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

Machin Learning: Employee Turnover Analytics

Course end project

```
from google.colab import files

uploaded = files.upload()
```

Choose Files 167387319…ma_sep.xlsx

- **1673873196_hr_comma_sep.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 686264 bytes, last modified: 2/8/2023 - 100% done

```
Saving 1673873196_hr_comma_sep.xlsx to 1673873196_hr_comma_sep.xlsx
```

```
import io

df =  pd.read_excel(io.BytesIO(uploaded['1673873196_hr_comma_sep.xlsx']))
print(df)
```

```
       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]
```

```
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
```

```
df.isna().sum()
```

```
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
```

There are no missing values in the data.

```
df.shape
```

```
(14999, 10)
```

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

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

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

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

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

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

```
df.satisfaction_level.unique()
```

```
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])
```

```
df.last_evaluation.unique()
```

```
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])
```

```
df.time_spend_company.unique()
```

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

```
df.salary.unique()
```

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

```
df.Work_accident.unique()
```

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

```
df.sales.unique()
```

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

```
df.corr()
```

```
<ipython-input-15-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  df.corr()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_a
satisfaction_level	1.000000	0.105021	-0.142970	-0.020048	-0.100866	0
last_evaluation	0.105021	1.000000	0.349333	0.339742	0.131591	-0
number_project	-0.142970	0.349333	1.000000	0.417211	0.196786	-0
average_montly_hours	-0.020048	0.339742	0.417211	1.000000	0.127755	-0
time_spend_company	-0.100866	0.131591	0.196786	0.127755	1.000000	0
Work_accident	0.058697	-0.007104	-0.004741	-0.010143	0.002120	1
left	-0.388375	0.006567	0.023787	0.071287	0.144822	-0
promotion_last_5years	0.025605	-0.008684	-0.006064	-0.003544	0.067433	0

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

```
<ipython-input-16-6812646cd074>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  sns.heatmap(df.corr(),annot=True)
```

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

```
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
```

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

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

```
dft
```

	index	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

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

```
dft
```

	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

```
dfmer = df1.merge(dft,how="left")
```

```
dfmer
```

	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

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

```
dfmer
```

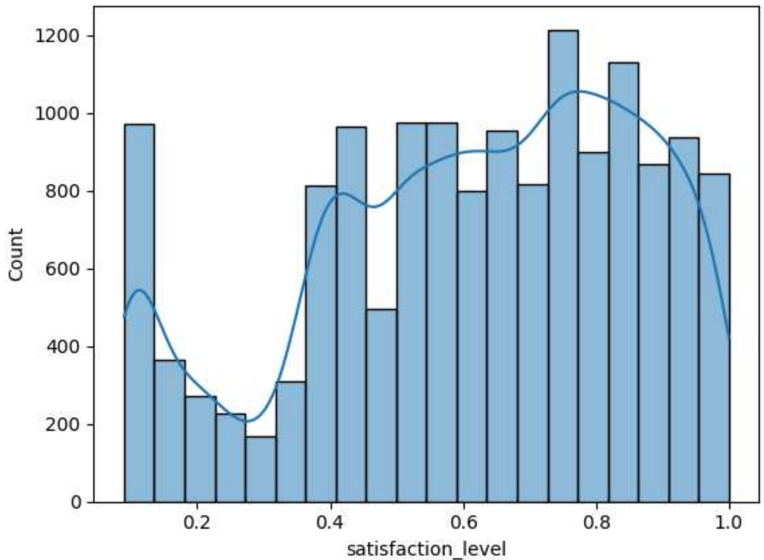
	sales	left	count	Total	normal
0	IT	0	954	1227	77.750611
1	IT	1	273	1227	22.249389
2	RandD	0	666	787	84.625159
3	RandD	1	121	787	15.374841
4	accounting	0	563	767	73.402868
5	accounting	1	204	767	26.597132
6	hr	0	524	739	70.906631
7	hr	1	215	739	29.093369
8	management	0	539	630	85.555556
9	management	1	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

