Employee_turnover_analytics (1)

June 9, 2024

[125]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

[126]:
```
df = pd.read_excel("/content/1673873196_hr_comma_sep.xlsx")
```

[127]:
```
df
```

[127]:
```
       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]
```

[128]:
```
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
```

[129]:
```
df.isna().sum()
```

[129]:
```
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
```

No missing values in the data

[130]:
```
df["left"].unique()
```

[130]:
```
array([1, 0])
```

[131]:
```
df["promotion_last_5years"].unique()
```

[131]:
```
array([0, 1])
```

[132]:
```
df["number_project"].unique()
```

[132]:
```
array([2, 5, 7, 6, 4, 3])
```

[133]:
```
df.satisfaction_level.unique()
```

[133]:
```
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])
```

[134]:
```
df.last_evaluation.unique()
```

[134]:
```
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])
```

[135]:
```
df.time_spend_company.unique()
```

[135]:
```
array([ 3,  6,  4,  5,  2,  8, 10,  7])
```

[136]:
```
df.Work_accident.unique()
```

[136]:
```
array([0, 1])
```

[137]:
```
df.sales.unique()
```

[137]:
```
array(['sales', 'accounting', 'hr', 'technical', 'support', 'management',
       'IT', 'product_mng', 'marketing', 'RandD'], dtype=object)
```

[138]:
```
df.salary.unique()
```

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

[139]:
```
df.corr()
```

[139]:

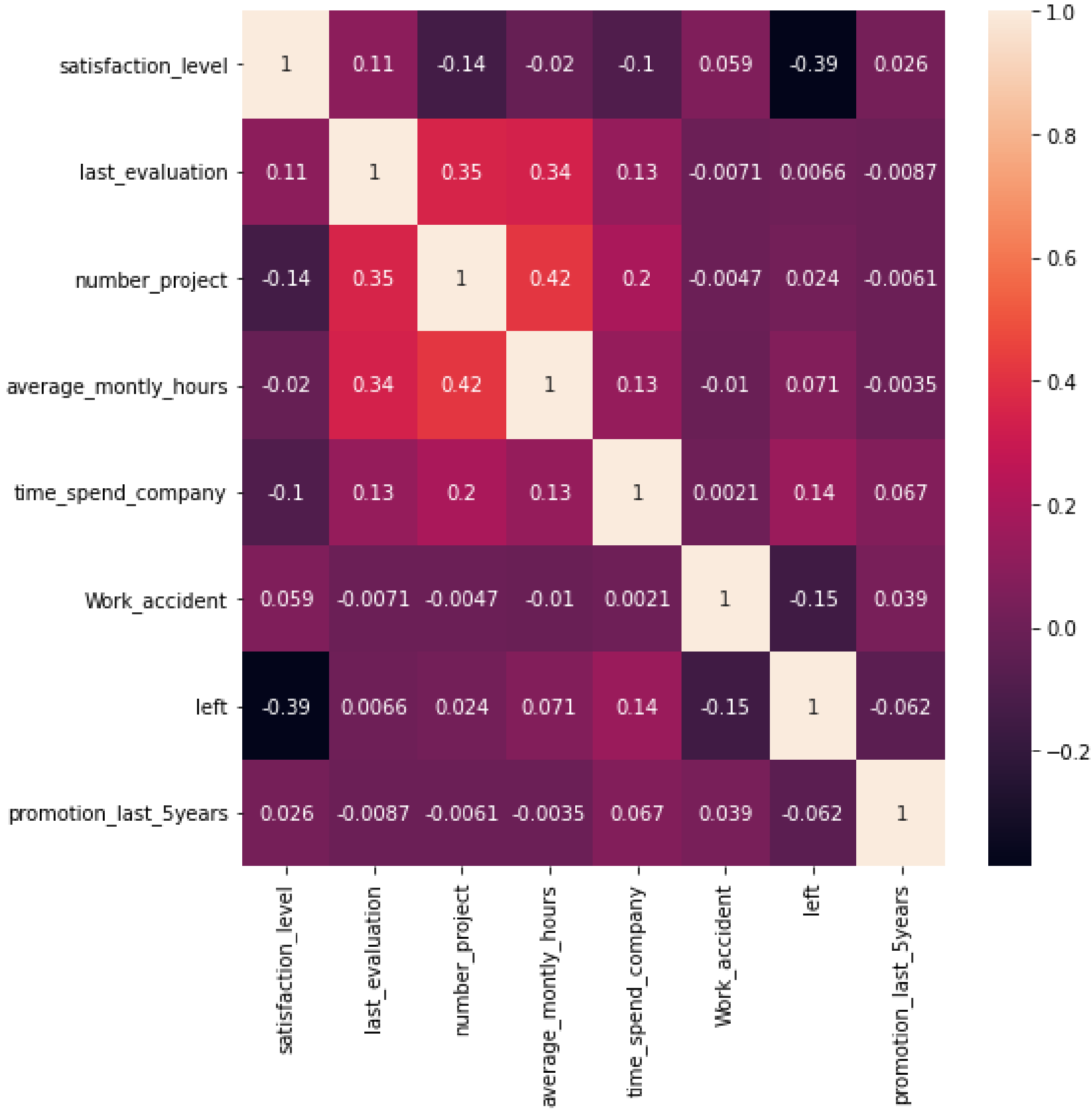
	satisfaction_level	last_evaluation	number_project
satisfaction_level	1.000000	0.105021	-0.142970
last_evaluation	0.105021	1.000000	0.349333
number_project	-0.142970	0.349333	1.000000
average_montly_hours	-0.020048	0.339742	0.417211
time_spend_company	-0.100866	0.131591	0.196786
Work_accident	0.058697	-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

[140]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)
```

[140]: <AxesSubplot:>



[141]:
```
df1= df.groupby(["sales"])["left"].value_counts().reset_index(name="count")
df1=pd.DataFrame(df1)
```

[142]:
```
df["sales"].value_counts()
```

[142]:
```
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
```

[143]:
```
dft=df["sales"].value_counts().reset_index(name="Total")
```

[144]:
```
dft=dft.rename(columns={"index":"sales"})
```

[145]:
```
dft
```

[145]:

	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

[146]:
```
dfmer=df1.merge(dft,how="left")
```

[147]:
```
dfmer
```

[147]:

	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
```

[148]:
```
dfmer["normal"]=dfmer["count"].div(dfmer["Total"].values)
dfmer["normal"]=dfmer["normal"]*100
```

