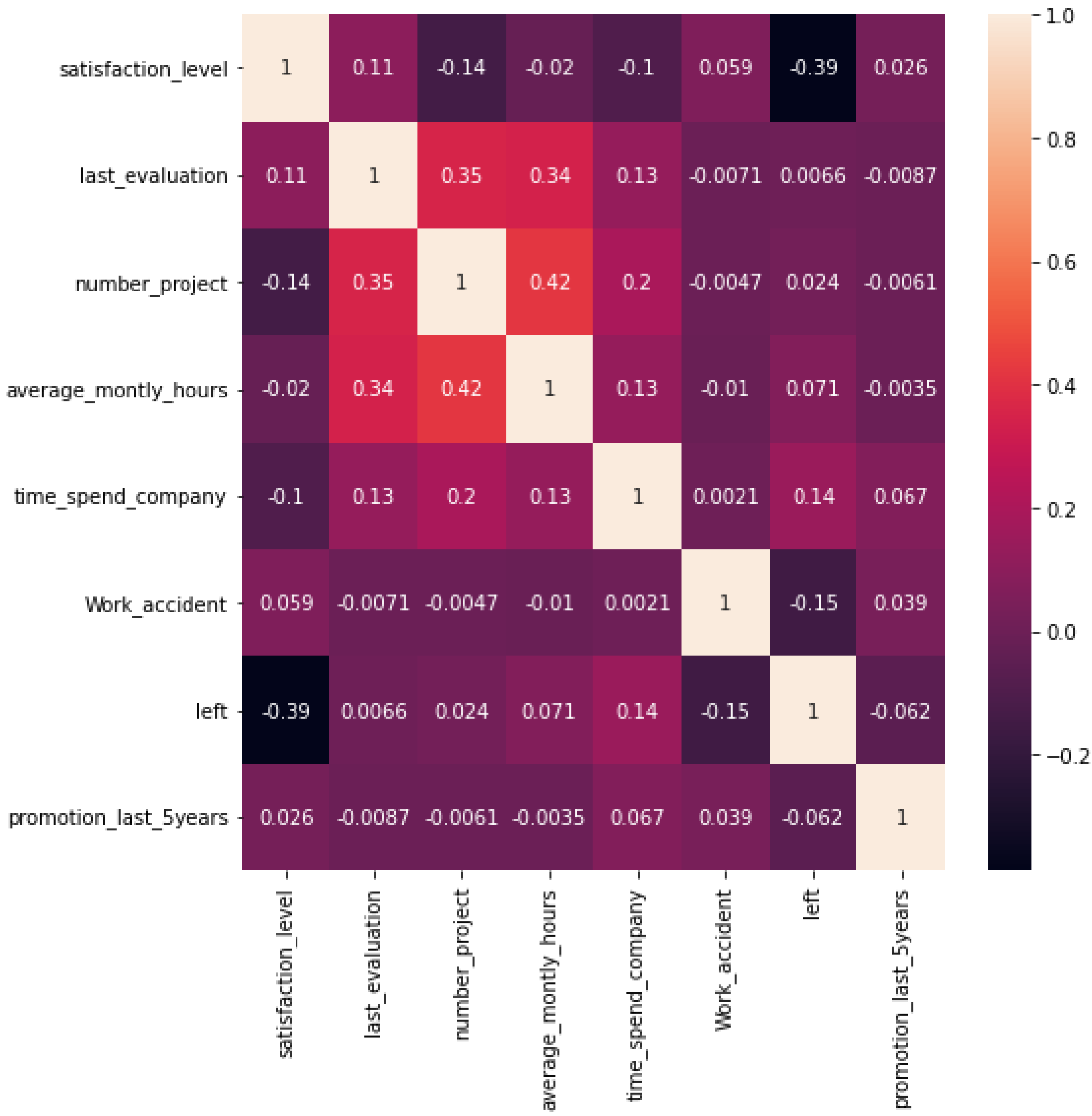
```
/tmp/ipykernel_72/1134722465.py:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.corr()
```

[17]:
```
                       satisfaction_level  last_evaluation  number_project  \
satisfaction_level               1.000000         0.105021       -0.142970
last_evaluation                  0.105021         1.000000        0.349333
number_project                  -0.142970         0.349333        1.000000
average_montly_hours            -0.020048         0.339742        0.417211
time_spend_company              -0.100866         0.131591        0.196786
Work_accident                    0.058697        -0.007104       -0.004741
left                            -0.388375         0.006567        0.023787
promotion_last_5years            0.025605        -0.008684       -0.006064

                       average_montly_hours  time_spend_company  \
satisfaction_level                -0.020048           -0.100866
last_evaluation                    0.339742            0.131591
number_project                     0.417211            0.196786
average_montly_hours               1.000000            0.127755
time_spend_company                 0.127755            1.000000
Work_accident                     -0.010143            0.002120
left                               0.071287            0.144822
promotion_last_5years             -0.003544            0.067433

                       Work_accident      left  promotion_last_5years
satisfaction_level          0.058697 -0.388375               0.025605
last_evaluation            -0.007104  0.006567              -0.008684
number_project             -0.004741  0.023787              -0.006064
average_montly_hours       -0.010143  0.071287              -0.003544
time_spend_company          0.002120  0.144822               0.067433
Work_accident               1.000000 -0.154622               0.039245
left                       -0.154622  1.000000              -0.061788
promotion_last_5years       0.039245 -0.061788               1.000000
```

[18]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)
```

```
/tmp/ipykernel_72/609742482.py:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

[18]: <AxesSubplot: >




```
[19]: df1= df.groupby(["sales"])["left"].value_counts().reset_index(name="count")
df1=pd.DataFrame(df1)
```

```
[20]: df["sales"].value_counts()
```

```
[20]: sales          4140
technical      2720
support        2229
IT             1227
product_mng     902
marketing       858
```

```
RandD           787
accounting      767
hr              739
management      630
Name: sales, dtype: int64
```

[21]:
```
dft=df["sales"].value_counts().reset_index(name="Total")
```

[22]:
```
dft=dft.rename(columns={"index":"sales"})
```

[23]:
```
dft
```

[23]:

| | sales | Total |
|---|---|---|
| 0 | sales | 4140 |
| 1 | technical | 2720 |
| 2 | support | 2229 |
| 3 | IT | 1227 |
| 4 | product_mng | 902 |
| 5 | marketing | 858 |
| 6 | RandD | 787 |
| 7 | accounting | 767 |
| 8 | hr | 739 |
| 9 | management | 630 |

[24]:
```
dfmer=df1.merge(dft,how="left")
```

[25]:
```
dfmer
```

[25]:

| | sales | left | count | Total |
|---|---|---|---|---|
| 0 | IT | 0 | 954 | 1227 |
| 1 | IT | 1 | 273 | 1227 |
| 2 | RandD | 0 | 666 | 787 |
| 3 | RandD | 1 | 121 | 787 |
| 4 | accounting | 0 | 563 | 767 |
| 5 | accounting | 1 | 204 | 767 |
| 6 | hr | 0 | 524 | 739 |
| 7 | hr | 1 | 215 | 739 |
| 8 | management | 0 | 539 | 630 |
| 9 | management | 1 | 91 | 630 |
| 10 | marketing | 0 | 655 | 858 |
| 11 | marketing | 1 | 203 | 858 |
| 12 | product_mng | 0 | 704 | 902 |
| 13 | product_mng | 1 | 198 | 902 |
| 14 | sales | 0 | 3126 | 4140 |
| 15 | sales | 1 | 1014 | 4140 |
| 16 | support | 0 | 1674 | 2229 |
| 17 | support | 1 | 555 | 2229 |

```
18     technical     0    2023    2720
19     technical     1     697    2720
```

[26]:
```python
dfmer["normal"]=dfmer["count"].div(dfmer["Total"].values)
dfmer["normal"]=dfmer["normal"]*100
```