```
df.columns
```

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

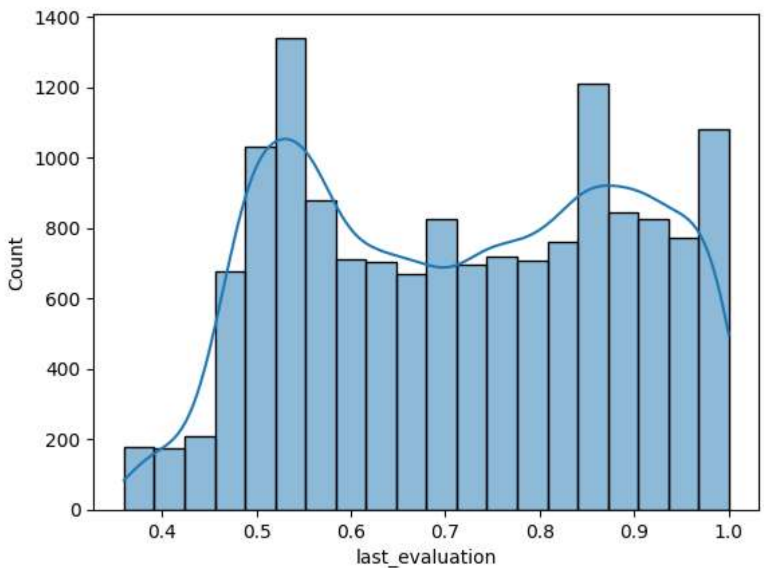
```
sns.histplot(data = df,x="satisfaction_level", kde = True,bins = 20)
```

```
<Axes: xlabel='satisfaction_level', ylabel='Count'>
```



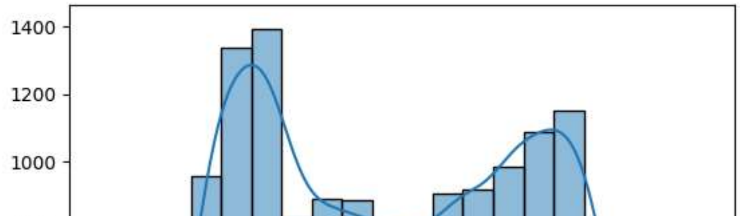

```
sns.histplot(data = df,x="last_evaluation", kde = True,bins = 20)
```

```
<Axes: xlabel='last_evaluation', ylabel='Count'>
```



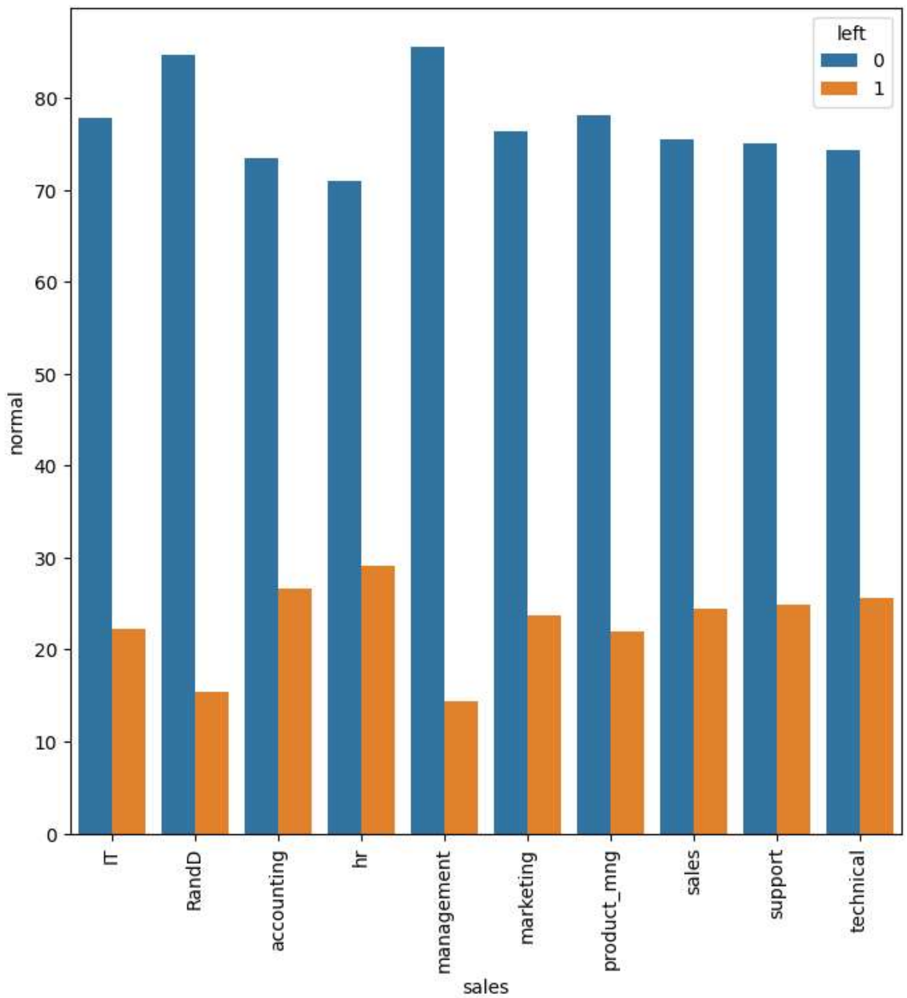