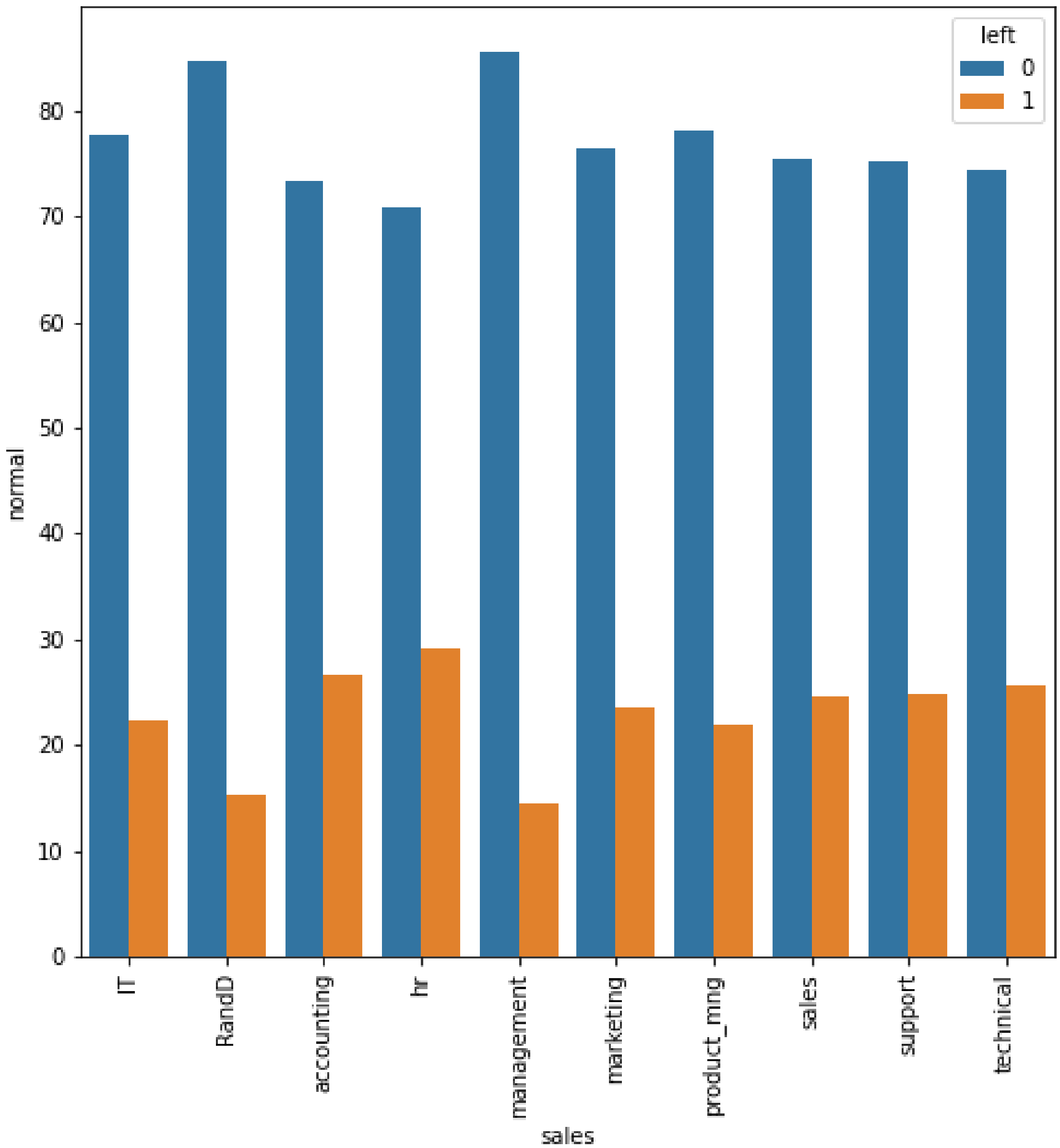
[149]:
```
dfmer
```

[149]:

	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

[150]:
```
plt.figure(figsize=(8,8))
sns.barplot(x="sales",y='normal',hue="left",data=dfmer)
plt.xticks(rotation=90)
```

[150]:
```
(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')])
```



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

[151]:
```
df1.head()
```

[151]:

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

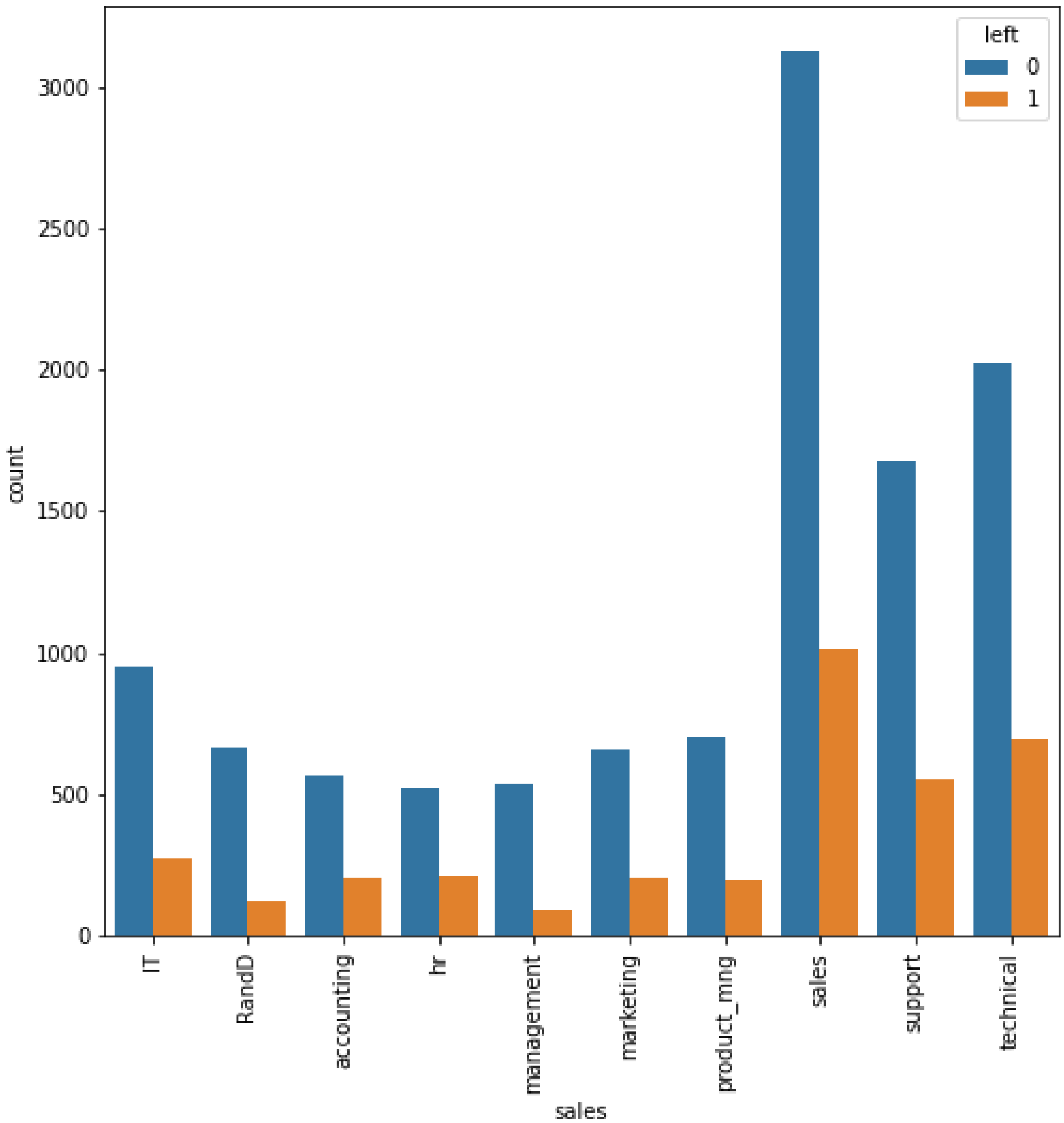
```
3        RandD      1     121
4   accounting      0     563
```

[152]:

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

[152]:

```
(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')])
```



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

[153]:
```
df2= df.groupby(["salary"])["left"].value_counts().reset_index(name="count")
df2=pd.DataFrame(df2)
```