[27]:
```python
dfmer
```

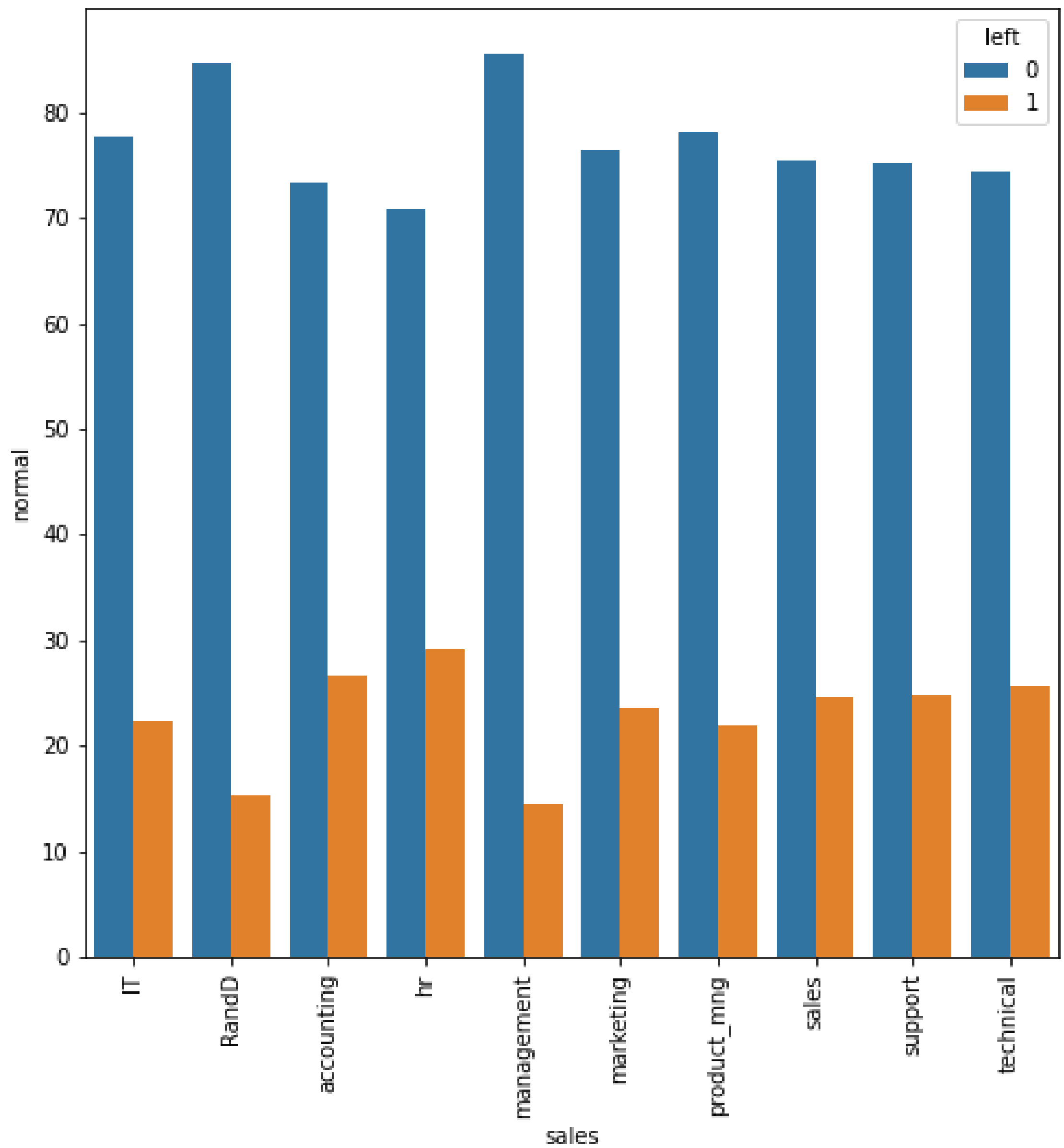
[27]:

| | sales | left | count | Total | normal |
|---|---|---|---|---|---|
| 0 | IT | 0 | 954 | 1227 | 77.750611 |
| 1 | IT | 1 | 273 | 1227 | 22.249389 |
| 2 | RandD | 0 | 666 | 787 | 84.625159 |
| 3 | RandD | 1 | 121 | 787 | 15.374841 |
| 4 | accounting | 0 | 563 | 767 | 73.402868 |
| 5 | accounting | 1 | 204 | 767 | 26.597132 |
| 6 | hr | 0 | 524 | 739 | 70.906631 |
| 7 | hr | 1 | 215 | 739 | 29.093369 |
| 8 | management | 0 | 539 | 630 | 85.555556 |
| 9 | management | 1 | 91 | 630 | 14.444444 |
| 10 | marketing | 0 | 655 | 858 | 76.340326 |
| 11 | marketing | 1 | 203 | 858 | 23.659674 |
| 12 | product_mng | 0 | 704 | 902 | 78.048780 |
| 13 | product_mng | 1 | 198 | 902 | 21.951220 |
| 14 | sales | 0 | 3126 | 4140 | 75.507246 |
| 15 | sales | 1 | 1014 | 4140 | 24.492754 |
| 16 | support | 0 | 1674 | 2229 | 75.100942 |
| 17 | support | 1 | 555 | 2229 | 24.899058 |
| 18 | technical | 0 | 2023 | 2720 | 74.375000 |
| 19 | technical | 1 | 697 | 2720 | 25.625000 |

[28]:
```python
plt.figure(figsize=(8,8))
sns.barplot(x="sales",y='normal',hue="left",data=dfmer)
plt.xticks(rotation=90)

#People from the hr department are leaving the highest based on the normalized␣
↪data.The Hr department has the highest percentage. Normal = (Count of people␣
↪from leaving category in a department)/(Total number of people in that␣
↪department)*100
```

[28]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'IT'),
  Text(1, 0, 'RandD'),
  Text(2, 0, 'accounting'),
  Text(3, 0, 'hr'),
  Text(4, 0, 'management'),
  Text(5, 0, 'marketing'),
  Text(6, 0, 'product_mng'),
```

```
Text(7, 0, 'sales'),
Text(8, 0, 'support'),
Text(9, 0, 'technical')])
```




[29]:
```
df1.head()
```

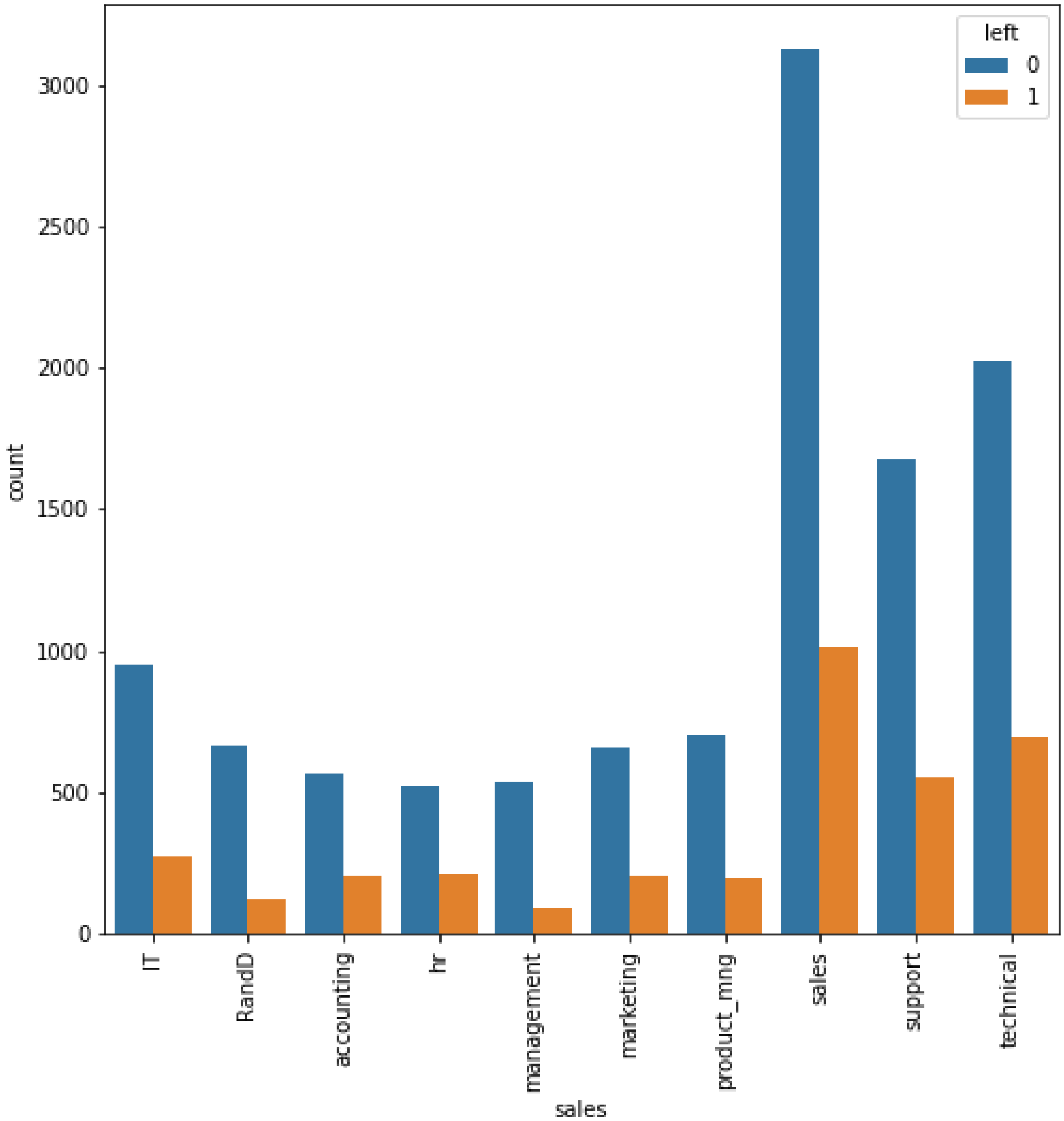
[29]:

| | sales | left | count |
|---|---|---|---|
| 0 | IT | 0 | 954 |
| 1 | IT | 1 | 273 |
| 2 | RandD | 0 | 666 |

```
3        RandD     1    121
4   accounting     0    563
```

[30]:
```
plt.figure(figsize=(8,8))
sns.barplot(x="sales",y='count',hue="left",data=df1)
plt.xticks(rotation=90)