```
sns.histplot(data = df,x="average_montly_hours", kde = True,bins = 20)
```

```
<Axes: xlabel='average_montly_hours', ylabel='Count'>
```




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

```
(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')])
```



Based on the normalized data, HR department has the maximum amount of employees leaving.

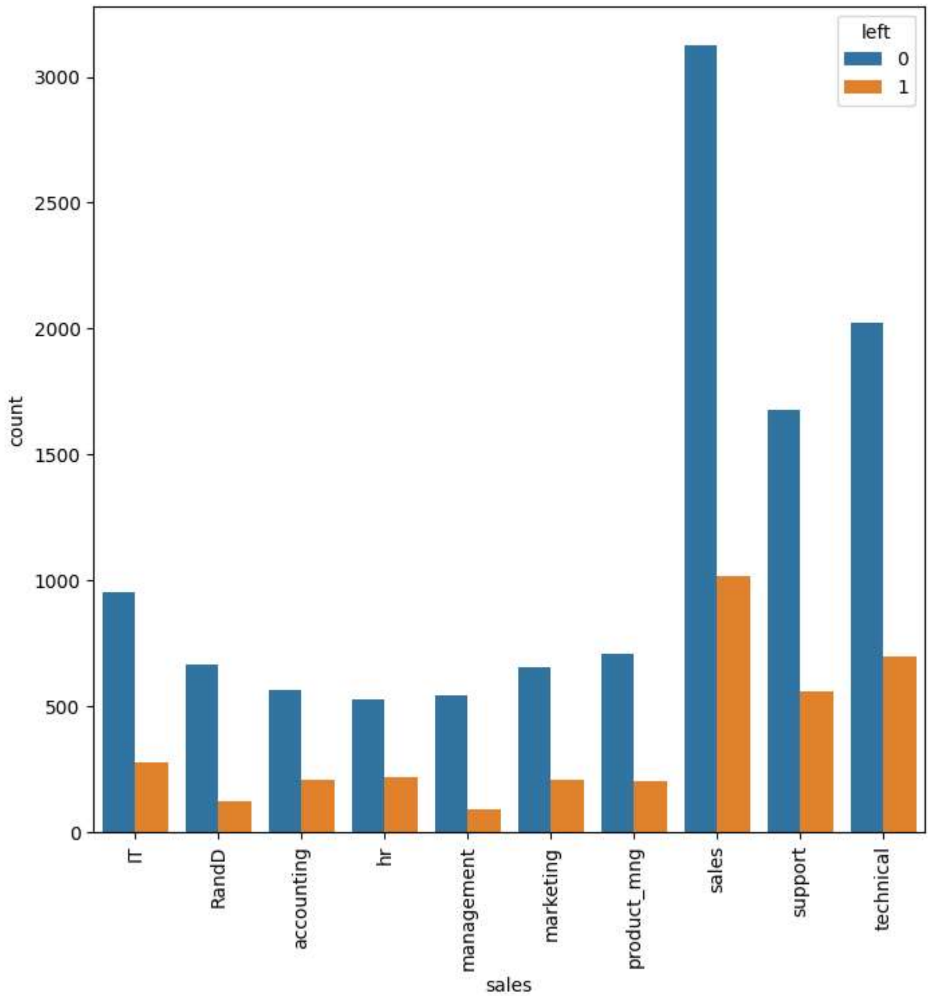
Normal = Number of people from the leaving category of a department)/(Total number of people in that department))*100

```
df1.head()
```

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

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