[154]:
```
df2.head()
```

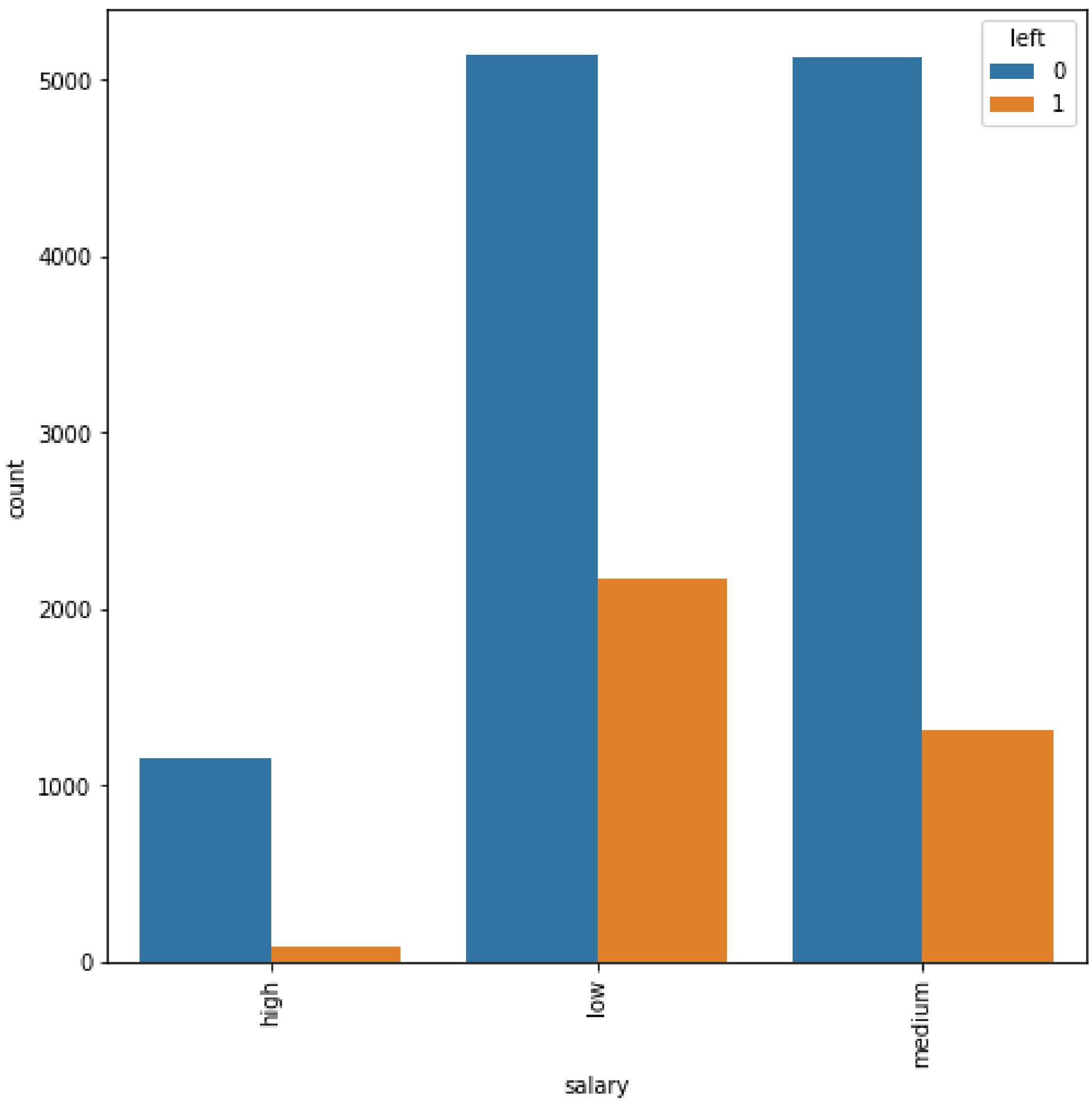
[154]:

	salary	left	count
0	high	0	1155
1	high	1	82

```
2     low     0   5144
3     low     1   2172
4  medium     0   5129
```

[155]:
```
plt.figure(figsize=(8,8))
sns.barplot(x="salary",y='count',hue="left",data=df2)
plt.xticks(rotation=90)
```

[155]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'high'), Text(1, 0, 'low'), Text(2, 0, 'medium')])
```

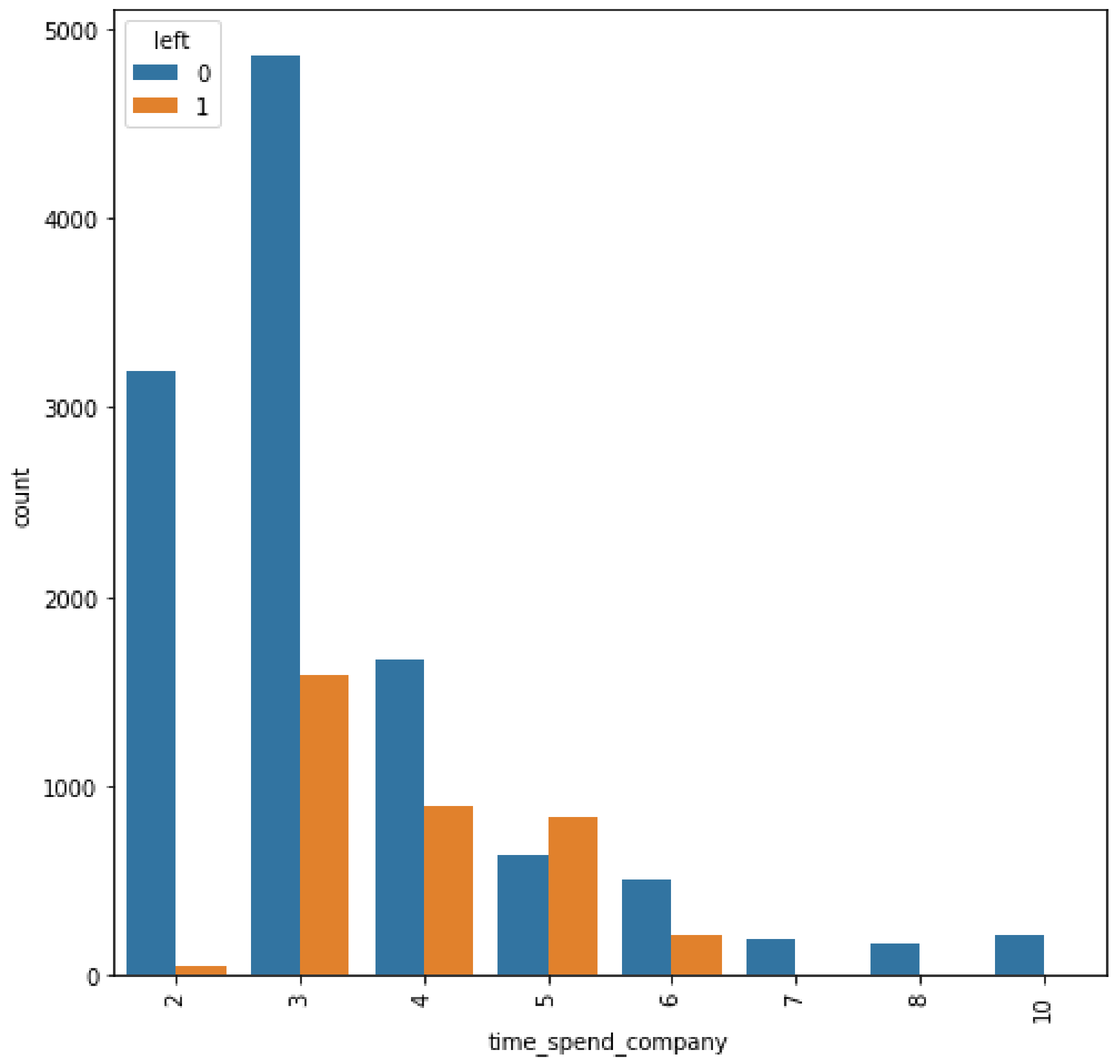


People with Lower Salaries are leaving the company

[156]:
```
df3= df.groupby(["time_spend_company"])["left"].value_counts().