#The people from the sales department are leaing the highest if we look at only␣
 ↪the count of leaving people.
```

[30]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'IT'),
  Text(1, 0, 'RandD'),
  Text(2, 0, 'accounting'),
  Text(3, 0, 'hr'),
  Text(4, 0, 'management'),
  Text(5, 0, 'marketing'),
  Text(6, 0, 'product_mng'),
  Text(7, 0, 'sales'),
  Text(8, 0, 'support'),
  Text(9, 0, 'technical')])
```




```
[31]: df2= df.groupby(["salary"])["left"].value_counts().reset_index(name="count")
      df2=pd.DataFrame(df2)
```

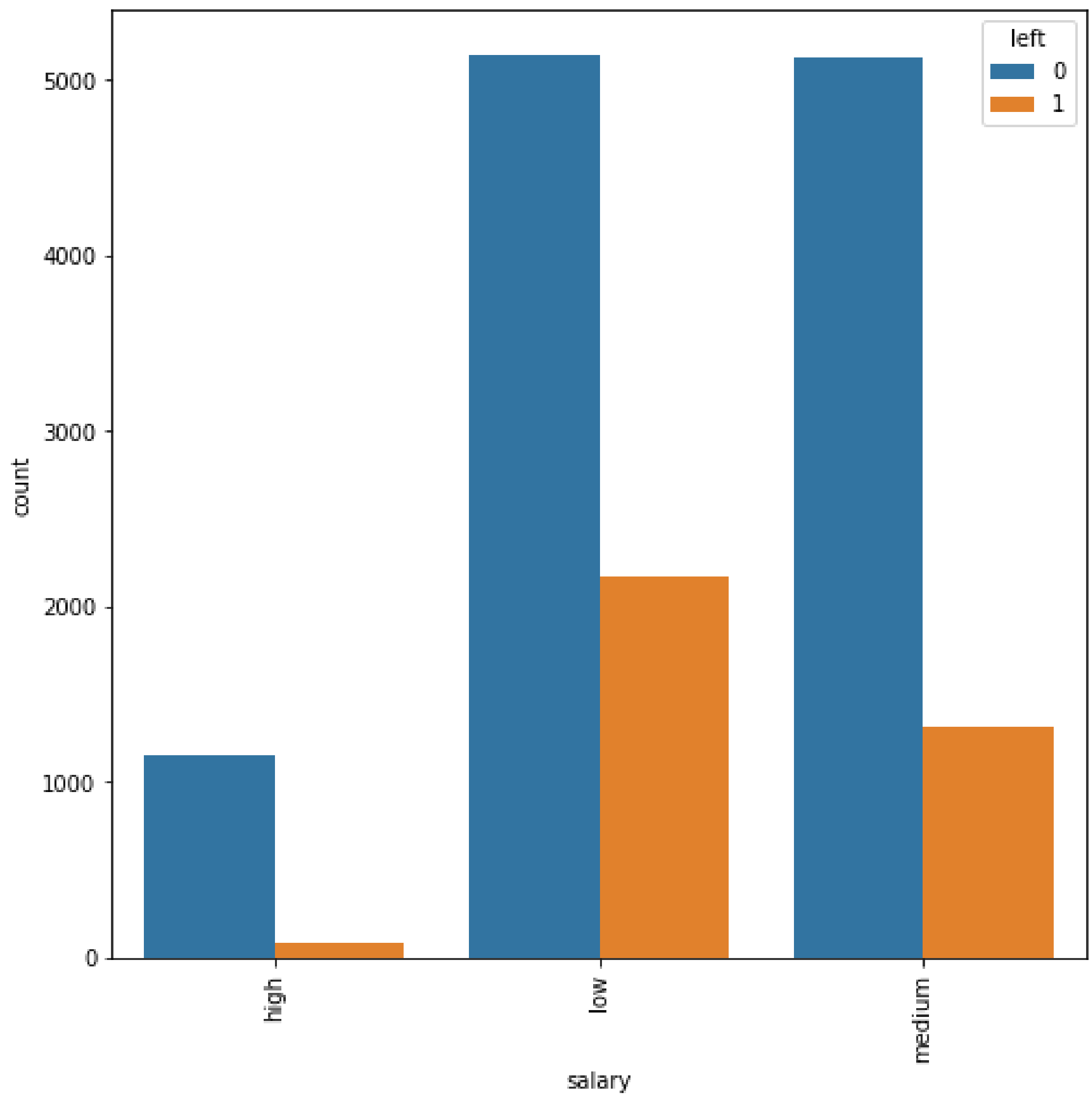
```
[32]: df2.head()
```

[32]:

| | salary | left | count |
|---|---|---|---|
| 0 | high | 0 | 1155 |
| 1 | high | 1 | 82 |
| 2 | low | 0 | 5144 |
| 3 | low | 1 | 2172 |
| 4 | medium | 0 | 5129 |

[33]:
```
plt.figure(figsize=(8,8))
sns.barplot(x="salary",y='count',hue="left",data=df2)
plt.xticks(rotation=90)

#People with Lower Salaries are leaving the company
```

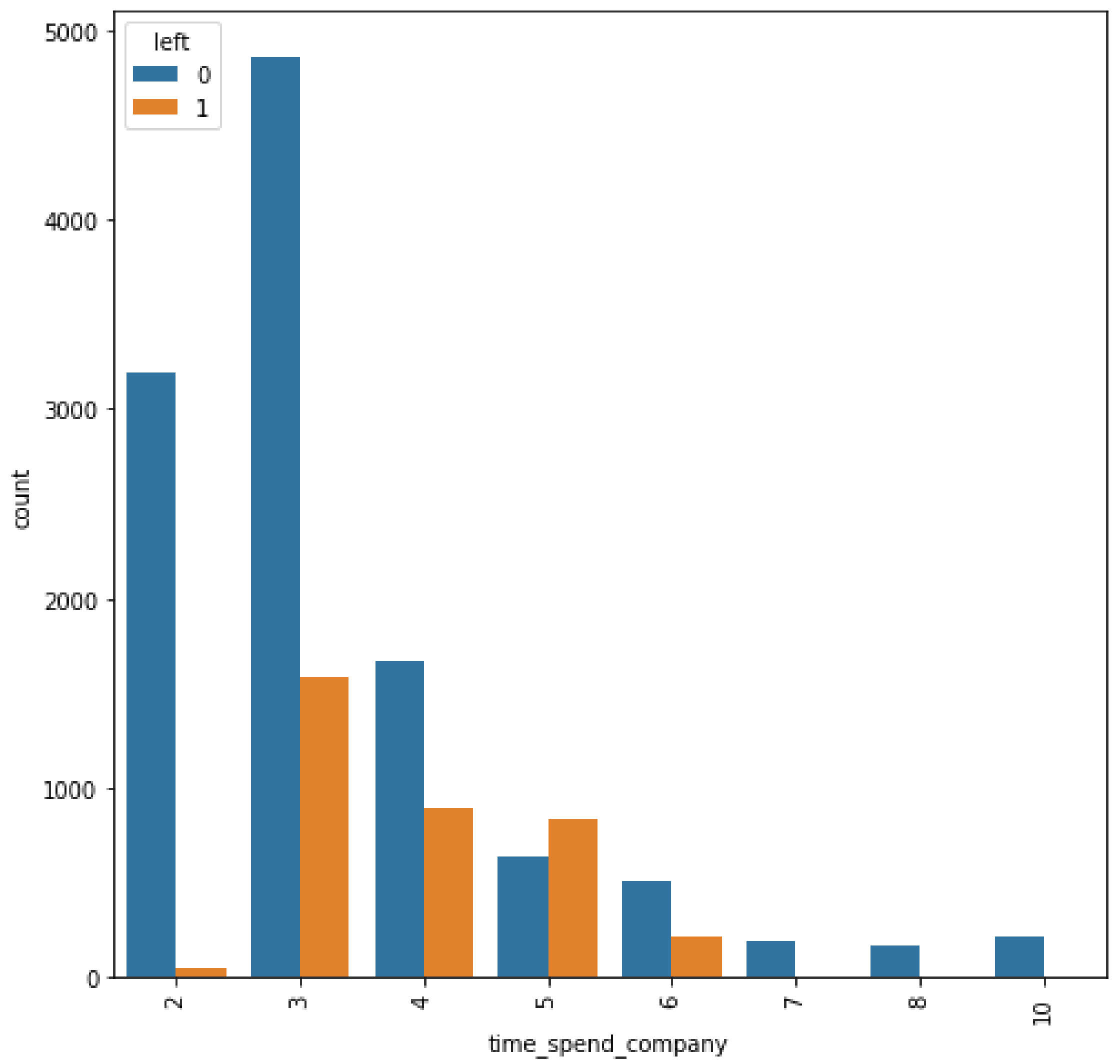
[33]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'high'), Text(1, 0, 'low'), Text(2, 0, 'medium')])
```




[34]:
```
df3= df.groupby(["time_spend_company"])["left"].value_counts().
 ↪reset_index(name="count")
df3=pd.DataFrame(df3)
```

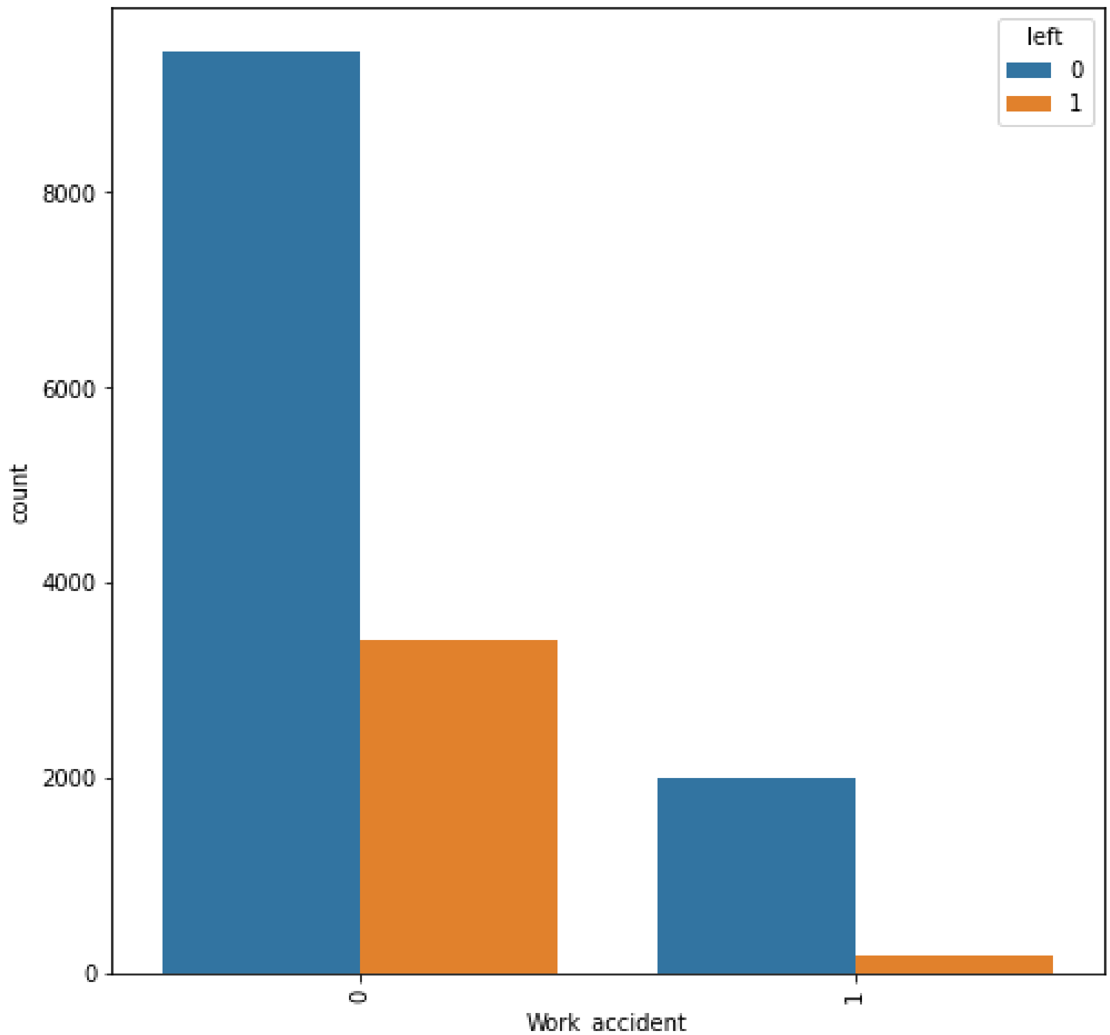
[35]:
```
#time_spend_company
plt.figure(figsize=(8,8))
sns.barplot(x="time_spend_company",y='count',hue="left",data=df3)
plt.xticks(rotation=90)
#People with experience of 3 to 5 years are leaving the comapny more.
```

[35]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7]),
 [Text(0, 0, '2'),
  Text(1, 0, '3'),
  Text(2, 0, '4'),
  Text(3, 0, '5'),
  Text(4, 0, '6'),
  Text(5, 0, '7'),
  Text(6, 0, '8'),
  Text(7, 0, '10')])
```