```
(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')])
```



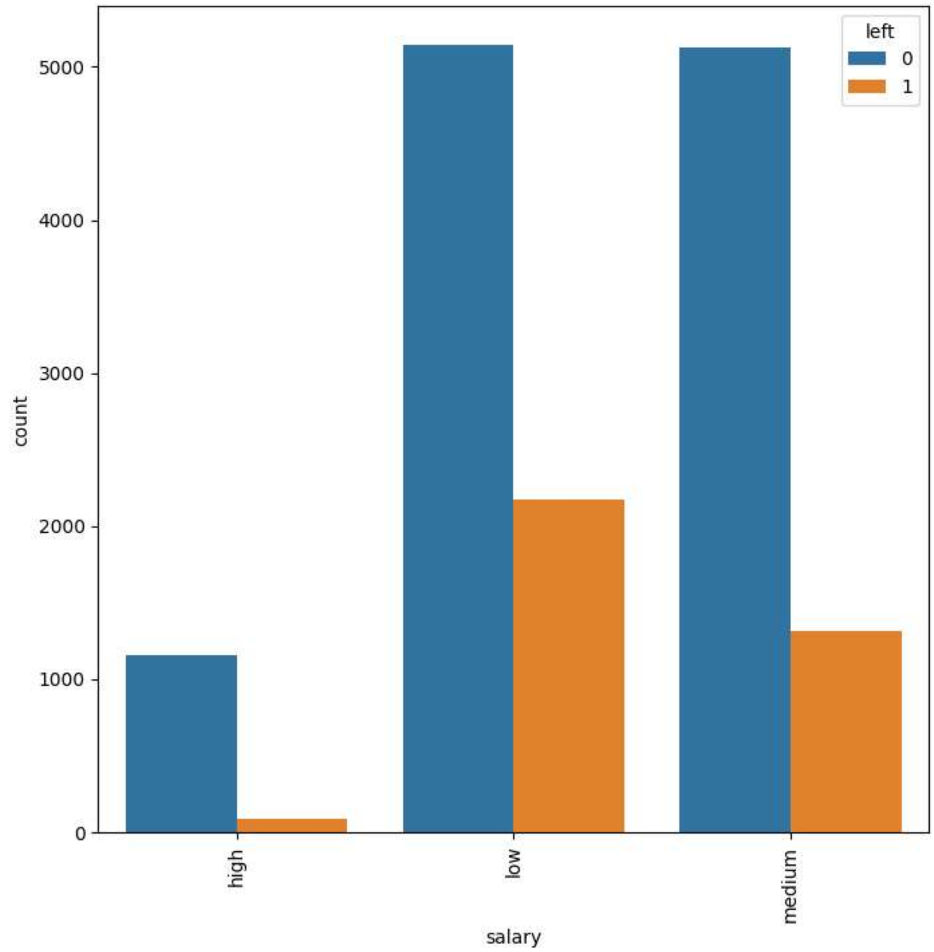
Taking the count of people leaving, maximum amount of people leaving is from the sales department.

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

	salary	left	count
0	high	0	1155

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

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



This showcases that people with lower salary are leaving the company.

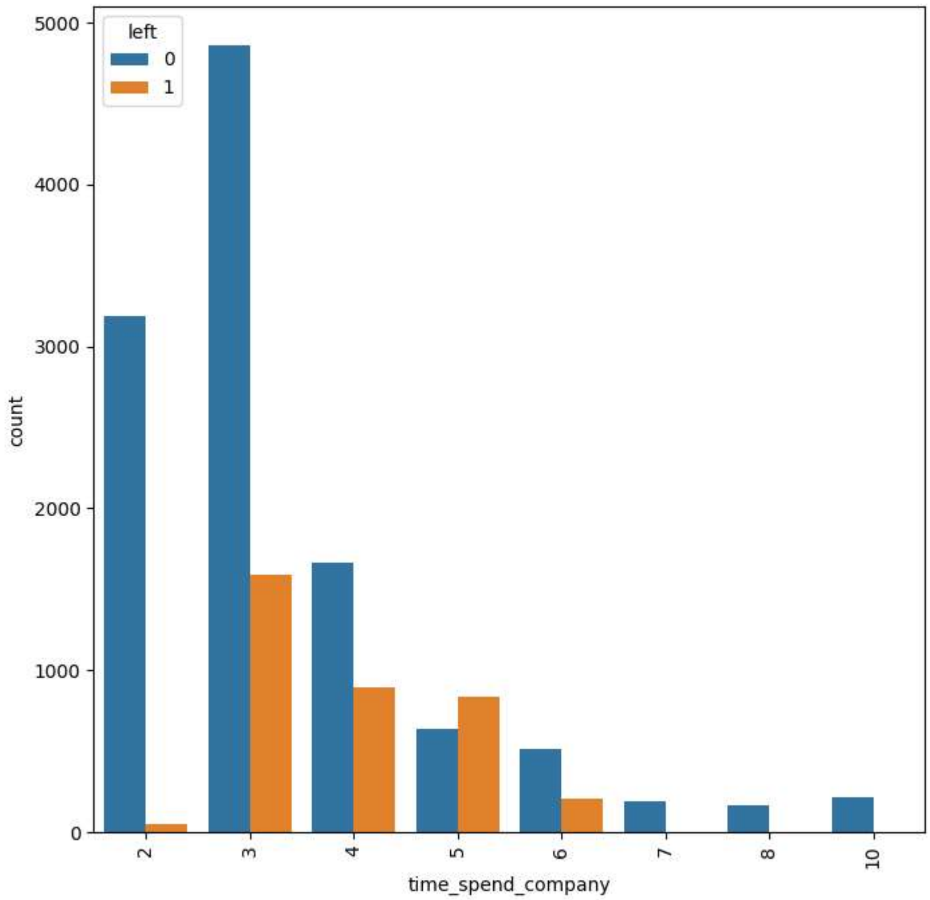
```
df3=df.groupby(["time_spend_company"])["left"].value_counts().reset_index(name="count")
df3=pd.DataFrame(df3)
```