 ↪reset_index(name="count")
df3=pd.DataFrame(df3)
```

[157]:
```
#time_spend_company
plt.figure(figsize=(8,8))
sns.barplot(x="time_spend_company",y='count',hue="left",data=df3)
plt.xticks(rotation=90)
```

[157]:
```
(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')])
```

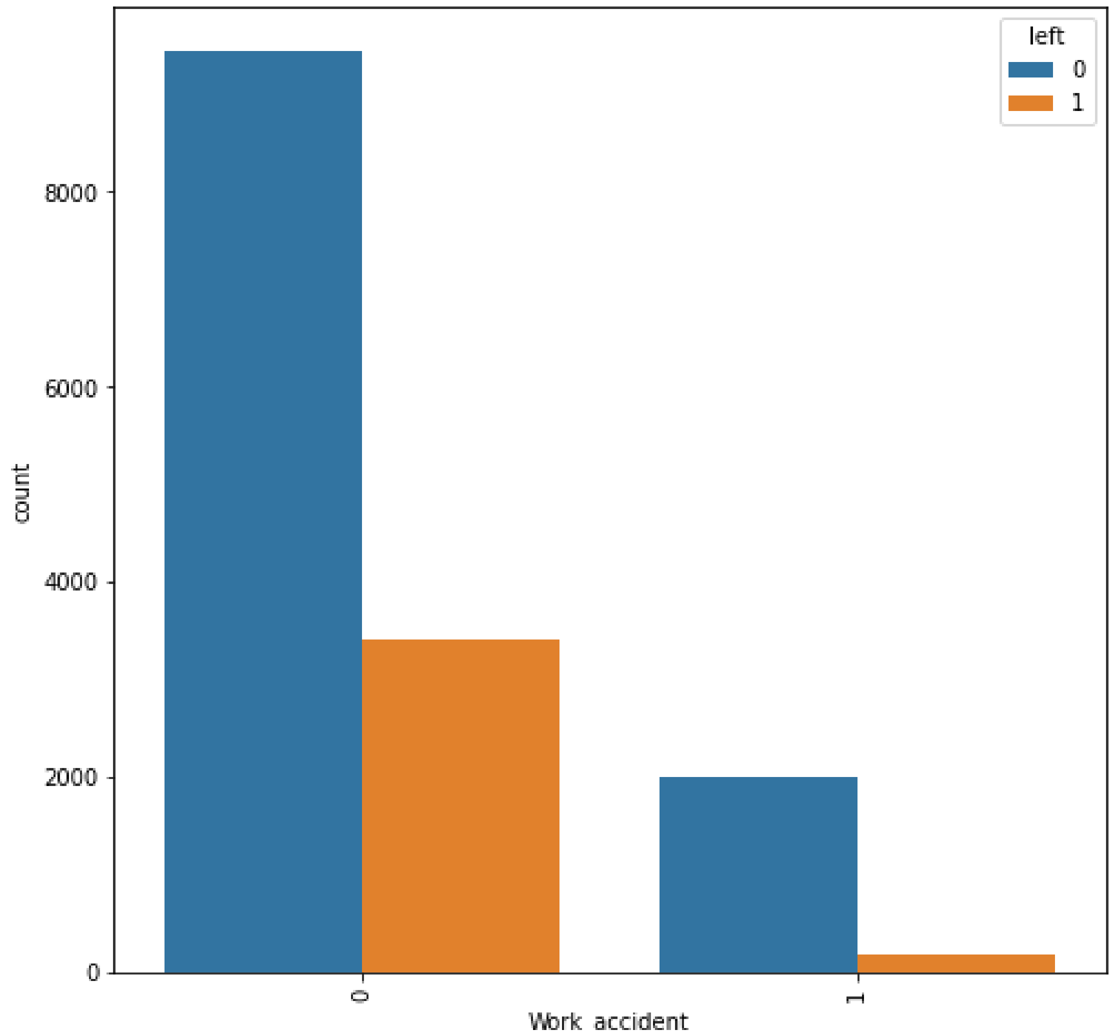


People with experience of 3 to 5 years are leaving the comapny more.

[158]:
```
plt.figure(figsize=(8,8))
sns.countplot("Work_accident",hue="left",data=df)
plt.xticks(rotation=90)
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[158]: (array([0, 1]), [Text(0, 0, '0'), Text(1, 0, '1')])