[37]:

```
# Set the figure size
plt.figure(figsize=(8, 8))

# Create the countplot
sns.countplot(x="Work_accident", hue="left", data=df)

# Rotate x-axis labels
plt.xticks(rotation=90)

# Display the plot
plt.show()
```



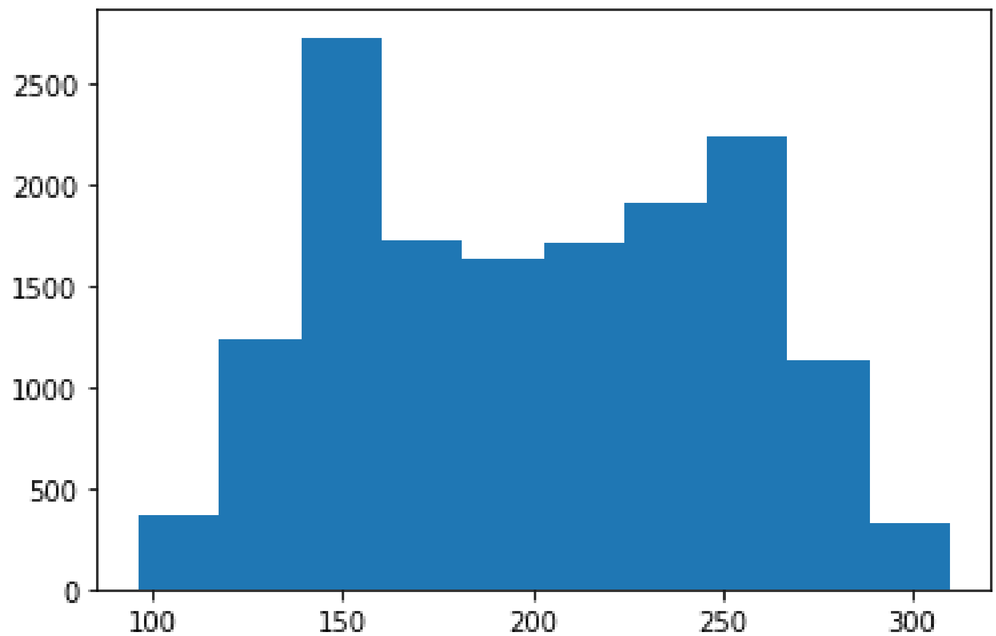

[38]:

```
df.columns
```

[38]: Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'sales', 'salary'],
      dtype='object')

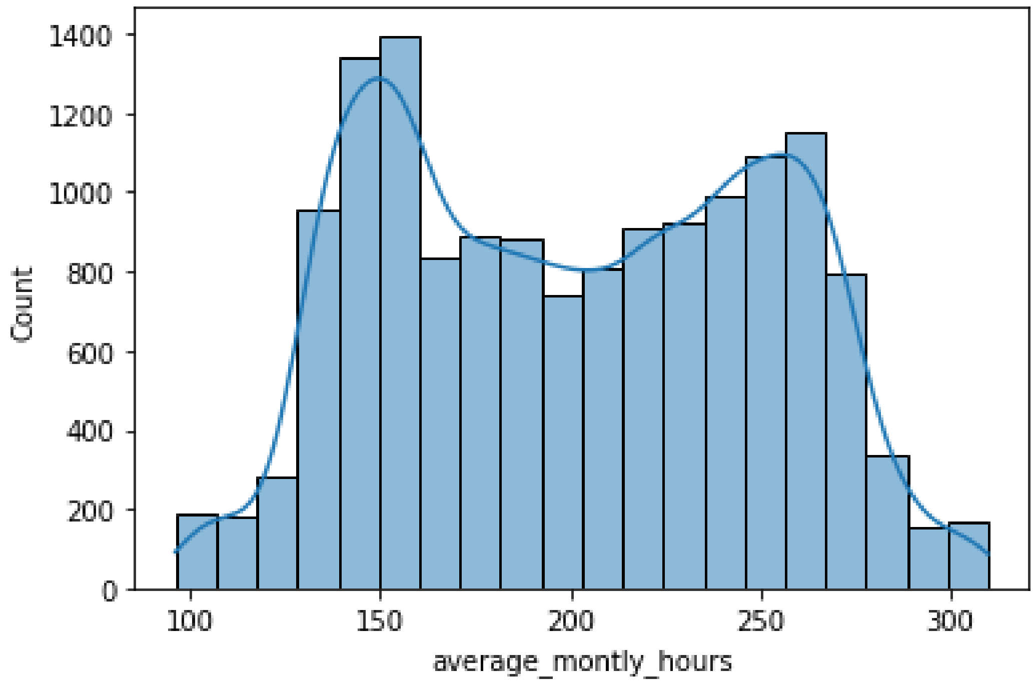
[39]:
```
plt.hist(df["average_montly_hours"])
```

[39]: (array([ 367., 1240., 2733., 1722., 1628., 1712., 1906., 2240., 1127.,
         324.]),
 array([ 96. , 117.4, 138.8, 160.2, 181.6, 203. , 224.4, 245.8, 267.2,
        288.6, 310. ]),
 <BarContainer object of 10 artists>)