```
df3
```

	time_spend_company	left	count
0	2	0	3191
1	2	1	53

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

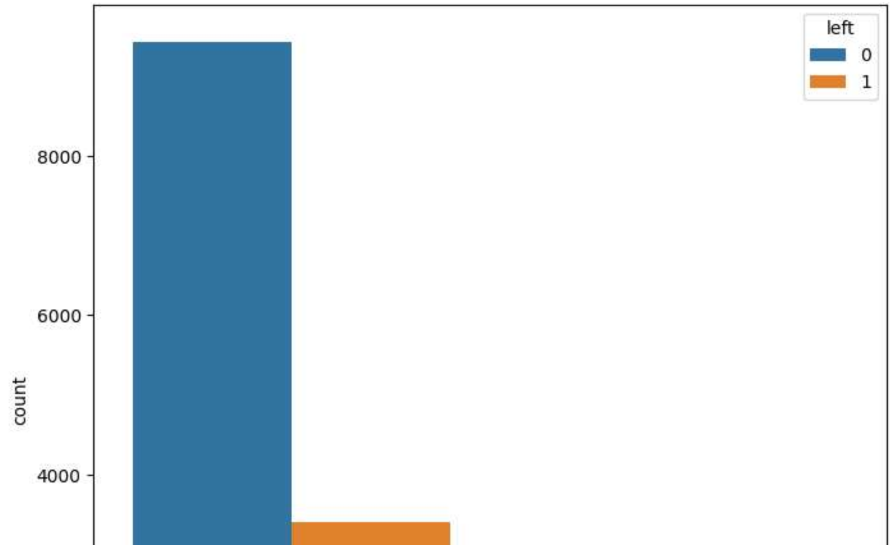
```
(array([0, 1, 2, 3, 4, 5, 6, 7]),
 [Text(0, 0, '2'),
  Text(1, 0, '3'),
  Text(2, 0, '4'),
  Text(3, 0, '5'),
  Text(4, 0, '6'),
  Text(5, 0, '7'),
  Text(6, 0, '8'),
  Text(7, 0, '10')])
```



People with 3 years of experience with the company are leaving the most.

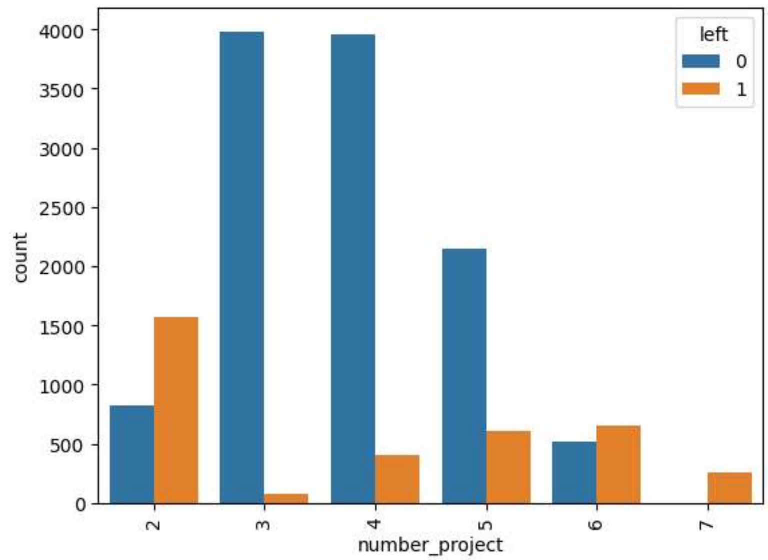
```
plt.figure(figsize=(8,8))
sns.countplot(x="Work_accident",hue="left",data=df )
plt.xticks(rotation=90)
```

```
(array([0, 1]), [Text(0, 0, '0'), Text(1, 0, '1')])
```




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

```
(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 2 projects left the company with the maximum amount.

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

```
dfclus
```

	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

```
from sklearn.cluster import KMeans

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```

```
label
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int32)
```

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

	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

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

	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

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

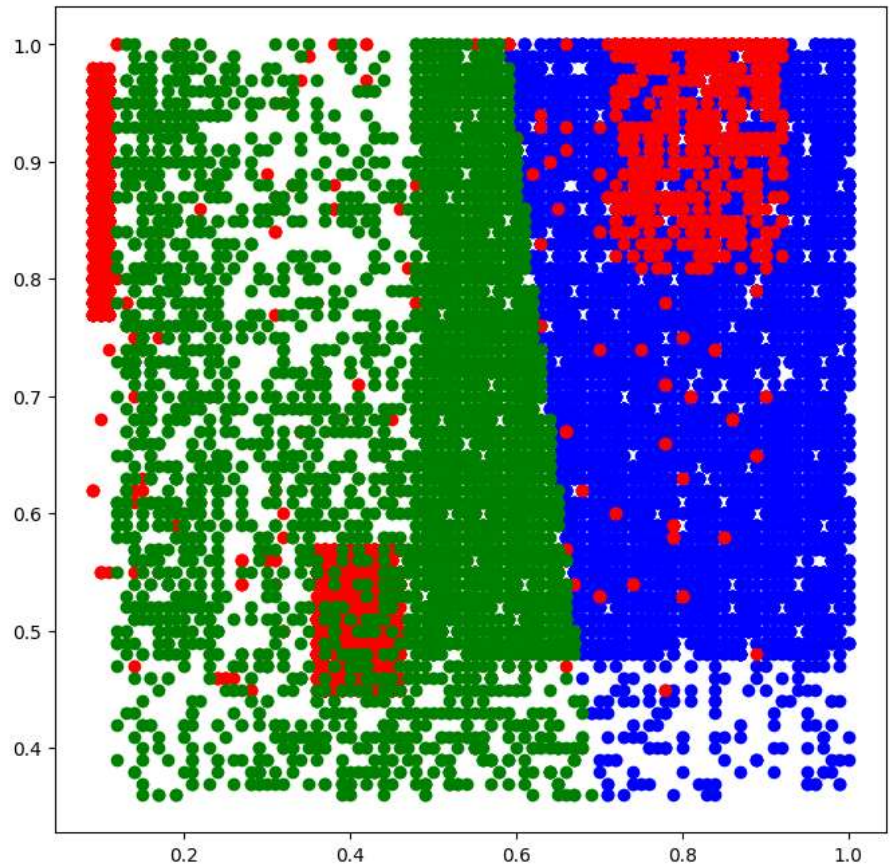
	satisfaction_level	last_evaluation	left
count	4708.000000	4708.000000	4708.0
mean	0.457984	0.680854	0.0
std	0.153456	0.165609	0.0
min	0.120000	0.360000	0.0

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

```
array([0.67, 0.82, 0.91, ..., 0.55, 0.95, 0.54])
```

```
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")
```

```
<matplotlib.collections.PathCollection at 0x7c32750e9870>
```



The Blue Cluster: Represents the people with the best satisfaction levels and they scored high in the last evaluation.

The Red Cluster: Represents the people with medium satisfaction levels and they scored high in the last evaluation.

The Green Cluster: Represents the people with lower satisfaction levels and scored fairly comparatively.

```
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
```

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

Converting categorical data into numerical data using One Hot Encoding

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

```
df_converted.head()
```

	sales_IT	sales_RandD	sales_accounting	sales_hr	sales_management	sales_marketing
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0

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

```
dfn.shape
```

```
(14999, 21)
```

```
dfn.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spen
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	

5 rows × 21 columns

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

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

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

```
xtrain.shape,ytrain.shape,xtest.shape,ytest.shape
```

```
((11999, 20), (11999,), (3000, 20), (3000,))
```

```
ytrain.value_counts()
```

```
0    9137
1    2862
Name: left, dtype: int64
```

Since the data is highly imbalanced (because of the lower record of people who left relatively to those who didn't leave) for further training the dataset, SMOTE would be used to balance the data for those who left.

```
from imblearn.over_sampling import SMOTE
```

```
sm = SMOTE(random_state=2)
xtrainres, ytrainres = sm.fit_resample(xtrain, ytrain)
```

```
ytrainres.value_counts()
```

```
0    9137
1    9137
Name: left, dtype: int64
```

```
xtrainres.value_counts()
```

```
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
0.460000            0.570000        2               139                   3                   0              0                      0
0           0                 0         0                0                 0                  1            0              0
0           1           0               14
0.440000            0.450000        2               156                   3                   0              0                      0
0           0                 0         0                0                 0                  1            0              0
0           0           1               11
0.450000            0.530000        2               155                   3                   0              0                      1
0           0                 0         0                0                 0                  0            0              0
0           1           0               10
0.410000            0.480000        2               136                   3                   0              0                      0
0           0                 0         0                1                 0                  0            0              0
0           0           1               10
0.370000            0.460000        2               156                   3                   0              0                      0
0           0                 0         0                0                 0                  1            0              0
0           1           0               10

..
0.430494            0.554395        2               158                   3                   0              0                      0
0           0                 0         0                0                 0                  1            0              0
0           1           0                1
0.430524            0.460000        2               149                   3                   0              0                      0
0           0                 0         0                0                 0                  0            0              1
0           0           0                1
0.430591            0.503375        2               154                   3                   0              0                      0
0           0                 0         0                0                 0                  0            1              0
0           0           1                1
0.430603            0.565179        2               129                   3                   0              0                      0
0           0                 0         0                0                 0                  1            0              0
0           0           1                1
1.000000            1.000000        5               142                   4                   0              0                      0
0           0                 0         0                0                 0                  1            0              0
0           1           0                1
Length: 14975, dtype: int64
```

```
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
```

```
logreg = LogisticRegression(solver='lbfgs', max_iter=1000)
```

```
print(cross_val_score(logreg, xtrainres, ytrainres, cv=5).mean())
```

```
0.8061742654827233
```

```
logreg.fit(xtrainres,ytrainres)
ypred = logreg.predict(xtest)
```

```
from sklearn.metrics import classification_report
```

Logistic Regression Report

```
metrics.confusion_matrix(ytest,ypred)
```

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

```
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
```

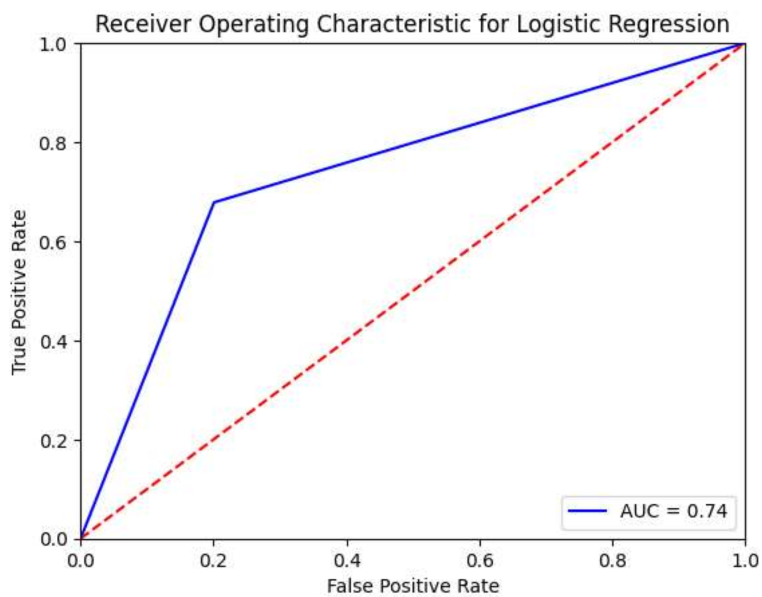
```
roc_auc_score(ytest,ypred)
```

```
0.7388173135941893
```

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

#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

```
randm=RandomForestClassifier(max_depth=5)
```

```
print(cross_val_score(randm, xtrainres, ytrainres, cv=5).mean())
```

```
0.9480683350592308
```

```
randm.fit(xtrainres,ytrainres)
ypred1=randm.predict(xtest)
```

Random Forest Classification Report

```
metrics.confusion_matrix(ytest,ypred1)
```

```
array([[2222,   69],
       [  56,  653]])
```

```
print(classification_report(ytest,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      2291
           1       0.90      0.92      0.91       709

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

```
roc_auc_score(ytest,ypred1)
```

```
0.9454488311717096
```

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

#plt
plt.title('Receiver Opertating Characteristics 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.         0.03011785 1.        ]
[0.         0.92101551 1.        ]
[2 1 0]
0.9454488311717096
```

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,max_depth=1, random_state=0)

print(cross_val_score(gb, xtrainres, ytrainres, cv=5).mean())
```

```
0.9478495915875037
```

```
gb.fit(xtrainres,ytrainres)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=1.0, max_depth=1, random_state=0)
```

```
ypred2 = gb.predict(xtest)
```

Gradient Boosting Classification Report

```
metrics.confusion_matrix(ytest,ypred2)
```

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

```
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
```

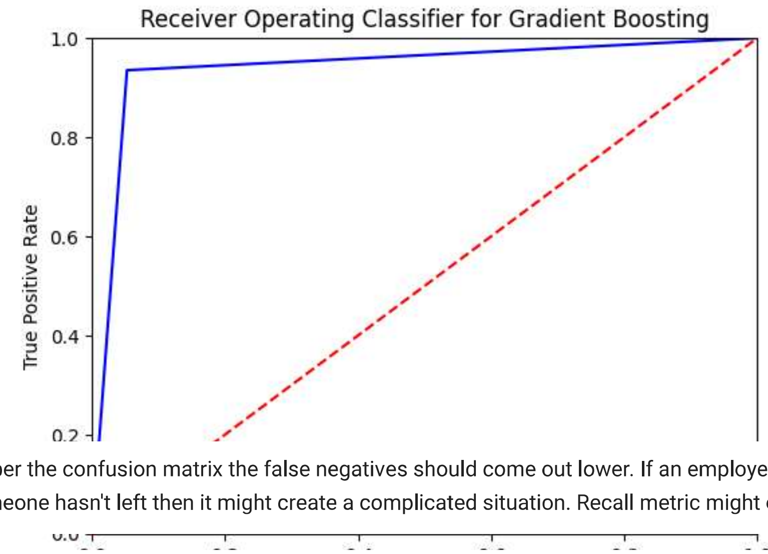
```
roc_auc_score(ytest,ypred2)
```

```
0.9413705066554046
```

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

#plt
plt.title('Receiver Operating Classifier 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
```



As per the confusion matrix the false negatives should come out lower. If an employee would leave the organization is misrepresented as someone hasn't left then it might create a complicated situation. Recall metric might can safegaurd the situation.

```
col = xtrainres.columns
```

```
col
```

```
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')
```

```
feature_labels = np.array(col)
```

```
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_accounting-0.02%
sales_technical-0.02%
sales_marketing-0.02%
sales_product_mng-0.03%
sales_sales-0.04%
sales_support-0.05%
sales_IT-0.05%
promotion_last_5years-0.12%
sales_RandD-0.20%
sales_management-0.20%
salary_medium-0.34%
salary_low-0.63%
salary_high-1.35%
Work_accident-3.42%
last_evaluation-10.38%
average_montly_hours-12.27%
number_project-18.27%
time_spend_company-25.09%
satisfaction_level-27.49%
```

The factors listed above are showcased in ascending order, where it can be observed that the employee's satisfaction level majorly influences the employee turnover rate. Improving the work culture to elevate the employee satisfaction could be a cue for lower employee turnover rates.

```
predict_probability =  randm.predict_proba(xtest)
```

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

```
array([0.06172975, 0.1218773 , 0.08789365, ..., 0.72357073, 0.07343516,
       0.12371904])
```

```
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")
```

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

```
colordict = dict(zip(categories, color))
```

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

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

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

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

```
clr.head(10)
```

	zone	probability	Color
0	Safe Zone	0.061730	Green
1	Safe Zone	0.121877	Green
2	Safe Zone	0.087894	Green
3	Safe Zone	0.077516	Green
4	Safe Zone	0.124883	Green
5	Safe Zone	0.077480	Green
6	High Risk Zone	0.943385	Red
7	Medium Risk Zone	0.751295	Orange
8	Safe Zone	0.116590	Green
9	Safe Zone	0.088607	Green

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

```
c = ["Greed","Red","Orange","Yellow"]
```

0s completed at 2:26 AM