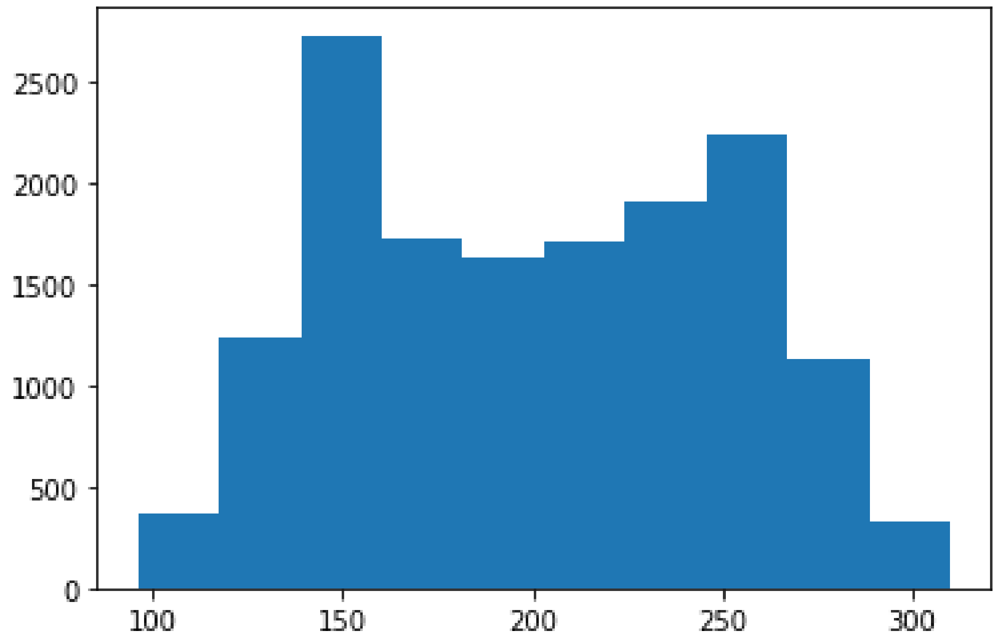


[159]:
```
df.columns
```

[159]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'sales', 'salary'],
      dtype='object')
```

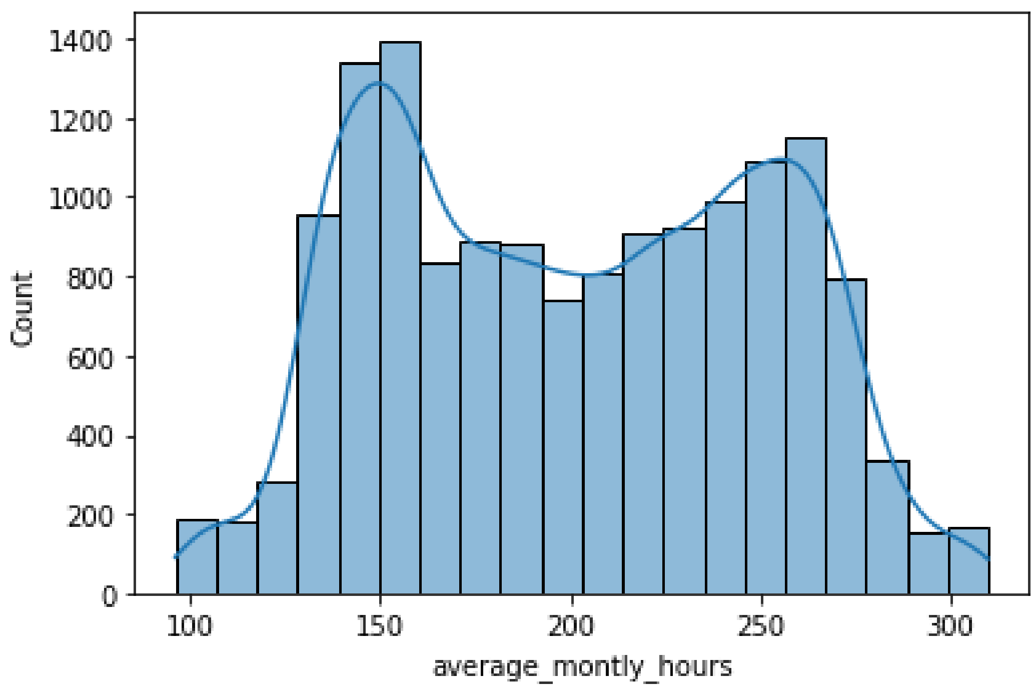
[160]:
```
plt.hist(df["average_montly_hours"])
```

[160]:
```
(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>)
```



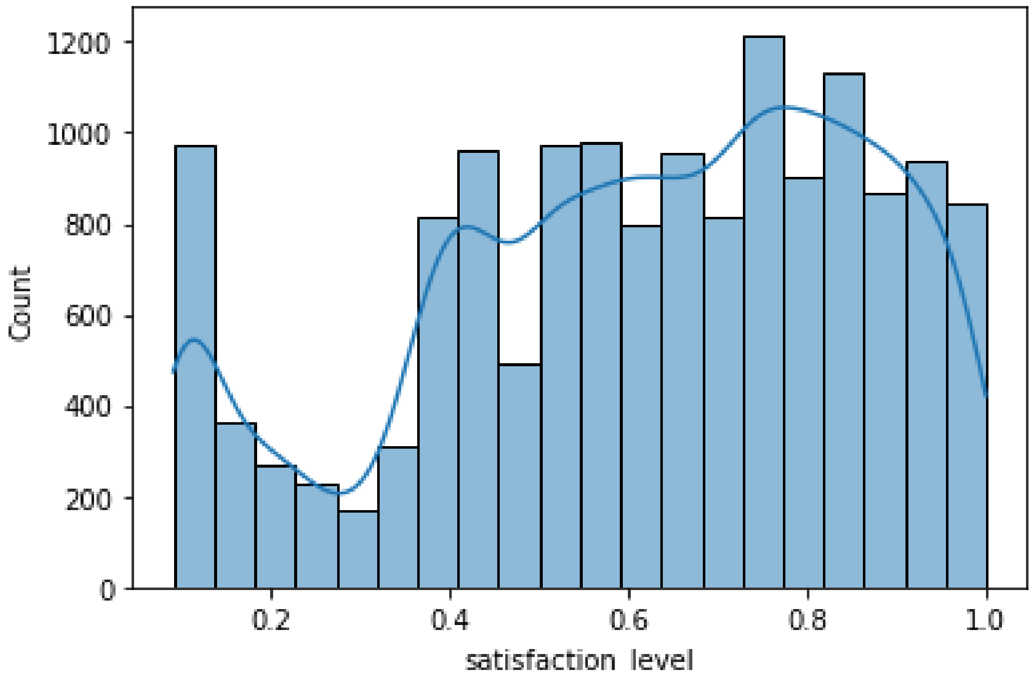
[161]:
```
sns.histplot(data = df,x="average_montly_hours", kde = True,bins=20)
```

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



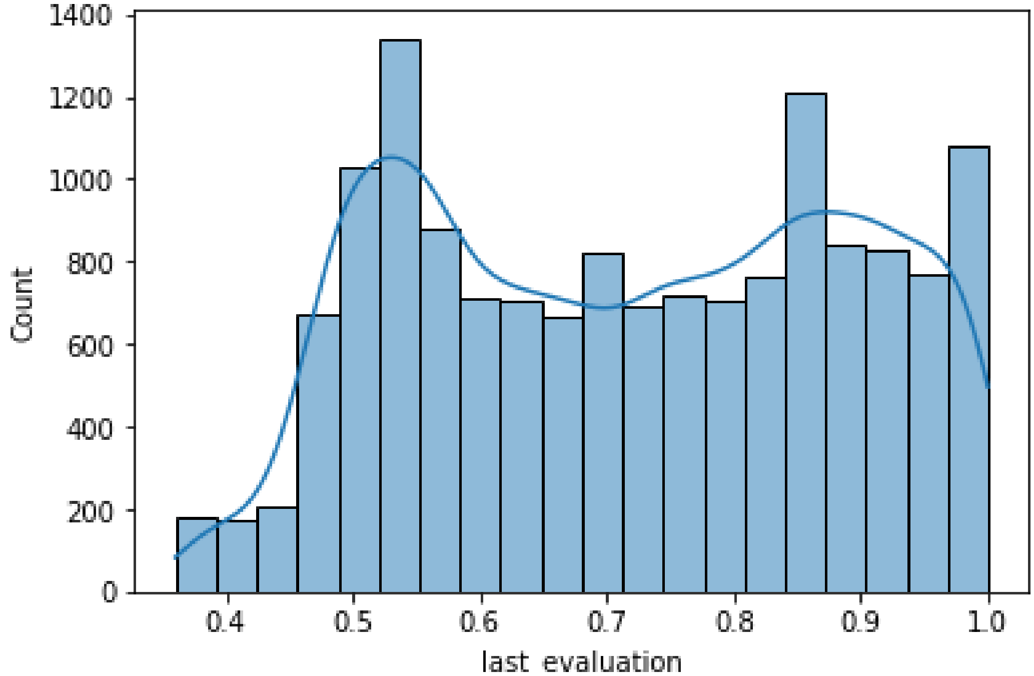
```
[162]: sns.histplot(data = df,x="satisfaction_level", kde = True,bins=20)
```

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



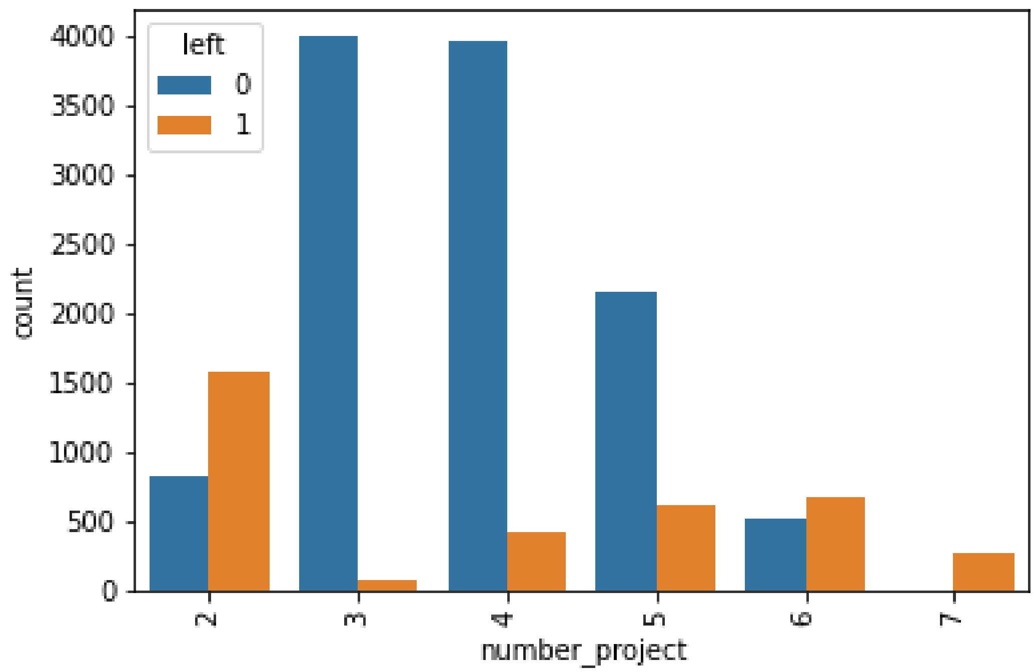