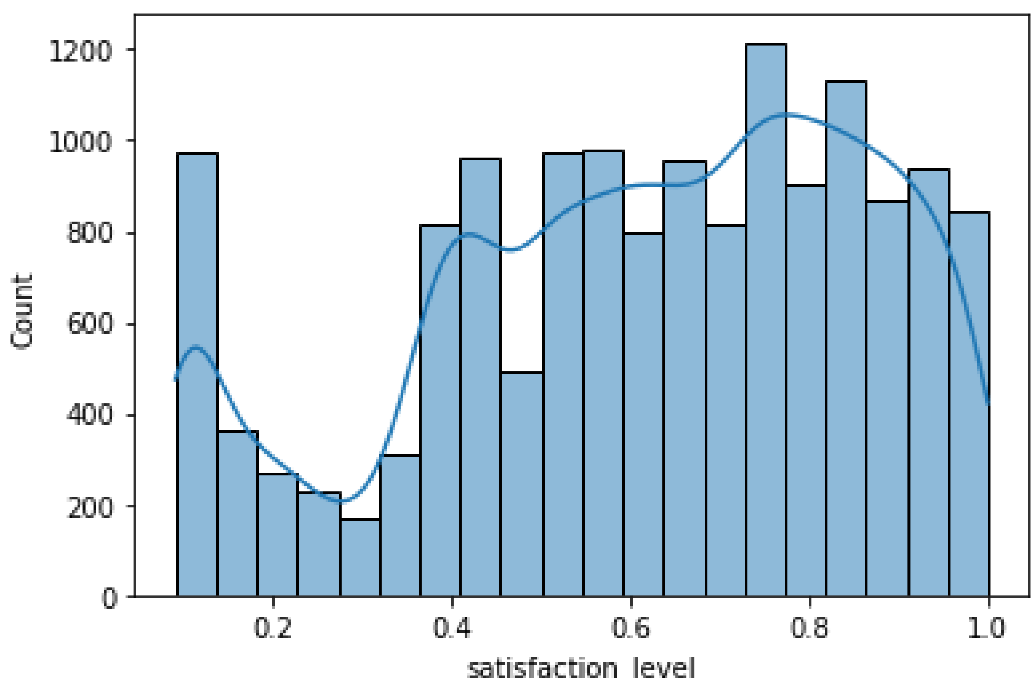

[40]:
```
sns.histplot(data = df,x="average_montly_hours", kde = True,bins=20)
```

[40]: <AxesSubplot: xlabel='average_montly_hours', ylabel='Count'>



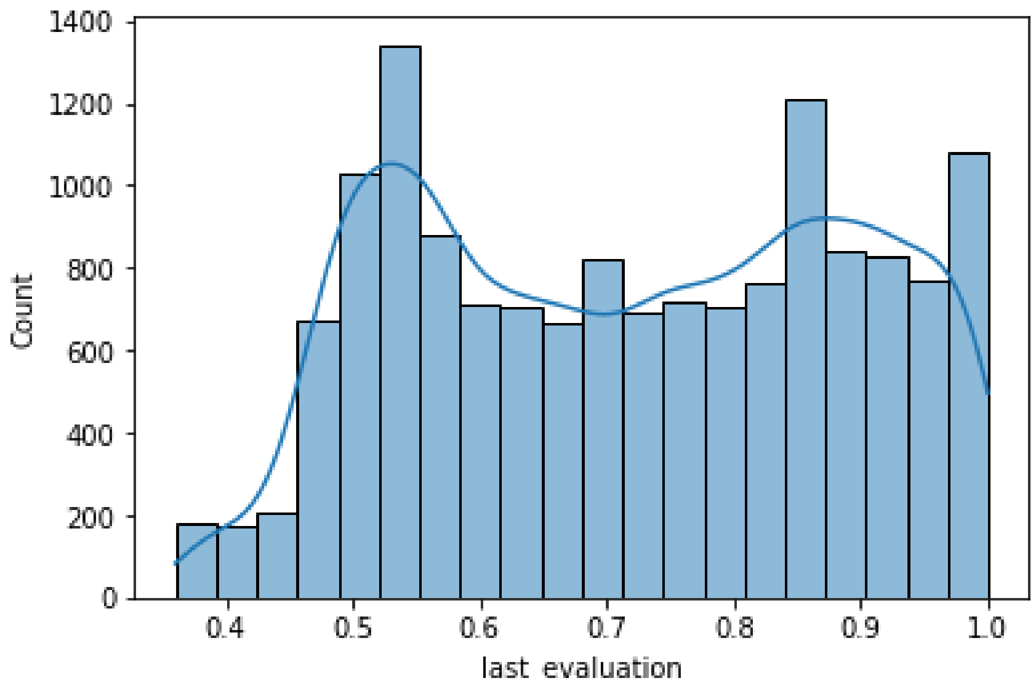

```
[41]: sns.histplot(data = df,x="satisfaction_level", kde = True,bins=20)
```

[41]: <AxesSubplot: xlabel='satisfaction_level', ylabel='Count'>



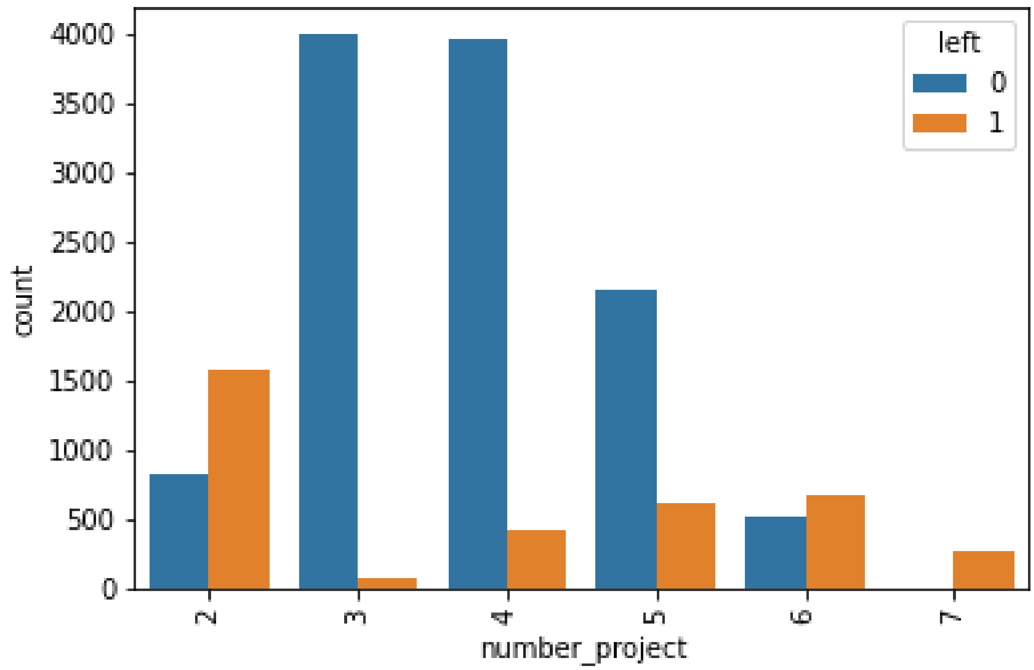
[42]:
```
sns.histplot(data = df,x="last_evaluation", kde = True,bins=20)
```

[42]: <AxesSubplot: xlabel='last_evaluation', ylabel='Count'>




[43]:
```
sns.countplot(x="number_project",hue="left",data=df)
plt.xticks(rotation=90)
#People who have worked on 3 or 4 projects have left the organisation more.
```

[43]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, '2'),
  Text(1, 0, '3'),
  Text(2, 0, '4'),
  Text(3, 0, '5'),
  Text(4, 0, '6'),
  Text(5, 0, '7')])
```




```
[44]: dfclus = df[["satisfaction_level","last_evaluation","left"]]
```

```
[45]: dfclus
```

```
[45]:        satisfaction_level  last_evaluation  left
0                      0.38             0.53     1
1                      0.80             0.86     1
2                      0.11             0.88     1
3                      0.72             0.87     1
4                      0.37             0.52     1
…                       …               …    …
14994                  0.40             0.57     1
14995                  0.37             0.48     1
14996                  0.37             0.53     1
14997                  0.11             0.96     1
14998                  0.37             0.52     1

[14999 rows x 3 columns]
```

```
[46]: from sklearn.cluster import KMeans
```

```
[47]: km=dfclus.iloc[:,:].values
kmeans = KMeans(n_clusters=3, random_state=0)
label = kmeans.fit_predict(dfclus)
labelarr = kmeans.fit_predict(km)
```

[48]:
```
label
```

[48]:
```
array([1, 1, 1, …, 1, 1, 1], dtype=int32)
```

[49]:
```
dfclus[label==0].describe()
```

[49]:
```
       satisfaction_level  last_evaluation    left
count         4596.000000      4596.000000  4596.0
mean             0.453814         0.679569     0.0
std              0.152887         0.165613     0.0
min              0.120000         0.360000     0.0
25%              0.340000         0.550000     0.0
50%              0.510000         0.670000     0.0
75%              0.570000         0.810000     0.0
max              0.690000         1.000000     0.0
```

[50]:
```
dfclus[label==1].describe()
```

[50]:
```
       satisfaction_level  last_evaluation    left
count         3571.000000      3571.000000  3571.0
mean             0.440098         0.718113     1.0
std              0.263933         0.197673     0.0
min              0.090000         0.450000     1.0
25%              0.130000         0.520000     1.0
50%              0.410000         0.790000     1.0
75%              0.730000         0.900000     1.0
max              0.920000         1.000000     1.0
```

[51]:
```
dfclus[label==2].describe()
```

[51]:
```
       satisfaction_level  last_evaluation    left
count         6832.000000      6832.000000  6832.0
mean             0.810095         0.739627     0.0
std              0.109845         0.154931     0.0
min              0.590000         0.360000     0.0
25%              0.720000         0.610000     0.0
50%              0.810000         0.740000     0.0
75%              0.910000         0.870000     0.0
max              1.000000         1.000000     0.0
```

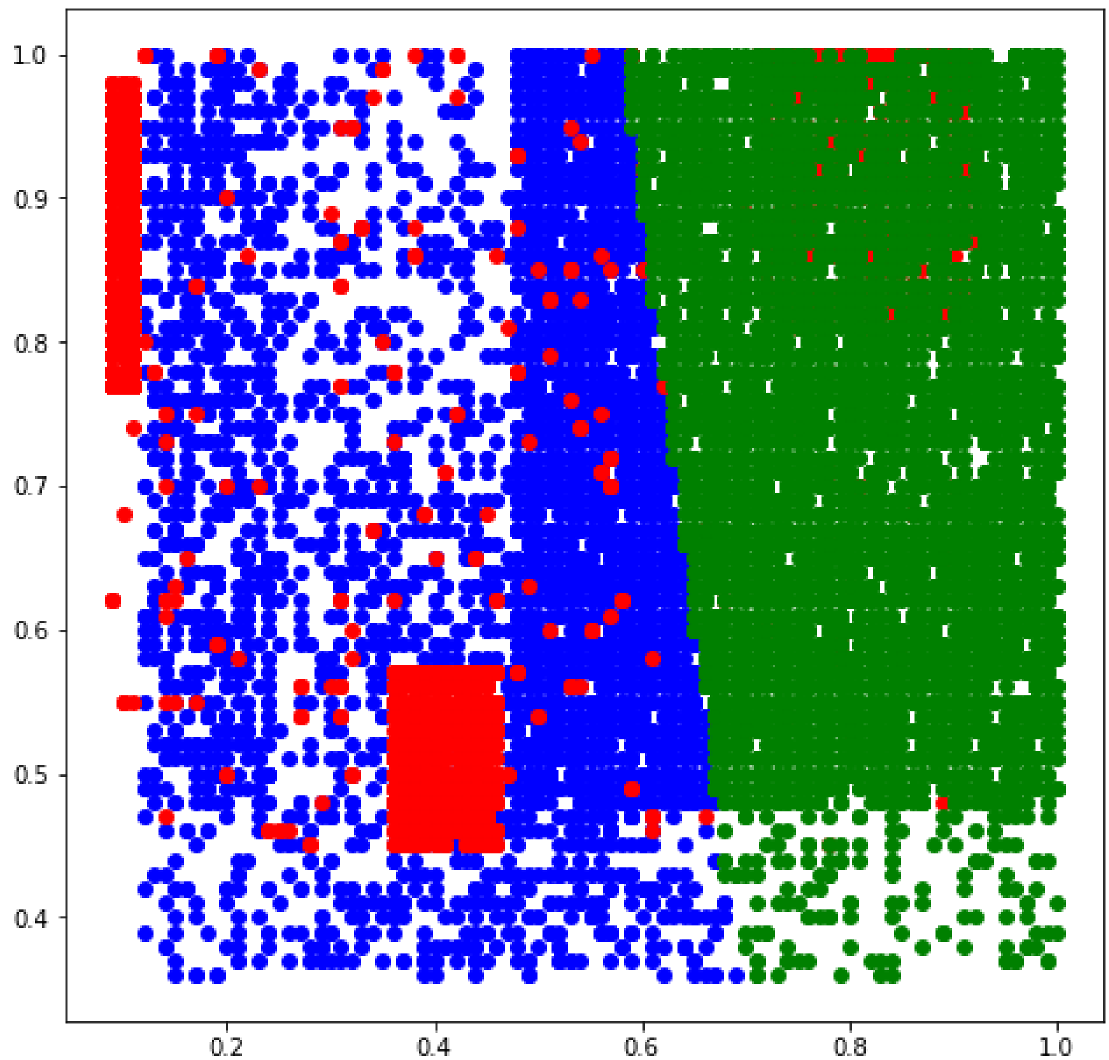
[52]:
```
km[label==0,1]
```

[52]:
```
array([0.74, 0.69, 0.6 , …, 0.94, 0.65, 0.73])
```

[53]:
```
plt.figure(figsize=(8,8))
plt.scatter(km[label==0,0],km[label==0,1],color="blue")
plt.scatter(km[label==1,0],km[label==1,1],color="red")
```

```
plt.scatter(km[label==2,0],km[label==2,1],color="green")

#The Blue cluster denotes people with best satisfaction levels and scored high
 ↪in the last evaluation.

#The Red cluster denotes people with medium satisfaction levels and scored
 ↪average to high in the last evaluation

#The green cluster denotes people with lower satisfaction levels and scored
 ↪fairly than the above mentioned clusters.
```

[53]: <matplotlib.collections.PathCollection at 0x7f8f778f37f0>




[54]:
```
df.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   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   sales                  14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

[55]:
```
df_numerical=df.select_dtypes(include=['int64','float64'])
df_categorical=df.select_dtypes(include=['object'])
```

[56]:
```
#df = pd.get_dummies(data=df,columns=['sales','salary'])
df_converted = pd.get_dummies(data=df_categorical)
```

[58]:
```
df_converted.head()
```

[58]:
```
   sales_IT  sales_RandD  sales_accounting  sales_hr  sales_management  \
0         0            0                 0         0                 0
1         0            0                 0         0                 0
2         0            0                 0         0                 0
3         0            0                 0         0                 0
4         0            0                 0         0                 0

   sales_marketing  sales_product_mng  sales_sales  sales_support  \
0                0                  0            1              0
1                0                  0            1              0
2                0                  0            1              0
3                0                  0            1              0
4                0                  0            1              0

   sales_technical  salary_high  salary_low  salary_medium
0                0            0           1              0
1                0            0           0              1
2                0            0           0              1
3                0            0           1              0
4                0            0           1              0
```

[59]:
```
dfn = pd.concat([df_numerical, df_converted], axis=1, join="inner")
```

[60]:
```
dfn.shape
```

[60]:
```
(14999, 21)
```

[61]:
```
dfn.head()
```

[61]:
```
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157
1                0.80             0.86               5                   262
2                0.11             0.88               7                   272
3                0.72             0.87               5                   223
4                0.37             0.52               2                   159

   time_spend_company  Work_accident  left  promotion_last_5years  sales_IT  \
0                   3              0     1                      0         0
1                   6              0     1                      0         0
2                   4              0     1                      0         0
3                   5              0     1                      0         0
4                   3              0     1                      0         0

   sales_RandD  …  sales_hr  sales_management  sales_marketing  \
0            0  …         0                 0                0
1            0  …         0                 0                0
2            0  …         0                 0                0
3            0  …         0                 0                0
4            0  …         0                 0                0

   sales_product_mng  sales_sales  sales_support  sales_technical  \
0                  0            1              0                0
1                  0            1              0                0
2                  0            1              0                0
3                  0            1              0                0
4                  0            1              0                0

   salary_high  salary_low  salary_medium
0            0           1              0
1            0           0              1
2            0           0              1
3            0           1              0
4            0           1              0

[5 rows x 21 columns]
```

[62]:
```
x =dfn.drop("left",axis=1)
y = dfn["left"]
```

[63]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=123)
```

[64]:
```
xtrain.shape,ytrain.shape,xtest.shape,ytest.shape
```

[64]:
```
((11999, 20), (11999,), (3000, 20), (3000,))
```

[65]:
```
ytrain.value_counts()
```

[65]:
```
0    9137
1    2862
Name: left, dtype: int64
```

[68]:
```
from imblearn.over_sampling import SMOTE
```

[69]:
```
sm = SMOTE(random_state = 2)
xtrainres, ytrainres = sm.fit_resample(xtrain, ytrain)
```

[70]:
```
ytrainres.value_counts()
```

[70]:
```
0    9137
1    9137
Name: left, dtype: int64
```

[71]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
import sklearn.metrics as metrics
```

[72]:
```
logreg = LogisticRegression(solver='lbfgs', max_iter=10000)
```

[73]:
```
print(cross_val_score(logreg, xtrainres, ytrainres, cv=5).mean())
```

```
0.8061195608957294
```

[74]:
```
logreg.fit(xtrainres,ytrainres)
ypred = logreg.predict(xtest)
```

[75]:
```
from sklearn.metrics import classification_report
```

[76]:
```
metrics.confusion_matrix(ytest,ypred)
```

[76]:
```
array([[1831,  460],
       [ 228,  481]])
```

[77]:
```
print(classification_report(ytest,ypred))
```

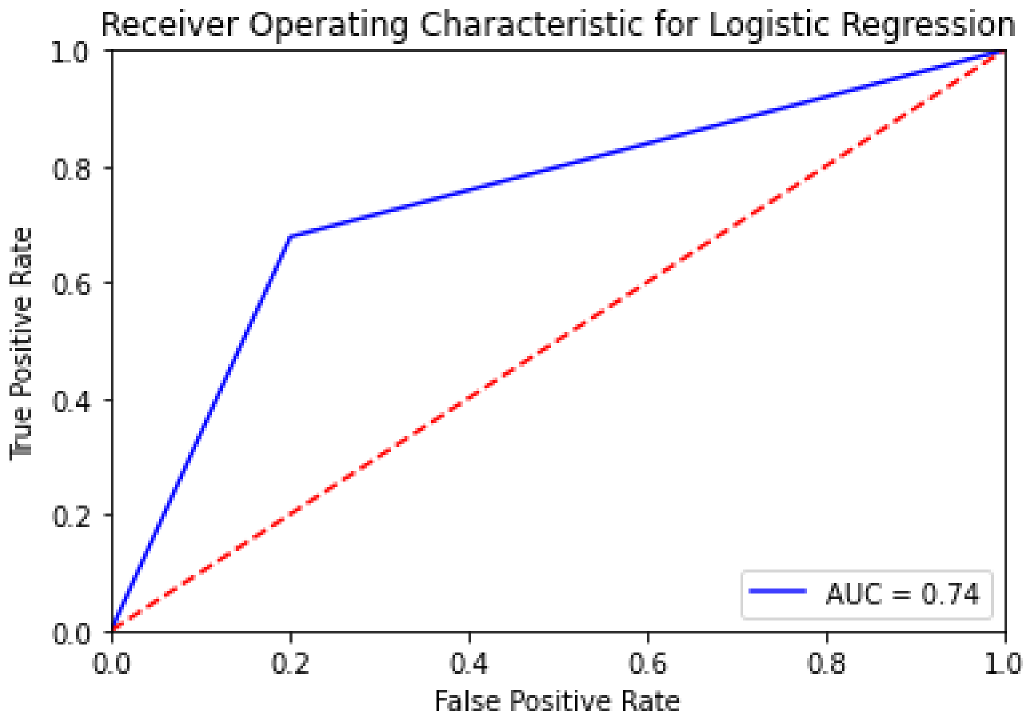
```
              precision    recall  f1-score   support

           0       0.89      0.80      0.84      2291
           1       0.51      0.68      0.58       709

    accuracy                           0.77      3000
   macro avg       0.70      0.74      0.71      3000
weighted avg       0.80      0.77      0.78      3000
```

[78]:
```
roc_auc_score(ytest,ypred)
```

[78]: 0.7388173135941893

[79]:
```
fpr, tpr, threshold = metrics.roc_curve(ytest, ypred)
print(fpr)
print(tpr)
print(threshold)
roc_auc = metrics.auc(fpr, tpr)
print(roc_auc)

# method I: plt
plt.title('Receiver Operating Characteristic for Logistic Regression')
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()
```

```
[0.         0.20078568 1.        ]
[0.         0.67842031 1.        ]
[inf  1.  0.]
0.7388173135941893
```




[80]:
```
randm=RandomForestClassifier(max_depth=5)
```

[81]:
```
print(cross_val_score(randm, xtrainres, ytrainres, cv=5).mean())
```

```
0.9476852531977773
```

[82]:
```
randm.fit(xtrainres,ytrainres)
ypred1=randm.predict(xtest)
```

[83]:
```
metrics.confusion_matrix(ytest,ypred1)
```

[83]:
```
array([[2216,   75],
       [  59,  650]])
```

[84]:
```
print(classification_report(ytest,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2291
           1       0.90      0.92      0.91       709

    accuracy                           0.96      3000
   macro avg       0.94      0.94      0.94      3000
```

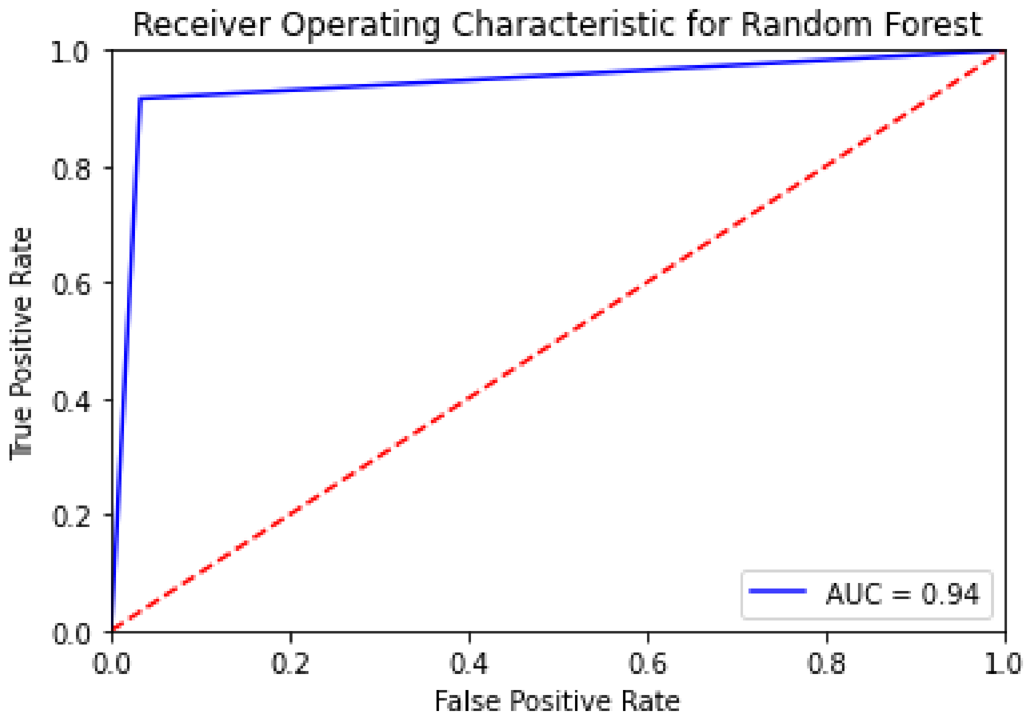
```
weighted avg       0.96      0.96      0.96      3000
```

[85]:
```
roc_auc_score(ytest,ypred1)
```

[85]:
```
0.9420237034720396
```

[86]:
```
fpr, tpr, threshold = metrics.roc_curve(ytest, ypred1)
print(fpr)
print(tpr)
print(threshold)
roc_auc = metrics.auc(fpr, tpr)
print(roc_auc)

# method I: plt
plt.title('Receiver Operating Characteristic for Random Forest')
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()
```

```
[0.        0.0327368 1.       ]
[0.        0.9167842 1.       ]
[inf  1.  0.]
0.9420237034720396
```




[87]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

[88]:
```
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=1.
 ↪0,max_depth=1, random_state=0)
```

[89]:
```
print(cross_val_score(gb, xtrainres, ytrainres, cv=5).mean())
```

```
0.9478495915875037
```

[90]:
```
gb.fit(xtrainres,ytrainres)
```

[90]:
```
GradientBoostingClassifier(learning_rate=1.0, max_depth=1, random_state=0)
```

[91]:
```
ypred2 = gb.predict(xtest)
```

[92]:
```
metrics.confusion_matrix(ytest,ypred2)
```

[92]:
```
array([[2171,  120],
       [  46,  663]])
```

[93]:
```
print(classification_report(ytest,ypred2))
```

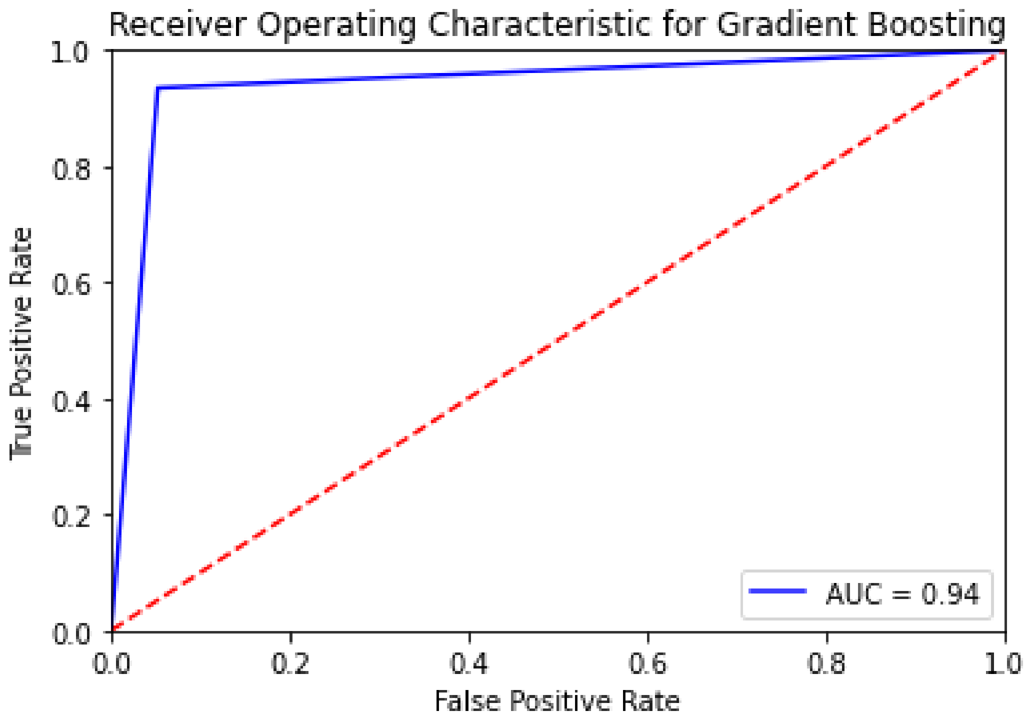
```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96      2291
           1       0.85      0.94      0.89       709

    accuracy                           0.94      3000
   macro avg       0.91      0.94      0.93      3000
weighted avg       0.95      0.94      0.95      3000
```

[94]:
```
roc_auc_score(ytest,ypred2)
```

[94]: 0.9413705066554046

[95]:
```
fpr, tpr, threshold = metrics.roc_curve(ytest, ypred2)
print(fpr)
print(tpr)
print(threshold)
roc_auc = metrics.auc(fpr, tpr)
print(roc_auc)

# method I: plt
plt.title('Receiver Operating Characteristic for Gradient Boosting')
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()
```

```
[0.         0.05237887 1.        ]
[0.         0.93511989 1.        ]
[inf  1.  0.]
0.9413705066554046
```




[96]:
```
#Based on the confusion matrix, the false negatives should be low because if an
↪employee who might leave the organisation is misclassified as someone who
↪won't leave then proper strategies to retain that person will not be
↪implemented on him or her. Hence Recall is better metric to be used
```

[97]:
```
col = xtrainres.columns
```

[98]:
```
col
```

[98]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident',
       'promotion_last_5years', 'sales_IT', 'sales_RandD', 'sales_accounting',
       'sales_hr', 'sales_management', 'sales_marketing', 'sales_product_mng',
       'sales_sales', 'sales_support', 'sales_technical', 'salary_high',
       'salary_low', 'salary_medium'],
      dtype='object')
```

[99]:
```
#Since Random Forest shows the highest accuracy with good f1 score, we will
↪conclude that to be our best performing model.
```

[100]:
```
feature_labels = np.array(col)
```

[101]:
```
importance = randm.feature_importances_
feature_indexes_by_importance = importance.argsort()
for index in feature_indexes_by_importance:
    print('{}-{:.2f}%'.format(feature_labels[index], (importance[index] *100.
↪0)))
```

```
sales_hr-0.01%
sales_support-0.01%
sales_accounting-0.01%
sales_marketing-0.01%
sales_technical-0.02%
sales_IT-0.04%
sales_sales-0.04%
sales_product_mng-0.05%
sales_RandD-0.22%
promotion_last_5years-0.23%
sales_management-0.25%
salary_medium-0.32%
salary_low-0.43%
salary_high-1.56%
Work_accident-3.67%
last_evaluation-11.16%
average_montly_hours-11.20%
number_project-16.26%
time_spend_company-24.81%
satisfaction_level-29.68%
```

[102]:
```
#The above lists the factors that influences the turnover in the ascending␣
↪order. It can be identified that the employee turnover is highly influenced␣
↪by the employee's satisfaction level in the organisation. Improvement of␣
↪work culture within the organiation can be a good way to prevent the␣
↪employees from leaving the organisation.
```

[103]:
```
predict_probability = randm.predict_proba(xtest)
```

[104]:
```
predict_probability[:,1]
```

[104]:
```
array([0.05807485, 0.1337665 , 0.09630147, …, 0.68267309, 0.06180619,
       0.14486645])
```

[105]:
```
zone=[]
prob=[]

for i in predict_probability[:,1]:
  prob.append(i)
  if (i<=0.2):
    zone.append("Safe Zone")
```

```
  elif (i>0.2 and i<=0.6):
    zone.append("Low Risk Zone")
  elif (i>0.6 and i<=0.9):
    zone.append("Medium Risk Zone ")
  else:
    zone.append("High Risk Zone ")
```

[106]:
```
categories = ["Safe Zone","Low Risk Zone","Medium Risk Zone ","High Risk Zone "]
color = ["Green","Yellow","Orange","Red"]
```

[107]:
```
colordict = dict(zip(categories, color))
```

[108]:
```
clr = pd.DataFrame({"zone":zone,"probability":prob})
```

[109]:
```
clr["zone"].unique()
```

[109]:
```
array(['Safe Zone', 'High Risk Zone ', 'Medium Risk Zone ',
       'Low Risk Zone'], dtype=object)
```

[110]:
```
clr["Color"] = clr["zone"].apply(lambda x: colordict[x])
```

[111]:
```
clr.head()
```

[111]:

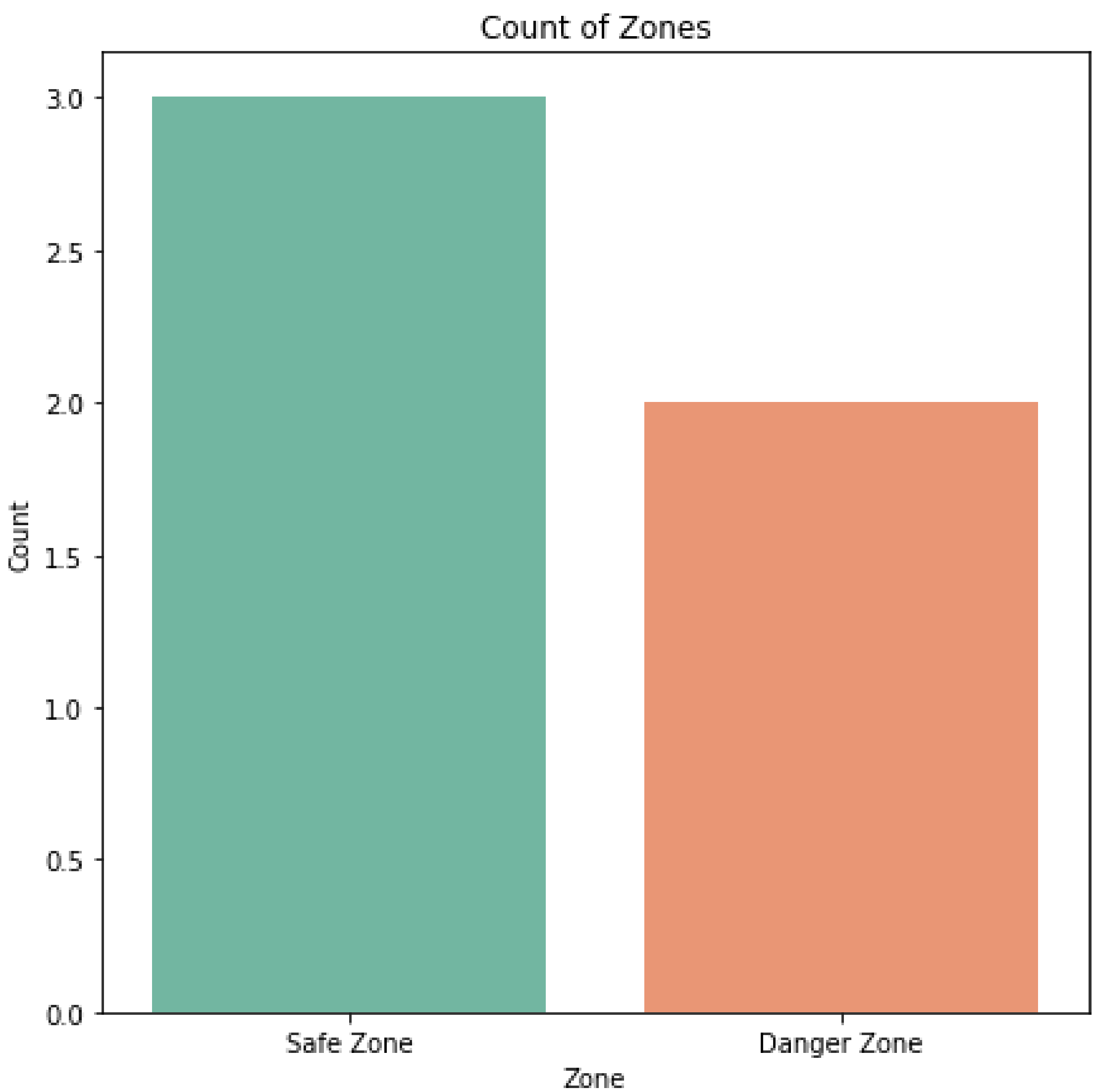
| | zone | probability | Color |
|---|---|---|---|
| 0 | Safe Zone | 0.058075 | Green |
| 1 | Safe Zone | 0.133767 | Green |
| 2 | Safe Zone | 0.096301 | Green |
| 3 | Safe Zone | 0.084050 | Green |
| 4 | Safe Zone | 0.127660 | Green |

[112]:
```
color= clr["Color"].tolist()
```

[113]:
```
c = ["Green","Red","Orange","Yellow"]
```

[118]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Example data (replace with your actual data)
zone = ['Safe Zone', 'Danger Zone', 'Safe Zone', 'Safe Zone', 'Danger Zone']
c = "Set2"  # Example palette

# Check if zone is a valid categorical variable
# Convert to categorical dtype if necessary (for Pandas Series)
# Ensure palette is correctly defined and accessible

# Set the figure size
plt.figure(figsize=(7, 7))
```

```
# Create the count plot with seaborn
sns.countplot(x=zone, palette=c)

# Add labels and title if needed
plt.xlabel('Zone')
plt.ylabel('Count')
plt.title('Count of Zones')

# Display the plot
plt.show()
```




[ ]:

```
```