```
[163]: sns.histplot(data = df,x="last_evaluation", kde = True,bins=20)
```

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



[164]:
```
sns.countplot(x="number_project",hue="left",data=df)
plt.xticks(rotation=90)
```

[164]:
```
(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')])
```



People who have worked on 3 or 4 projects have left the organisation more.

[165]:
```
dfclus = df[["satisfaction_level","last_evaluation","left"]]
```

[166]:
```
dfclus
```

[166]:
```
       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]
```

[167]:
```
from sklearn.cluster import KMeans
```

[168]:
```
km=dfclus.iloc[:,:].values
kmeans = KMeans(n_clusters=3, random_state=0)
```

```
label = kmeans.fit_predict(dfclus)
labelarr = kmeans.fit_predict(km)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[169]:
```
label
```

[169]:
```
array([1, 1, 1, …, 1, 1, 1], dtype=int32)
```

[170]:
```
dfclus[label==0].describe()
```

[170]:

	satisfaction_level	last_evaluation	left
count	6720.000000	6720.000000	6720.0
mean	0.813112	0.739728	0.0
std	0.108167	0.154900	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

[171]:
```
dfclus[label==1].describe()
```

[171]:

	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

[172]:
```
dfclus[label==2].describe()
```

[172]:

	satisfaction_level	last_evaluation	left
count	4708.000000	4708.000000	4708.0
mean	0.457984	0.680854	0.0
std	0.153456	0.165609	0.0
min	0.120000	0.360000	0.0

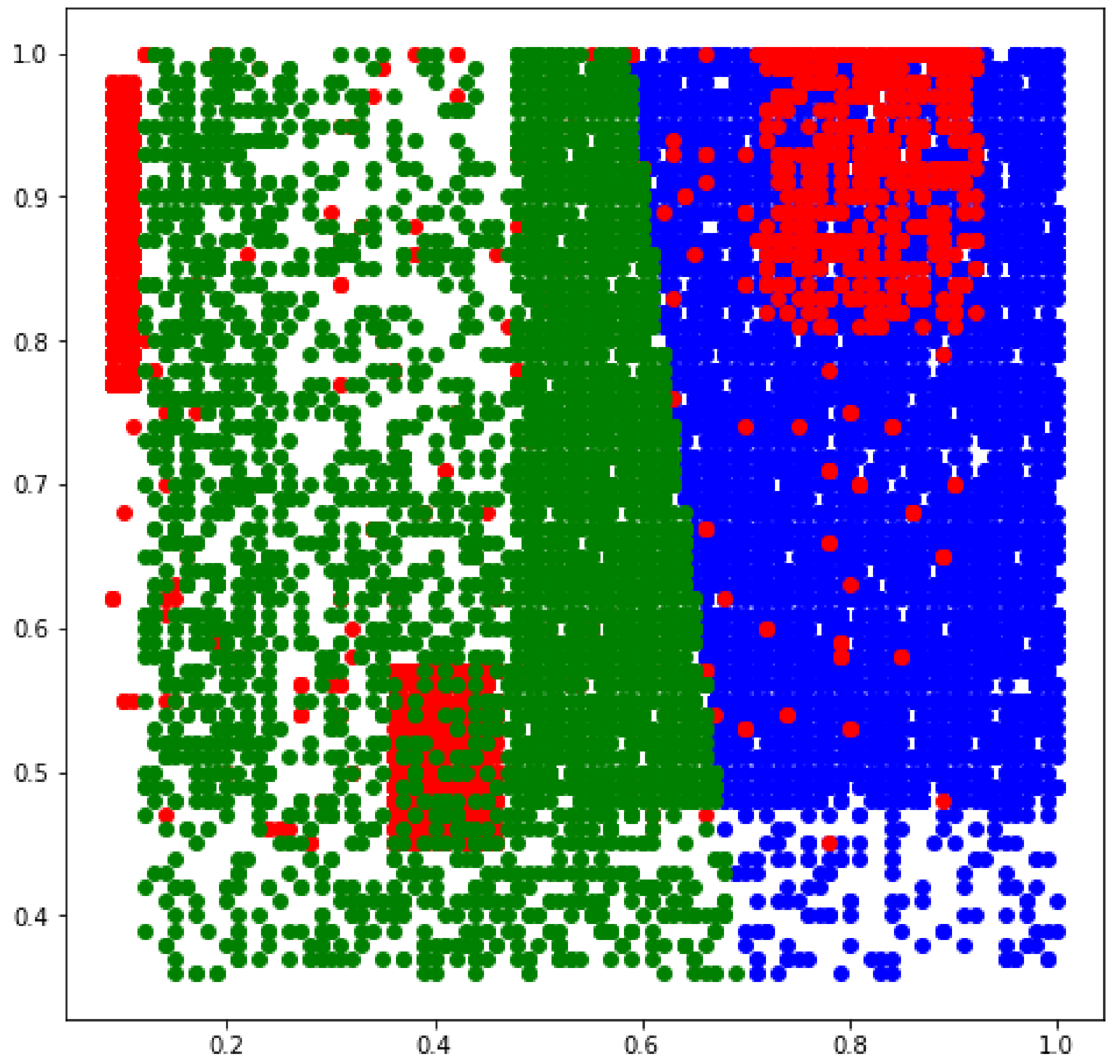
```
25%             0.350000        0.550000     0.0
50%             0.510000        0.670000     0.0
75%             0.570000        0.810000     0.0
max             0.690000        1.000000     0.0
```

[173]:
```
km[label==0,1]
```

[173]:
```
array([0.67, 0.82, 0.91, …, 0.55, 0.95, 0.54])
```

[174]:
```
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")
```

[174]:
```
<matplotlib.collections.PathCollection at 0x7f62de72a1f0>
```



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.

[175]:
```
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
```

[176]:
```
df_numerical=df.select_dtypes(include=['int64','float64'])
df_categorical=df.select_dtypes(include=['object'])
```

Converting the categorical data into numerical using one hot encoding

[177]:
```
#df = pd.get_dummies(data=df,columns=['sales','salary'])
df_converted = pd.get_dummies(data=df_categorical)
```

[178]:
```
df_converted.head()
```

[178]:
```
   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
```

[179]:
```
dfn = pd.concat([df_numerical, df_converted], axis=1, join="inner")
```

[180]:
```
dfn.shape
```

[180]:
```
(14999, 21)
```

[181]:
```
dfn.head()
```

[181]:
```
   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]
```

Splitting the dataset into training and testing in the ratio of 80:20 with random state = 123.

[182]:
```
x =dfn.drop("left",axis=1)
y = dfn["left"]
```

[183]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=123)
```

[184]:
```
xtrain.shape,ytrain.shape,xtest.shape,ytest.shape
```

[184]:
```
((11999, 20), (11999,), (3000, 20), (3000,))
```

[185]:
```
ytrain.value_counts()
```

[185]:
```
0    9137
1    2862
Name: left, dtype: int64
```

Data is highly imbalanced for the training dataset as the record of people who left is very low in comparision to the record of people who didn't leave.

Using SMOTE to handle the imbalance for the left category

[186]:
```
from imblearn.over_sampling import SMOTE
```

[187]:
```
sm = SMOTE(random_state = 2)
xtrainres, ytrainres = sm.fit_resample(xtrain, ytrain)
```

[188]:
```
ytrainres.value_counts()
```

[188]:
```
0    9137
1    9137
Name: left, dtype: int64
```

[189]:
```
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
```

[190]:
```
logreg = LogisticRegression(solver='lbfgs', max_iter=10000)
```

[191]:
```
print(cross_val_score(logreg, xtrainres, ytrainres, cv=5).mean())
```

```
0.8061742654827233
```

[192]:
```
logreg.fit(xtrainres,ytrainres)
ypred = logreg.predict(xtest)
```

[193]:
```
from sklearn.metrics import classification_report
```

Logistic regression report

[194]:
```
metrics.confusion_matrix(ytest,ypred)
```

[194]:
```
array([[1831,  460],
       [ 228,  481]])
```

[195]:
```
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
```

[196]:
```
roc_auc_score(ytest,ypred)
```

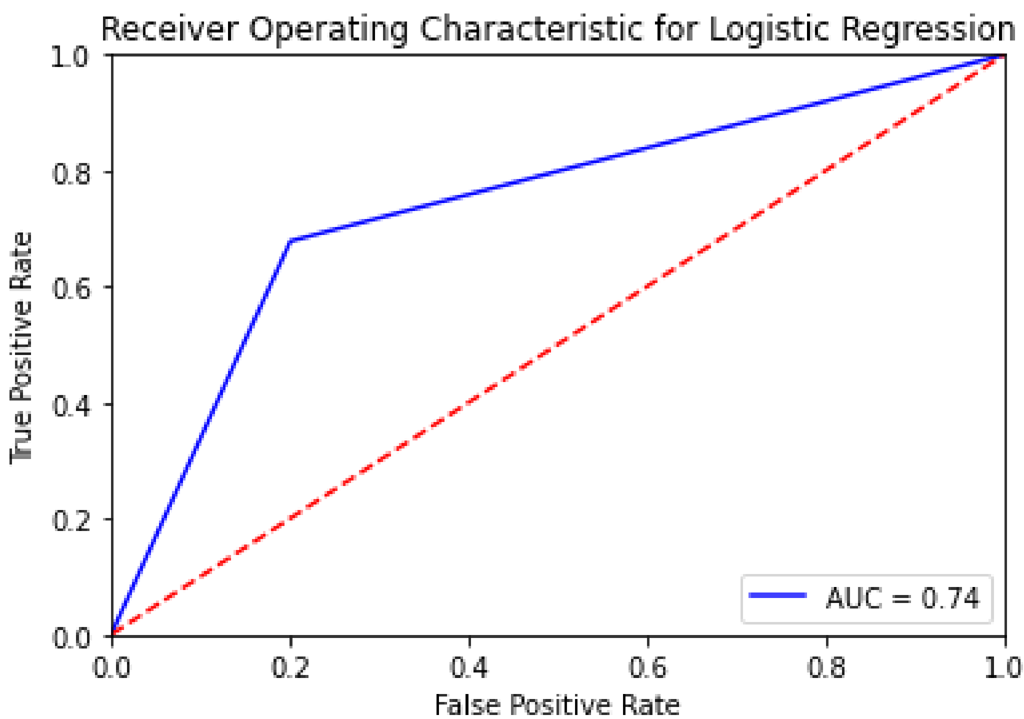
[196]:
```
0.7388173135941893
```

[197]:
```
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.        ]
[2 1 0]
0.7388173135941893
```



Random Forest Classifier

[198]:
```
randm=RandomForestClassifier(max_depth=5)
```

[199]:
```
print(cross_val_score(randm, xtrainres, ytrainres, cv=5).mean())
```

```
0.9476854478760229
```

[200]:
```
randm.fit(xtrainres,ytrainres)
ypred1=randm.predict(xtest)
```

Random Forest Classification report

[201]:
```
metrics.confusion_matrix(ytest,ypred1)
```

[201]: array([[2229,   62],
       [55, 654]])

[202]:
```
print(classification_report(ytest,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      2291
           1       0.91      0.92      0.92       709

    accuracy                           0.96      3000
   macro avg       0.94      0.95      0.95      3000
weighted avg       0.96      0.96      0.96      3000
```

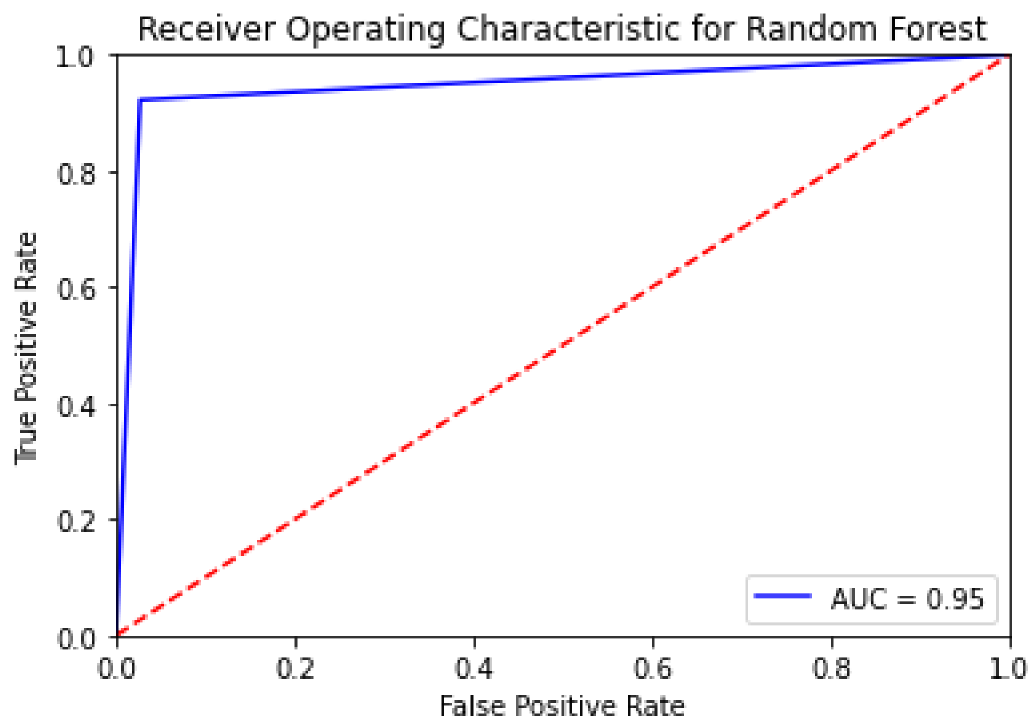
[203]:
```
roc_auc_score(ytest,ypred1)
```

[203]: 0.9476817669435622

[204]:
```
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.         0.02706242 1.        ]
[0.         0.92242595 1.        ]
[2 1 0]
0.9476817669435622
```



Gradient Boosting Classifier

[205]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

[206]:
```
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=1.
↪0,max_depth=1, random_state=0)
```

[207]:
```
print(cross_val_score(gb, xtrainres, ytrainres, cv=5).mean())
```

```
0.9478495915875037
```

[208]:
```
gb.fit(xtrainres,ytrainres)
```

[208]:
```
GradientBoostingClassifier(learning_rate=1.0, max_depth=1, random_state=0)
```

[209]:
```
ypred2 = gb.predict(xtest)
```

Gradient boosting Classification Report

[210]:
```
metrics.confusion_matrix(ytest,ypred2)
```

[210]:
```
array([[2171,  120],
       [  46,  663]])
```

[211]:
```
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
```

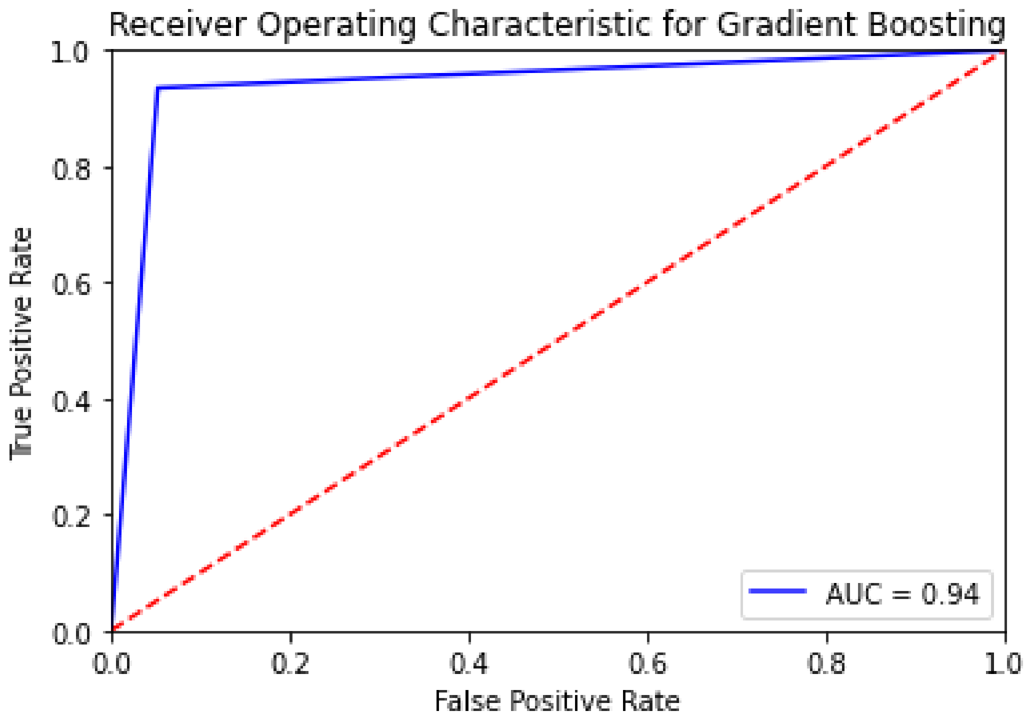
[212]:
```
roc_auc_score(ytest,ypred2)
```

[212]: 0.9413705066554046

[213]:
```
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.        ]
[2 1 0]
0.9413705066554046
```



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

[214]:
```
col = xtrainres.columns
```

[215]:
```
col
```

[215]:
```
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')
```

Since Random Forest shows the highest accuracy with good f1 score, we will conclude that to be our best performing model.

[216]:
```
feature_labels = np.array(col)
```

[217]:
```
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_technical-0.02%
sales_accounting-0.02%
sales_marketing-0.02%
sales_support-0.02%
sales_IT-0.05%
sales_sales-0.05%
sales_product_mng-0.07%
promotion_last_5years-0.08%
sales_management-0.09%
sales_RandD-0.13%
salary_medium-0.19%
salary_low-0.57%
salary_high-1.02%
Work_accident-3.39%
last_evaluation-10.70%
average_montly_hours-13.04%
number_project-17.36%
time_spend_company-24.02%
satisfaction_level-29.15%
```

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.

[218]:
```
predict_probability = randm.predict_proba(xtest)
```

[219]:
```
predict_probability[:,1]
```

[219]:
```
array([0.05155685, 0.09000749, 0.07828477, …, 0.71304061, 0.0654831 ,
       0.12558198])
```

[220]:
```
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 ")
```

[221]:
```
categories = ["Safe Zone","Low Risk Zone","Medium Risk Zone ","High Risk Zone "]
color = ["Green","Yellow","Orange","Red"]
```

[222]:
```
colordict = dict(zip(categories, color))
```

[223]:
```
clr = pd.DataFrame({"zone":zone,"probability":prob})
```

[224]:
```
clr["zone"].unique()
```

[224]:
```
array(['Safe Zone', 'High Risk Zone ', 'Medium Risk Zone ',
       'Low Risk Zone'], dtype=object)
```

[225]:
```
clr["Color"] = clr["zone"].apply(lambda x: colordict[x])
```

[230]:
```
clr.head(10)
```

[230]:

	zone	probability	Color
0	Safe Zone	0.051557	Green
1	Safe Zone	0.090007	Green
2	Safe Zone	0.078285	Green
3	Safe Zone	0.075446	Green
4	Safe Zone	0.094901	Green
5	Safe Zone	0.067924	Green
6	High Risk Zone	0.947130	Red
7	Medium Risk Zone	0.750938	Orange
8	Safe Zone	0.105290	Green
9	Safe Zone	0.062026	Green

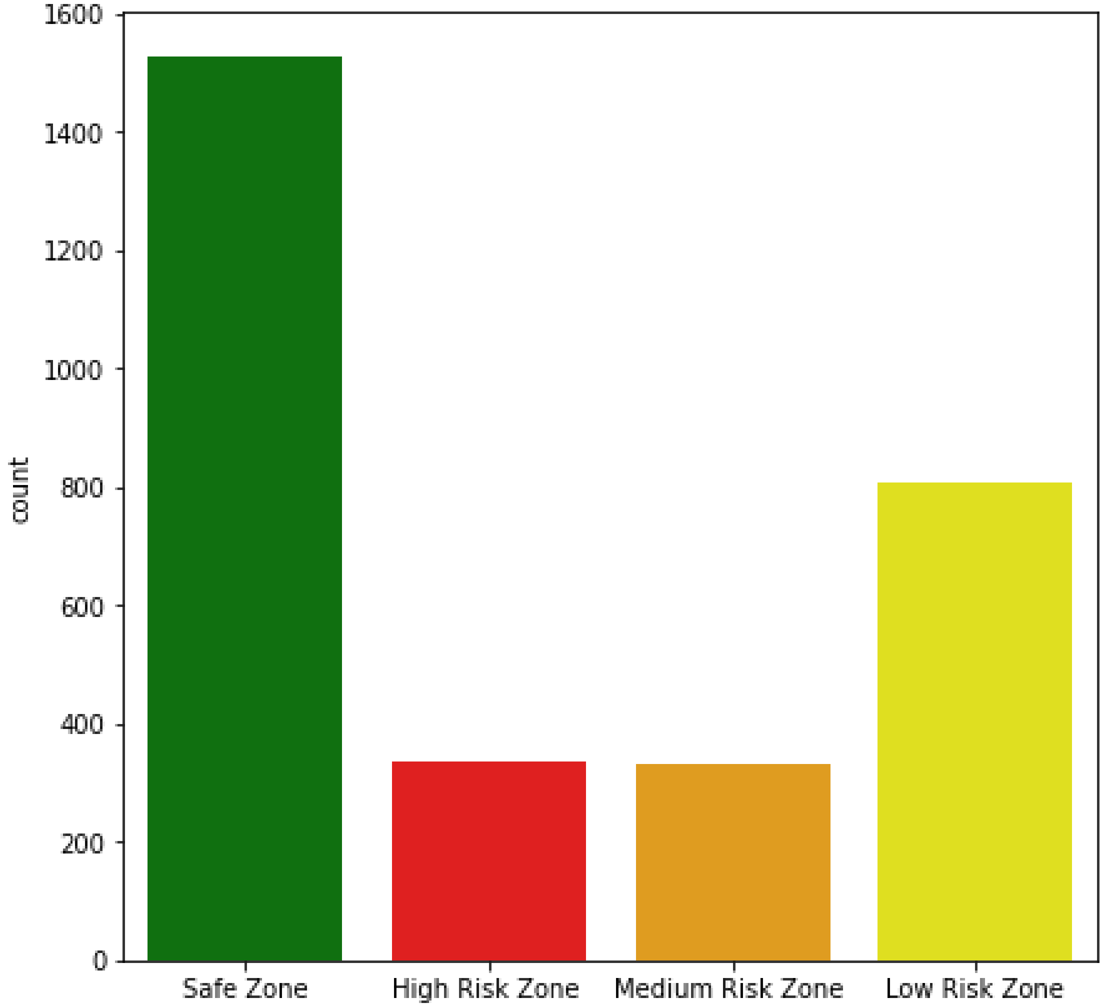
[227]:
```
color= clr["Color"].tolist()
```

[228]:
```
c = ["Green","Red","Orange","Yellow"]
```

[229]:
```
plt.figure(figsize=(7,7))
sns.countplot(zone,palette=c)
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[229]: <AxesSubplot:ylabel='count'>



[229]:

[229]:

[229]:

[229]:

[229]: