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

```
trainhr=pd.read_csv('/content/train.csv')
```

```
testhr=pd.read_csv('/content/test.csv')
```

```
print(trainhr.shape)
print(testhr.shape)
```

```
(54808, 14)
(23490, 13)
```

```
trainhr.head()
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_serv
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5.0	
1	65141	Operations	region_22	Bachelor's	m	other	1	30	5.0	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3.0	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	1.0	
4	48945	Technology	region_26	Bachelor's	m	other	1	45	3.0	

```
testhr.head()
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_serv
0	8724	Technology	region_26	Bachelor's	m	sourcing	1	24	NaN	
1	74430	HR	region_4	Bachelor's	f	other	1	31	3.0	
2	72255	Sales & Marketing	region_13	Bachelor's	m	other	1	31	1.0	
3	38562	Procurement	region_2	Bachelor's	f	other	3	31	2.0	
4	64486	Finance	region_29	Bachelor's	m	sourcing	1	30	4.0	

```
trainhr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   employee_id           54808 non-null  int64  
 1   department            54808 non-null  object 
 2   region                54808 non-null  object 
 3   education             52399 non-null  object 
 4   gender                54808 non-null  object 
 5   recruitment_channel   54808 non-null  object 
 6   no_of_trainings       54808 non-null  int64  
 7   age                   54808 non-null  int64  
 8   previous_year_rating  50684 non-null  float64
 9   length_of_service     54808 non-null  int64  
 10  KPIs_met >80%         54808 non-null  int64  
 11  awards_won?           54808 non-null  int64  
 12  avg_training_score    54808 non-null  int64  
 13  is_promoted           54808 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.9+ MB
```

```
testhr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23490 entries, 0 to 23489
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   employee_id           23490 non-null  int64  
 1   department            23490 non-null  object 
 2   region                23490 non-null  object 
 3   education             22456 non-null  object 
 4   gender                23490 non-null  object 
 5   recruitment_channel   23490 non-null  object 
 6   no_of_trainings       23490 non-null  int64  
 7   age                   23490 non-null  int64  
 8   previous_year_rating  21678 non-null  float64
 9   length_of_service     23490 non-null  int64  
 10  KPIs_met >80%         23490 non-null  int64  
 11  awards_won?           23490 non-null  int64  
 12  avg_training_score    23490 non-null  int64  
dtypes: float64(1), int64(7), object(5)
memory usage: 2.3+ MB
```

```
testhr.describe().T
```

	count	mean	std	min	25%	50%	75%	max
employee_id	23490.0	39041.399149	22640.809201	3.0	19370.25	38963.5	58690.0	78295.0
no_of_trainings	23490.0	1.254236	0.600910	1.0	1.00	1.0	1.0	9.0
age	23490.0	34.782929	7.679492	20.0	29.00	33.0	39.0	60.0
previous_year_rating	21678.0	3.339146	1.263294	1.0	3.00	3.0	4.0	5.0
length_of_service	23490.0	5.810387	4.207917	1.0	3.00	5.0	7.0	34.0
KPIs_met >80%	23490.0	0.358834	0.479668	0.0	0.00	0.0	1.0	1.0
awards_won?	23490.0	0.022776	0.149191	0.0	0.00	0.0	0.0	1.0
avg_training_score	23490.0	63.263133	13.411750	39.0	51.00	60.0	76.0	99.0

```
trainhr.describe().T
```

	count	mean	std	min	25%	50%	75%	max
employee_id	54808.0	39195.830627	22586.581449	1.0	19669.75	39225.5	58730.5	78298.0
no_of_trainings	54808.0	1.253011	0.609264	1.0	1.00	1.0	1.0	10.0
age	54808.0	34.803915	7.660169	20.0	29.00	33.0	39.0	60.0
previous_year_rating	50684.0	3.329256	1.259993	1.0	3.00	3.0	4.0	5.0
length_of_service	54808.0	5.865512	4.265094	1.0	3.00	5.0	7.0	37.0
KPIs_met >80%	54808.0	0.351974	0.477590	0.0	0.00	0.0	1.0	1.0
awards_won?	54808.0	0.023172	0.150450	0.0	0.00	0.0	0.0	1.0
avg_training_score	54808.0	63.386750	13.371559	39.0	51.00	60.0	76.0	99.0
is_promoted	54808.0	0.085170	0.279137	0.0	0.00	0.0	0.0	1.0

```
trainhr.isnull().sum()
```

```
employee_id              0
department               0
region                   0
education             2409
gender                   0
recruitment_channel      0
no_of_trainings          0
age                      0
previous_year_rating  4124
length_of_service        0
KPIs_met >80%            0
awards_won?              0
avg_training_score       0
is_promoted              0
dtype: int64
```

```
#testhr.isnull().any()
testhr.isnull().sum()/len(testhr)
```

```
employee_id            0.000000
department             0.000000
region                 0.000000
education              0.044019
gender                 0.000000
recruitment_channel    0.000000
no_of_trainings        0.000000
age                    0.000000
previous_year_rating   0.077139
length_of_service      0.000000
KPIs_met >80%          0.000000
awards_won?            0.000000
avg_training_score     0.000000
dtype: float64
```

```
testhr['is_promoted']='test'
```

```
combineddf=pd.concat([trainhr,testhr],axis=0)
#axis 0 for row
#axis 1 for column
```

```
combineddf.isnull().sum()
```

```
employee_id               0
department                0
region                    0
education              3443
gender                    0
recruitment_channel       0
no_of_trainings           0
age                       0
previous_year_rating   5936
length_of_service         0
KPIs_met >80%             0
awards_won?               0
avg_training_score        0
is_promoted               0
dtype: int64
```

```
combineddf.education.value_counts(dropna=False)
```

```
Bachelor's          52247
Master's & above    21429
NaN                  3443
Below Secondary      1179
Name: education, dtype: int64
```

```
combineddf.education.fillna("Bachelor's",inplace=True)
```

```
combineddf.previous_year_rating.value_counts(dropna=False)
```

```
3.0    26539
5.0    16838
4.0    14126
1.0     8903
2.0     5956
NaN     5936
Name: previous_year_rating, dtype: int64
```

```
combineddf.previous_year_rating.mean()
```

```
3.3322185677565574
```

```
combineddf.previous_year_rating.median()
```

```
3.0
```

```
combineddf.previous_year_rating.fillna(3.0,inplace=True)
```

```
combineddf.isnull().any()
```

```
employee_id           False
department            False
region                False
education             False
gender                False
recruitment_channel   False
no_of_trainings       False
age                   False
previous_year_rating  False
length_of_service     False
KPIs_met >80%         False
awards_won?           False
avg_training_score    False
is_promoted           False
dtype: bool
```

```
combineddf.no_of_trainings=combineddf.no_of_trainings.astype('object')
```

```
combineddf.previous_year_rating=combineddf.previous_year_rating.astype('object')
```

```
combineddf['KPIs_met >80%']=combineddf['KPIs_met >80%'].astype('object')
```

```
combineddf['awards_won?']=combineddf['awards_won?'].astype('object')
```

```
combineddf=combineddf.drop('employee_id',axis=1)
```

```
numcols=combineddf.select_dtypes(include=np.number)
objcols=combineddf.select_dtypes(include=['object'])
```

```
numcols.head()
```

	age	length_of_service	avg_training_score
0	35	8	49
1	30	4	60
2	34	7	50
3	39	10	50
4	45	2	73

```
objcols.head()
```

	department	region	education	gender	recruitment_channel	no_of_trainings	previous_year_rating	KPIs_met >80%	awards_won?	is_promote
0	Sales & Marketing	region_7	Master's & above	f	sourcing	1	5.0	1	0	
1	Operations	region_22	Bachelor's	m	other	1	5.0	0	0	
2	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	3.0	0	0	

```
numcols.columns
```

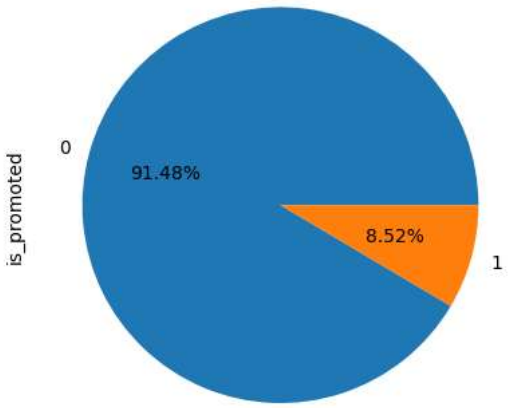
```
Index(['age', 'length_of_service', 'avg_training_score'], dtype='object')
```

```
objcols.columns
```

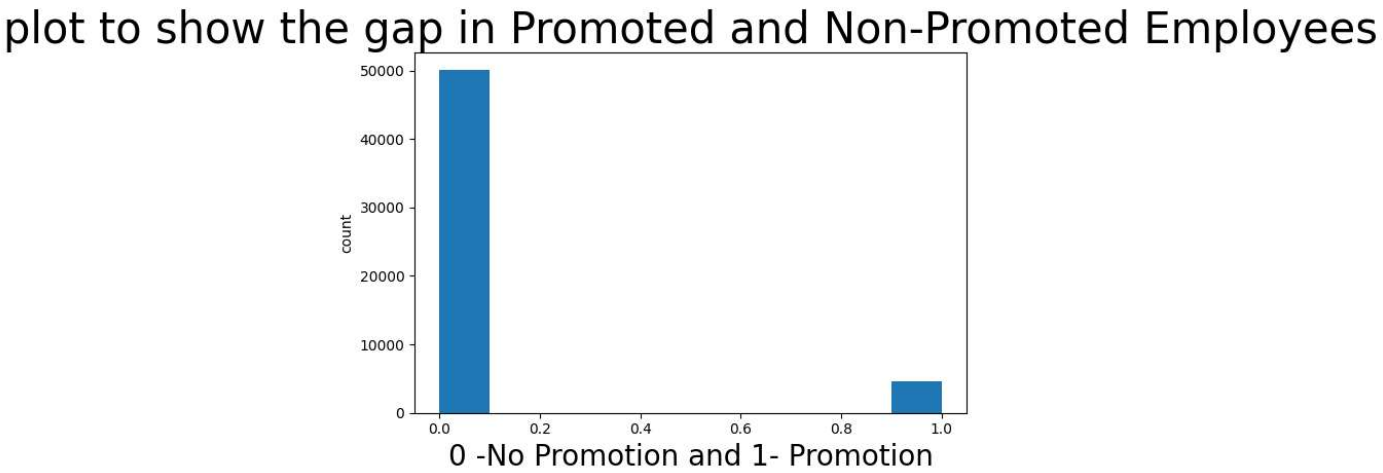
```
Index(['department', 'region', 'education', 'gender', 'recruitment_channel',
       'no_of_trainings', 'previous_year_rating', 'KPIs_met >80%',
       'awards_won?', 'is_promoted'],
      dtype='object')
```

```
trainhr.is_promoted.value_counts().plot(kind='pie',autopct='%0.2f%%')
```

```
<Axes: ylabel='is_promoted'>
```




```
#plotting a scatter plot
plt.hist(trainhr.is_promoted)
plt.title('plot to show the gap in Promoted and Non-Promoted Employees', fontsize = 30)
plt.xlabel('0 -No Promotion and 1- Promotion', fontsize = 20)
plt.ylabel('count')
plt.tight_layout()
plt.show()
```

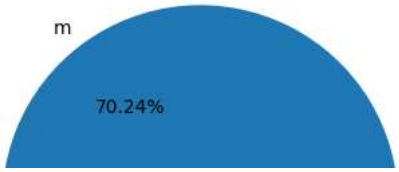



```
trainhr.gender.value_counts()
```

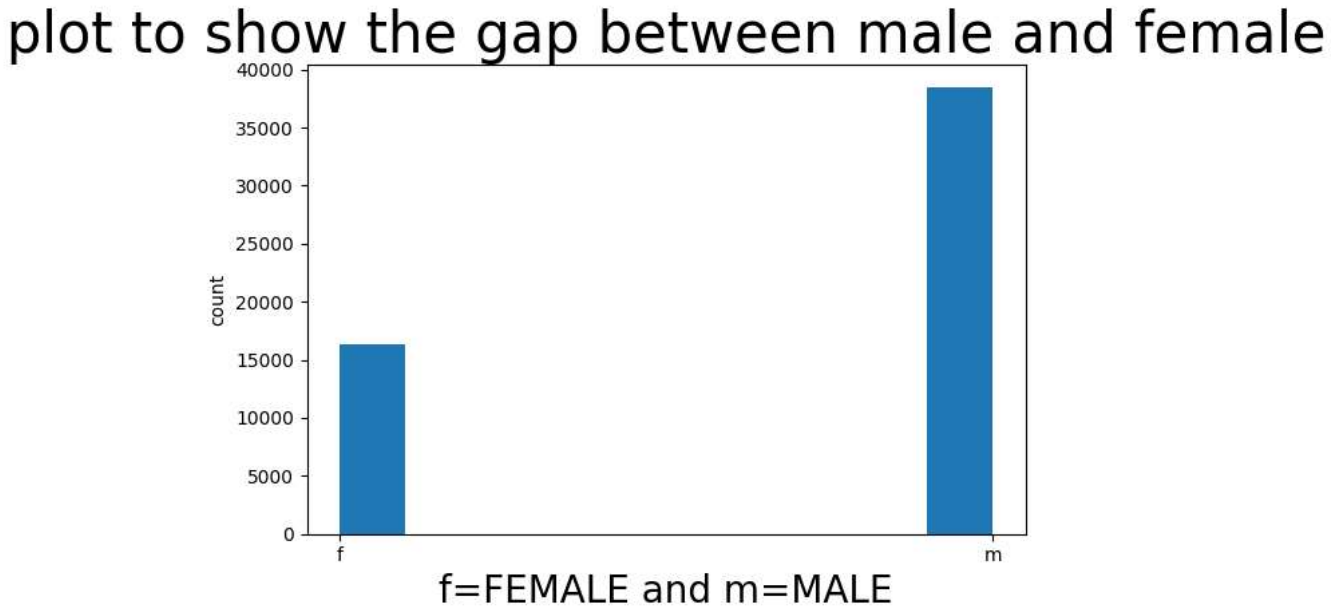
```
m    38496
f    16312
Name: gender, dtype: int64
```

```
trainhr.gender.value_counts().plot(kind='pie',autopct='%0.2f%%')
```

```
<Axes: ylabel='gender'>
```

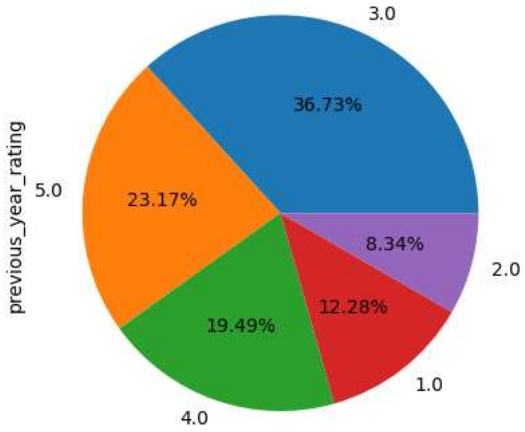



```
#plotting a scatter plot
plt.hist(trainhr.gender)
plt.title('plot to show the gap between male and female', fontsize = 30)
plt.xlabel('f=FEMALE and m=MALE', fontsize = 20)
plt.ylabel('count')
plt.tight_layout()
plt.show()
```



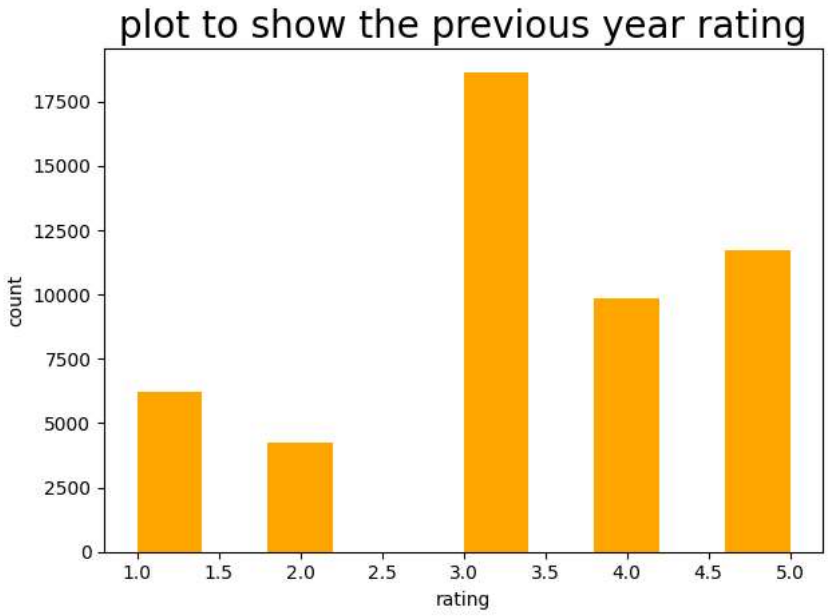

```
trainhr.previous_year_rating.value_counts().plot(kind='pie',autopct='%0.2f%%')
```

```
<Axes: ylabel='previous_year_rating'>
```



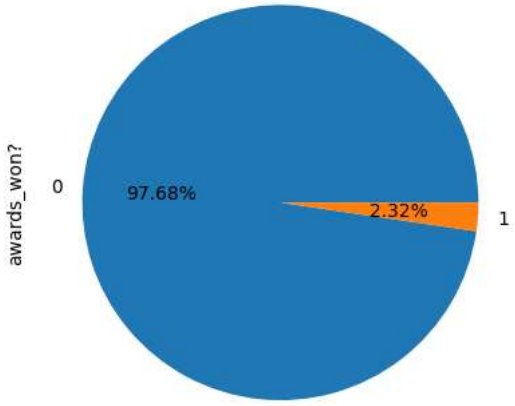

```
#plotting a scatter plot

plt.hist(trainhr.previous_year_rating,color='orange')
plt.title('plot to show the previous year rating', fontsize = 20)
plt.xlabel('rating', fontsize = 10)
```

```
plt.ylabel('count')
plt.tight_layout()
plt.show()
```

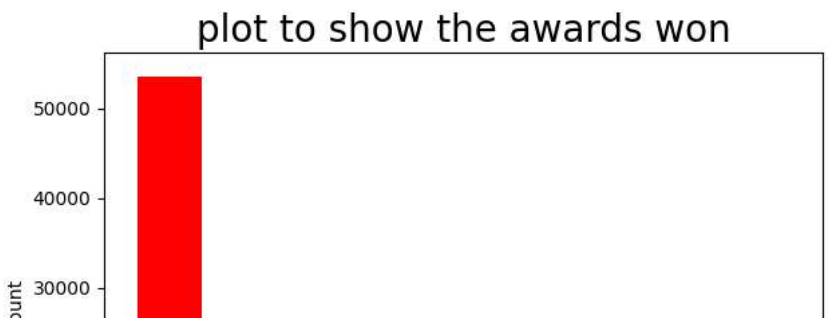



```
trainhr['awards_won?'].value_counts().plot(kind='pie',autopct='%0.2f%%')
```

```
<Axes: ylabel='awards_won?'>
```

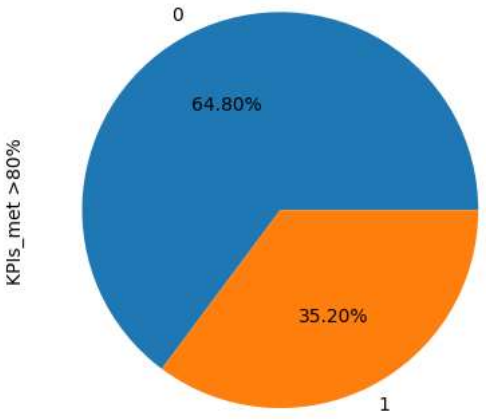



```
#plotting a scatter plot
plt.hist(trainhr['awards_won?'],color='red')
plt.title('plot to show the awards won', fontsize = 20)
plt.xlabel('0=Awards Won and 1=NO Awards Won', fontsize = 10)
plt.ylabel('count')
plt.tight_layout()
plt.show()
```

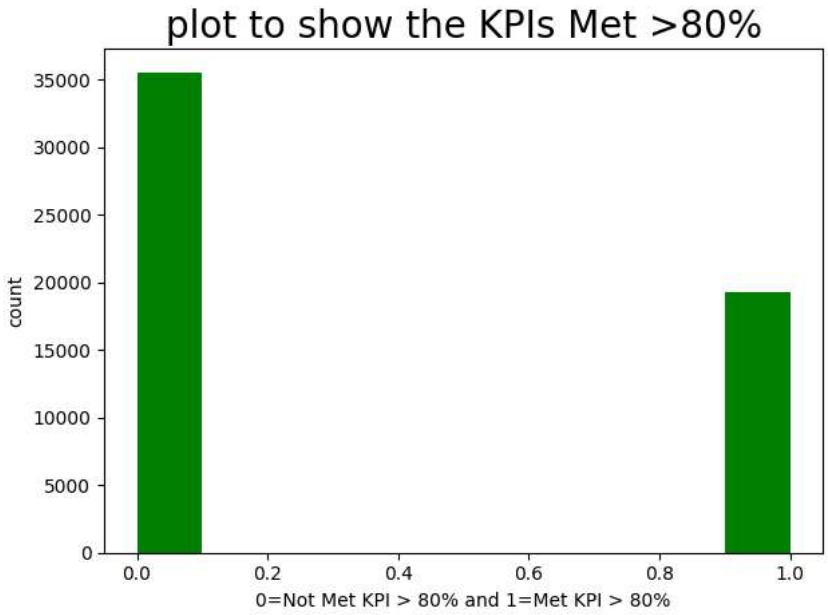



```
trainhr['KPIs_met >80%'].value_counts().plot(kind='pie',autopct='%0.2f%%')
```

```
<Axes: ylabel='KPIs_met >80%'>
```

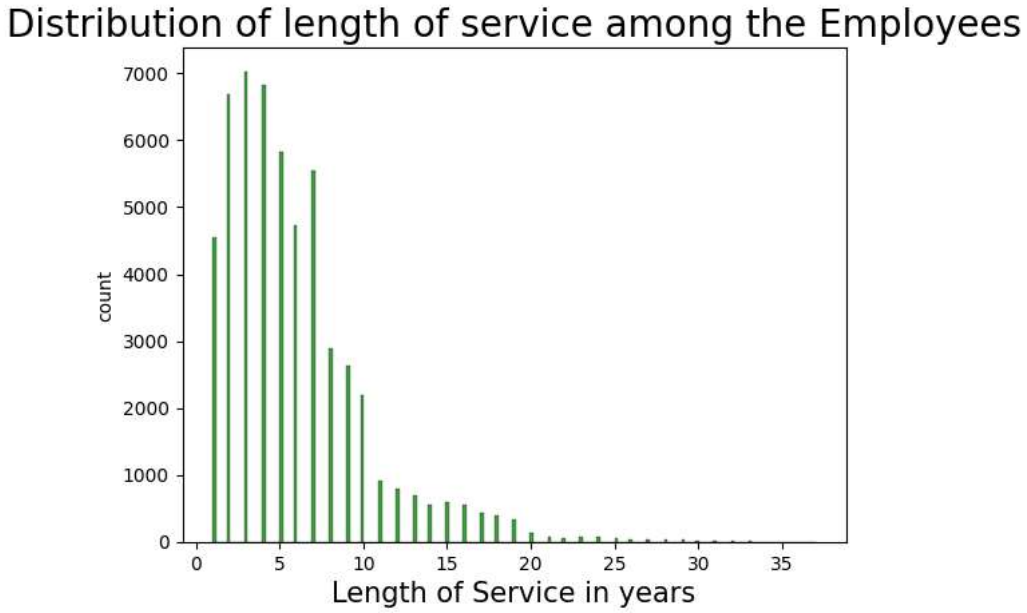



```
#plotting a scatter plot
plt.hist(trainhr['KPIs_met >80%'],color='green')
plt.title('plot to show the KPIs Met >80%', fontsize = 20)
plt.xlabel('0=Not Met KPI > 80% and 1=Met KPI > 80%', fontsize = 10)
plt.ylabel('count')
plt.tight_layout()
plt.show()
```

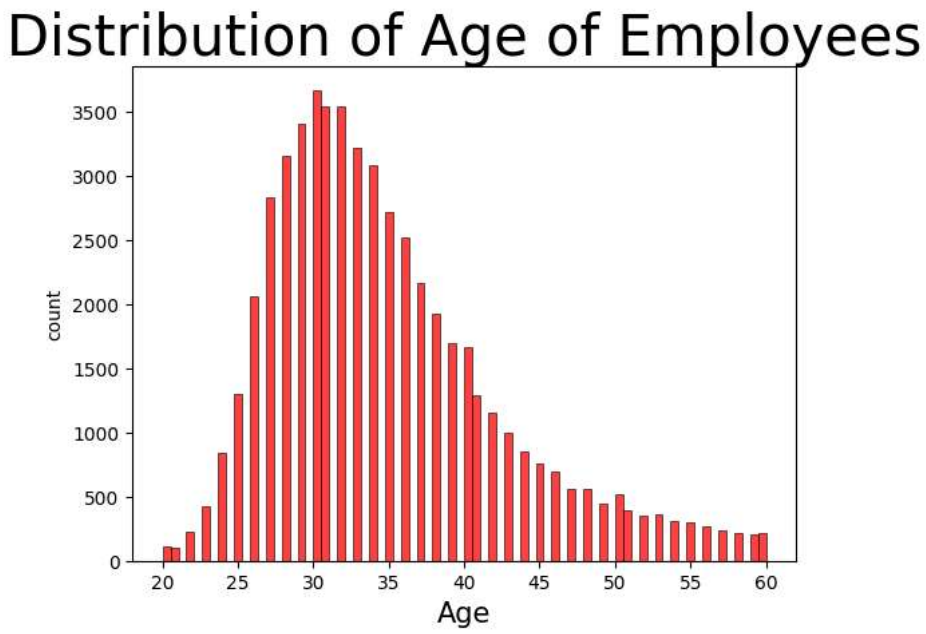



```
# checking the distribution of length of service
sns.histplot(trainhr['length_of_service'], color = 'green')
plt.title('Distribution of length of service among the Employees', fontsize = 20)
```

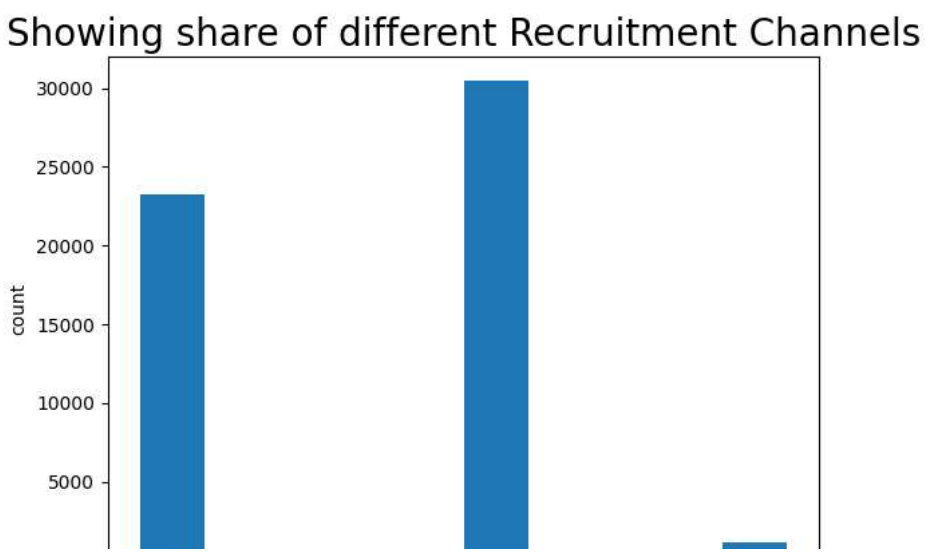
```
plt.xlabel('Length of Service in years', fontsize = 15)
plt.ylabel('count')
plt.show()
```




```
# checking the distribution of age of Employees in the company
sns.histplot(trainhr['age'], color = 'red')
plt.title('Distribution of Age of Employees', fontsize = 30)
plt.xlabel('Age', fontsize = 15)
plt.ylabel('count')
plt.show()
```

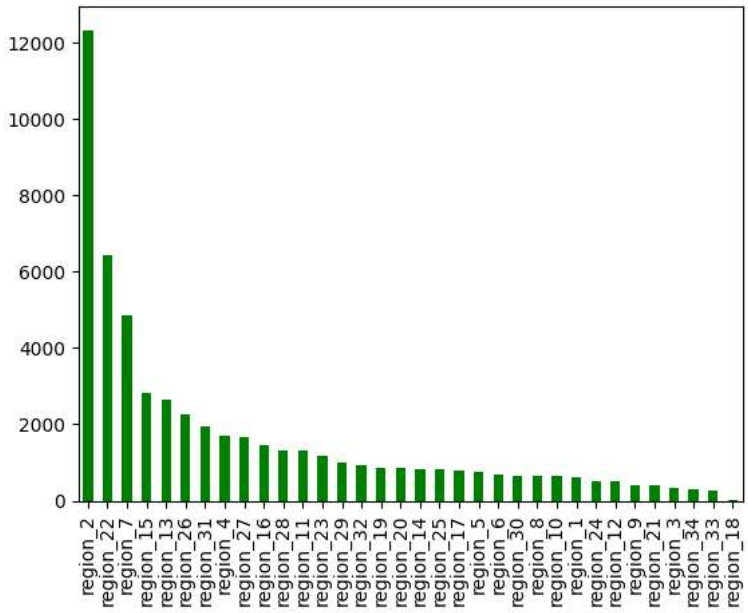



```
#plotting a scatter plot
plt.hist(trainhr['recruitment_channel'])
plt.title('Showing share of different Recruitment Channels', fontsize = 20)
plt.xlabel('Others, Sourcing, Refered', fontsize = 10)
plt.ylabel('count')
plt.tight_layout()
plt.show()
```




```
trainhr['region'].value_counts().sort_values(ascending=False).plot(kind='bar',color='green')
```

```
<Axes: >
```



Scaling the numeric columns

```
#standard scaler
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()

numcols_scaled=scaler.fit_transform(numcols)

numcols_scaled=pd.DataFrame(numcols_scaled,columns=numcols.columns)

numcols_scaled.head()
```

	age	length_of_service	avg_training_score
0	0.026400	0.506357	-1.072184
1	-0.625841	-0.435253	-0.250282

```
from sklearn.preprocessing import MinMaxScaler

minmax=MinMaxScaler()

numcols_minmax=minmax.fit_transform(numcols)

numcols_minmax=pd.DataFrame(numcols_minmax,columns=numcols.columns)

numcols_minmax.head()
```

	age	length_of_service	avg_training_score
0	0.375	0.194444	0.166667
1	0.250	0.083333	0.350000
2	0.350	0.166667	0.183333
3	0.475	0.250000	0.183333
4	0.625	0.027778	0.566667

```
from sklearn.preprocessing import RobustScaler

robust=RobustScaler()

numcols_robust=robust.fit_transform(numcols)

numcols_robust=pd.DataFrame(numcols_robust,columns=numcols.columns)

numcols_robust.head()
```

	age	length_of_service	avg_training_score
0	0.2	0.75	-0.44
1	-0.3	-0.25	0.00
2	0.1	0.50	-0.40
3	0.6	1.25	-0.40
4	1.2	-0.75	0.52

Dummy encoding

```
objcols.columns
```

```
Index(['department', 'region', 'education', 'gender', 'recruitment_channel',
       'no_of_trainings', 'previous_year_rating', 'KPIs_met >80%',
       'awards_won?', 'is_promoted'],
      dtype='object')
```

```
objcols_dummy=pd.get_dummies(objcols,columns=['department', 'region', 'education', 'gender', 'recruitment_channel',
       'no_of_trainings', 'previous_year_rating', 'KPIs_met >80%', 'awards_won?'])

objcols_dummy.head()
```

	is_promoted	department_Analytics	department_Finance	department_HR	department_Legal	department_Operations	department_Procurement
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

5 rows × 71 columns

```
combinedf_clean=pd.concat([numcols_minmax.reset_index(),objcols_dummy.reset_index()],axis=1)
```

```
combinedf_clean.head()
```

	index	age	length_of_service	avg_training_score	index	is_promoted	department_Analytics	department_Finance	department_HR	depa
0	0	0.375	0.194444	0.166667	0	0	0	0	0	
1	1	0.250	0.083333	0.350000	1	0	0	0	0	
2	2	0.350	0.166667	0.183333	2	0	0	0	0	
3	3	0.475	0.250000	0.183333	3	0	0	0	0	
4	4	0.625	0.027778	0.566667	4	0	0	0	0	

5 rows × 76 columns

```
combinedf_clean.drop(['index'],axis=1,inplace=True)
```

```
#split data back to train and test
trainhr_df=combinedf_clean[combinedf_clean.is_promoted!='test']
testhr_df=combinedf_clean[combinedf_clean.is_promoted=='test']
```

```
#split data back to train and test for EDA Hypothesis testing as this dataset doesn't have scaled numerical variables
#and dummies in categorical columns.
trainhr_EDA=combineddf[combineddf.is_promoted!='test']
testhr_EDA=combineddf[combineddf.is_promoted=='test']
```

```
pd.crosstab(trainhr_EDA.is_promoted,trainhr_EDA.recruitment_channel)
```

recruitment_channel	other	referred	sourcing
is_promoted			
0	27890	1004	21246
1	2556	138	1974

There is a significant number of promotions came from the "sourcing" channel.

```
pd.crosstab(trainhr_EDA.is_promoted,trainhr_EDA['KPIs_met >80%'])
```

KPIs_met >80%	0	1
is_promoted		
0	34111	16029
1	1406	3262

Again Having a good KPI score increases the chances of getting promoted in the company.

```
pd.crosstab(trainhr_EDA.is_promoted,trainhr_EDA['awards_won?'])
```

awards_won?	0	1
is_promoted		
0	49429	711

There is a very good chance of getting promoted if the employee has won an award

```
pd.crosstab(trainhr_EDA.age, trainhr_EDA['is_promoted'])
```

is_promoted	0	1
age		
20	109	4
21	93	5

This is Very Impressive that the company promotes employees of all the ages equally even the freshers have equal share of promotion and also the senior citizen employees are getting the equal share of Promotion in the Company

```
pd.crosstab(trainhr_EDA.department,trainhr_EDA['is_promoted'])
```

is_promoted	0	1
department		
Analytics	4840	512
Finance	2330	206
HR	2282	136
Legal	986	53
Operations	10325	1023
Procurement	6450	688
R&D	930	69
Sales & Marketing	15627	1213
Technology	6370	768

Again, Each of the departments have equal no. of promotions showing an equal development in each of the departments of the company.

```
pd.crosstab(trainhr_EDA.gender,trainhr_EDA['is_promoted'])
```

is_promoted	0	1
gender		
f	14845	1467
m	35295	3201

The above plot shows that there is no partiality between males and females in terms of promotion

```
#GROUPBY
```

```
trainhr_EDA.avg_training_score.groupby(trainhr_EDA.is_promoted).mean()
```

```
is_promoted
0    62.647686
1    71.325193
Name: avg_training_score, dtype: float64
```

```
trainhr_EDA.length_of_service.groupby(trainhr_EDA.is_promoted).mean()
```

```
is_promoted
0    5.879398
1    5.716367
Name: length_of_service, dtype: float64
```

```
trainhr_EDA.age.groupby(trainhr_EDA.is_promoted).mean()
```

```
is_promoted
0    34.844037
1    34.372965
Name: age, dtype: float64
```

```
#performing hypothesis testing using T-Test
one=trainhr_EDA[trainhr_EDA.is_promoted==1]
```

```
zero=trainhr_EDA[trainhr_EDA.is_promoted==0]


#Null hypothesis - There is no significant difference between the two given variables. (p-values greater than 0.05)

#Alternative Hypothesis - There is significant difference between the two given variables.(p-values less than 0.05)


from scipy.stats import ttest_ind
ttest_ind(one.avg_training_score,zero.avg_training_score,equal_var=False)
#since pvalue=7.662329172468838e-291 is lessthan 0.05,reject null
```

```
TtestResult(statistic=38.82675007357188, pvalue=7.662329172468838e-291, df=5363.307824110073)
```

```
ttest_ind(one.length_of_service,zero.length_of_service,equal_var=False)
#since pvalue=0.008262946987836755 is lessthan 0.05,reject null
```

```
TtestResult(statistic=-2.6420549711884886, pvalue=0.008262946987836755, df=5708.750739466446)
```

```
trainhr_EDA.avg_training_score.groupby(trainhr_EDA.previous_year_rating).mean()
```

```
previous_year_rating
1.0    60.064760
2.0    61.924024
3.0    64.045423
4.0    64.119773
5.0    63.781364
Name: avg_training_score, dtype: float64
```

```
ones=trainhr_EDA[trainhr_EDA.previous_year_rating==1.0]
two=trainhr_EDA[trainhr_EDA.previous_year_rating==2.0]
three=trainhr_EDA[trainhr_EDA.previous_year_rating==3.0]
four=trainhr_EDA[trainhr_EDA.previous_year_rating==4.0]
five=trainhr_EDA[trainhr_EDA.previous_year_rating==5.0]


from scipy.stats import f_oneway
f_oneway(ones.avg_training_score,two.avg_training_score,three.avg_training_score,four.avg_training_score,five.avg_training_s
#since pvalue=6.957044805766572e-114) is lessthan 0.05,reject null
```

```
F_onewayResult(statistic=133.71725352260415, pvalue=6.957044805766572e-114)
```

```
f_oneway(ones.length_of_service,two.length_of_service,three.length_of_service,four.length_of_service,five.length_of_service)
#since pvalue=2.89045896511553e-124 is lessthan 0.05,reject null
```

```
F_onewayResult(statistic=145.83450031447595, pvalue=2.89045896511553e-124)
```

```
from scipy.stats import chi2_contingency
chi2_contingency(pd.crosstab(trainhr.is_promoted,trainhr.gender))
#since pvalue=0.009765091521176657 is lessthan 0.05,reject null
```

```
Chi2ContingencyResult(statistic=6.677254566546107, pvalue=0.009765091521176657, dof=1, expected_freq=array([[14922.70617428,
35217.29382572],
       [ 1389.29382572,  3278.70617428]]))
```

```
chi2_contingency(pd.crosstab(trainhr['KPIs_met >80%'],trainhr.is_promoted))
#since pvalue=0.0 is lessthan 0.05,reject null
```

```
Chi2ContingencyResult(statistic=2689.3220548467057, pvalue=0.0, dof=1, expected_freq=array([[32492.01539921,  3024.98460079],
       [17647.98460079,  1643.01539921]]))
```

```
chi2_contingency(pd.crosstab(trainhr.is_promoted,trainhr['awards_won?']))
#since pvalue=0.0 is lessthan 0.05,reject null
```

```
Chi2ContingencyResult(statistic=2098.0719210465427, pvalue=0.0, dof=1, expected_freq=array([[48978.16596117,  1161.83403883],
       [ 4559.83403883,   108.16596117]]))
```

```
print(trainhr_EDA.shape)
print(testhr_EDA.shape)
```

```
(54808, 13)
(23490, 13)
```

```
y=trainhr_df.is_promoted
X=trainhr_df.drop('is_promoted',axis=1)
testhr_df=testhr_df.drop('is_promoted',axis=1)
```

```
X.head()
```

	age	length_of_service	avg_training_score	department_Analytics	department_Finance	department_HR	department_Legal	department_Op
0	0.375	0.194444	0.166667	0	0	0	0	
1	0.250	0.083333	0.350000	0	0	0	0	
2	0.350	0.166667	0.183333	0	0	0	0	
3	0.475	0.250000	0.183333	0	0	0	0	
4	0.625	0.027778	0.566667	0	0	0	0	

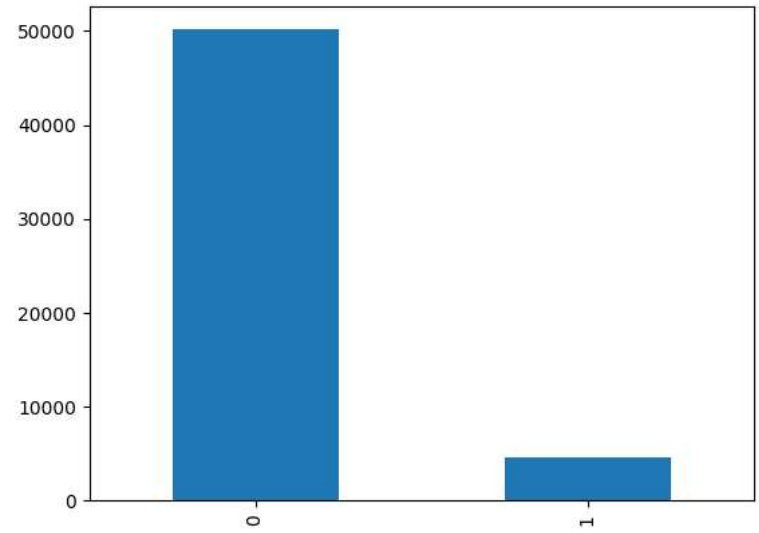
5 rows × 73 columns

```
# we using the labelencoder for dependent variable(y), to change dependent variable to 0's and 1's
from sklearn.preprocessing import LabelEncoder
```

```
y=LabelEncoder().fit_transform(y)
```

```
trainhr_df.is_promoted.value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
 # importing the smote as the data is been oversampled
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X,y = oversample.fit_resample(X, y)
```

Logistice Regression

```
from sklearn.linear_model import LogisticRegression
```

```
logit=LogisticRegression(max_iter=5000)
```

```
logitmodel=logit.fit(X,y)
```

```
logitmodel.score(X,y)
```

```
0.8995213402473076
```

```
logitpredict=logitmodel.predict(X)
```

```
pd.crosstab(y,logitpredict)
```

col_0	0	1
row_0		
0	46068	4072
1	6004	44136

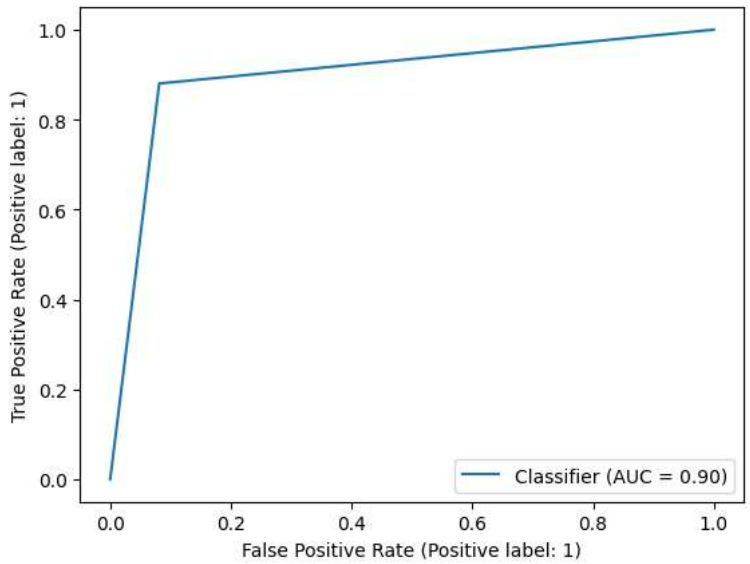
```
from sklearn.metrics import classification_report,RocCurveDisplay
from sklearn.model_selection import cross_val_score
```

```
print(classification_report(y,logitpredict))
```

```
              precision    recall  f1-score   support

           0       0.88      0.92      0.90     50140
           1       0.92      0.88      0.90     50140

    accuracy                           0.90    100280
   macro avg       0.90      0.90      0.90    100280
weighted avg       0.90      0.90      0.90    100280
```

```
RocCurveDisplay.from_predictions(y,logitpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7cff2f9450>
```




```
cross_val_score(logit,X,y)
```

```
array([0.79741723, 0.91663343, 0.918927  , 0.91852812, 0.91912645])
```

```
np.mean([0.79741723, 0.91663343, 0.918927  , 0.91852812, 0.91912645])
```

```
0.8941264459999999
```

```
logittest=logitmodel.predict(testhr_df)
```

```
pd.DataFrame(logittest).to_csv("Logit1.csv")
```

Decission Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree=DecisionTreeClassifier(max_depth=10)
```

```
treemodel=tree.fit(X,y)
```

```
treemodel.score(X,y)
```

```
0.8684383725568409
```

```
treepredict=treemodel.predict(X)
```

```
pd.crosstab(y,treepredict)
```

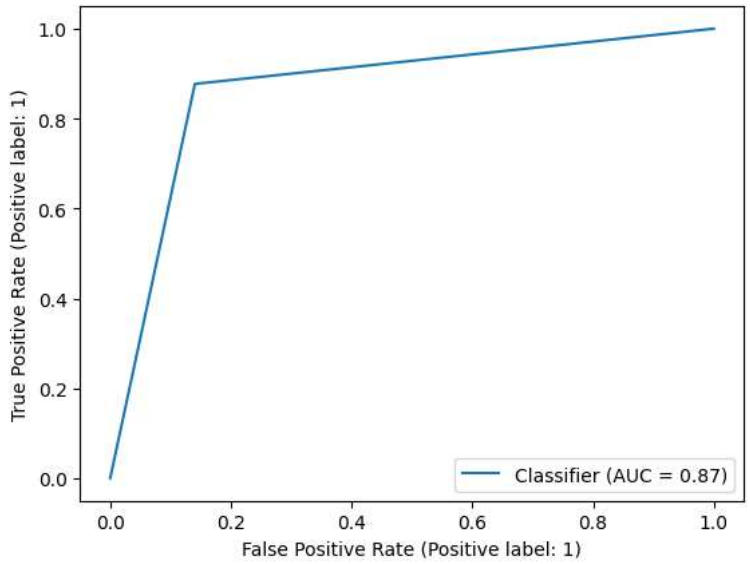
col_0	0	1
row_0		
0	43106	7034
1	6159	43981

```
print(classification_report(y,treepredict))
```

```
              precision    recall  f1-score   support

           0       0.87      0.86      0.87     50140
           1       0.86      0.88      0.87     50140

    accuracy                           0.87    100280
   macro avg       0.87      0.87      0.87    100280
weighted avg       0.87      0.87      0.87    100280
```

```
RocCurveDisplay.from_predictions(y,treepredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7d029e3d60>
```




```
cross_val_score(tree,X,y)
```

```
array([0.81367172, 0.87958716, 0.8718588 , 0.8723574 , 0.87345433])
```

```
np.mean([0.81367172, 0.87958716, 0.8718588 , 0.8723574 , 0.87345433])
```

```
0.862185882
```

```
treetest=treemodel.predict(testhr_df)
```

```
pd.DataFrame(treetest).to_csv('tree1.csv')
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
RF=RandomForestClassifier(max_depth=5,n_estimators=500)
```

```
RFmodel=RF.fit(X,y)
```

```
RFmodel.score(X,y)
```

```
0.7651076984443558
```

```
Ranpredict=RFmodel.predict(X)
```

```
pd.crosstab(y,Ranpredict)
```

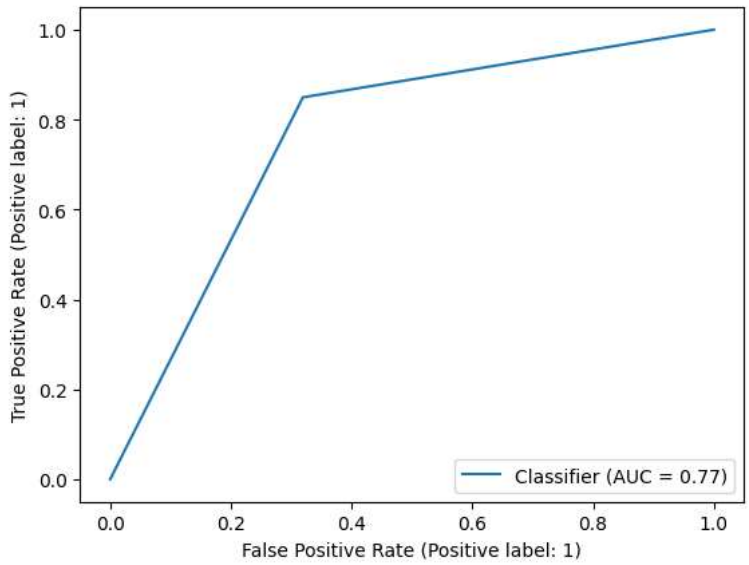
col_0	0	1
row_0		
0	34132	16008
1	7547	42593

```
print(classification_report(y,Ranpredict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.68      0.74     50140
           1       0.73      0.85      0.78     50140

    accuracy                           0.77    100280
   macro avg       0.77      0.77      0.76    100280
weighted avg       0.77      0.77      0.76    100280
```

```
RocCurveDisplay.from_predictions(y,Ranpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7cfc3cff70>
```



```
cross_val_score(RF,X,y)
```

```
array([0.76042082, 0.76695253, 0.75822696, 0.767501  , 0.76839848])
```

```
np.mean([0.76042082, 0.76695253, 0.75822696, 0.767501  , 0.76839848])
```

```
0.764299958
```

```
RFpredict=RFmodel.predict(testhr_df)
```

```
pd.DataFrame(RFpredict).to_csv('RF1.csv')
```

Gradient Boosting Machine

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
gbc=GradientBoostingClassifier(max_depth=4,n_estimators=300)
```

```
gbcmodel=gbc.fit(X,y)
```

```
gbcmodel.score(X,y)
```

```
0.9531711208615875
```

```
gbcpredict=gbcmodel.predict(X)
```

```
pd.crosstab(y,gbcpredict)
```

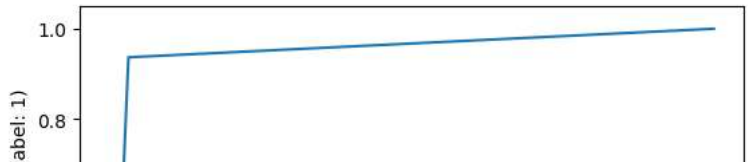
col_0	0	1
row_0		
0	48624	1516
1	3180	46960

```
print(classification_report(y,gbcpredict))
```

```
                precision    recall  f1-score   support

             0       0.94      0.97      0.95     50140
             1       0.97      0.94      0.95     50140

      accuracy                           0.95    100280
     macro avg       0.95      0.95      0.95    100280
  weighted avg       0.95      0.95      0.95    100280
```

```
RocCurveDisplay.from_predictions(y,gbcpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7cff009ae0>
```




```
cross_val_score(gbc,X,y)
```

```
array([0.8394994 , 0.96808935, 0.97402274, 0.97521939, 0.97252692])
```

```
np.mean([0.8394994 , 0.96808935, 0.97402274, 0.97521939, 0.97252692])
```

```
0.94587156
```

```
gbcpredict=gbcmodel.predict(testhr_df)
```

```
pd.DataFrame(gbcpredict).to_csv('gbc1.csv')
```

Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
```

```
mul=MultinomialNB()
```

```
mulmodel=mul.fit(X,y)
```

```
mulmodel.score(X,y)
```

```
0.7646190666134822
```

```
mulpredict=mulmodel.predict(X)
```

```
pd.crosstab(y,mulpredict)
```

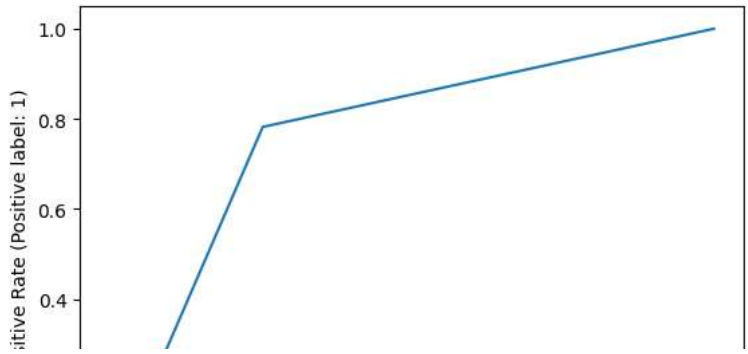
col_0	0	1
row_0		
0	37468	12672
1	10932	39208

```
print(classification_report(y,mulpredict))
```

```
                precision    recall  f1-score   support

             0       0.77      0.75      0.76     50140
             1       0.76      0.78      0.77     50140

      accuracy                           0.76    100280
     macro avg       0.76      0.76      0.76    100280
  weighted avg       0.76      0.76      0.76    100280
```

```
RocCurveDisplay.from_predictions(y,mulpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7cff267820>
```




```
cross_val_score(mul,X,y)
```

```
array([0.74217192, 0.76854807, 0.76386119, 0.77143997, 0.77059234])
```

```
np.mean([0.74217192, 0.76854807, 0.76386119, 0.77143997, 0.77059234])
```

```
0.763322698
```

```
mulpredict=mulmodel.predict(testhr_df)
```

```
pd.DataFrame(mulpredict).to_csv('mul1.csv')
```

Gaussian

```
from sklearn.naive_bayes import GaussianNB
```

```
nbg=GaussianNB()
```

```
nbgmodel=nbg.fit(X,y)
```

```
nbgmodel.score(X,y)
```

```
0.6562724371759074
```

```
nbgpredict=nbgmodel.predict(X)
```

```
pd.crosstab(y,nbgpredict)
```

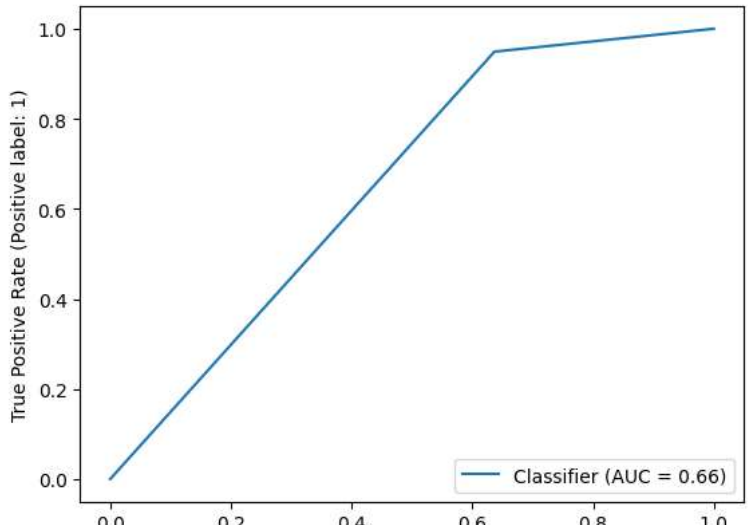
col_0	0	1
row_0		
0	18225	31915
1	2554	47586

```
print(classification_report(y,nbgpredict))
```

```
                precision    recall  f1-score   support

             0       0.88      0.36      0.51     50140
             1       0.60      0.95      0.73     50140

      accuracy                           0.66    100280
     macro avg       0.74      0.66      0.62    100280
  weighted avg       0.74      0.66      0.62    100280
```

```
RocCurveDisplay.from_predictions(y,nbgpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7d01c8f8e0>
```




```
cross_val_score(nbg,X,y)
```

```
array([0.62086159, 0.66229557, 0.66563622, 0.66169725, 0.66070004])
```

```
np.mean([0.62086159, 0.66229557, 0.66563622, 0.66169725, 0.66070004])
```

```
0.6542381340000001
```

```
nbgpredict=nbgmodel.predict(testhr_df)
```

```
pd.DataFrame(nbgpredict).to_csv('ngb1.csv')
```

Bernoulli's

```
from sklearn.naive_bayes import BernoulliNB
```

```
bb=BernoulliNB()
```

```
bbmodel=bb.fit(X,y)
```

```
bbmodel.score(X,y)
```

```
0.7433585959313921
```

```
bbpredict=bbmodel.predict(X)
```

```
pd.crosstab(y,bbpredict)
```

col_0	0	1
row_0		
0	36727	13413
1	12323	37817

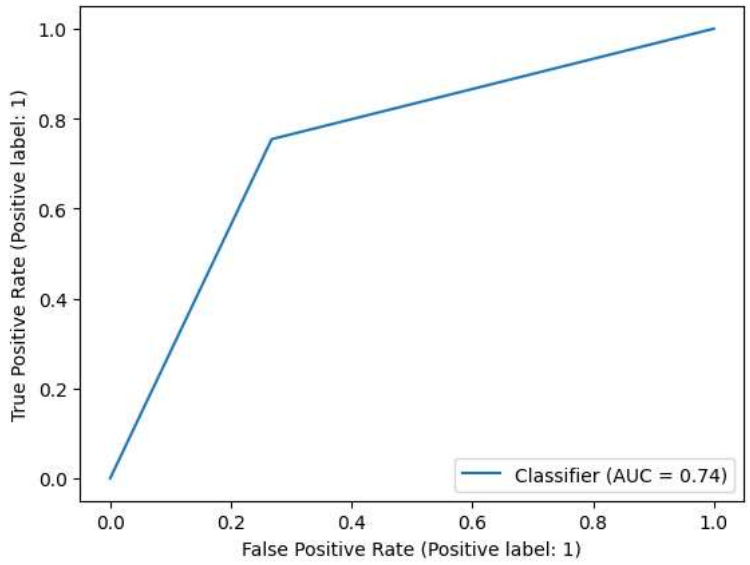
```
print(classification_report(y,bbpredict))
```

```
              precision    recall  f1-score   support

           0       0.75      0.73      0.74     50140
           1       0.74      0.75      0.75     50140

    accuracy                           0.74    100280
   macro avg       0.74      0.74      0.74    100280
```

```
weighted avg       0.74      0.74      0.74    100280
```

```
RocCurveDisplay.from_predictions(y,bbpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7cfefee320>
```




```
cross_val_score(bb,X,y)
```

```
array([0.73105305, 0.74406661, 0.74097527, 0.75054846, 0.7497507 ])
```

```
np.mean([0.73105305, 0.74406661, 0.74097527, 0.75054846, 0.7497507 ])
```

```
0.743278818
```

```
bbpredict=bbmodel.predict(testhr_df)
```

```
pd.DataFrame(bbpredict).to_csv('bb1.csv')
```

SVC

```
from sklearn.svm import SVC
```

```
svc=SVC(C=3)
```

```
svcmodel=svc.fit(X,y)
```

```
svcmodel.score(X,y)
```

```
0.9382030315117671
```

```
svcpredict=svcmodel.predict(X)
```

```
pd.crosstab(y,svcpredict)
```

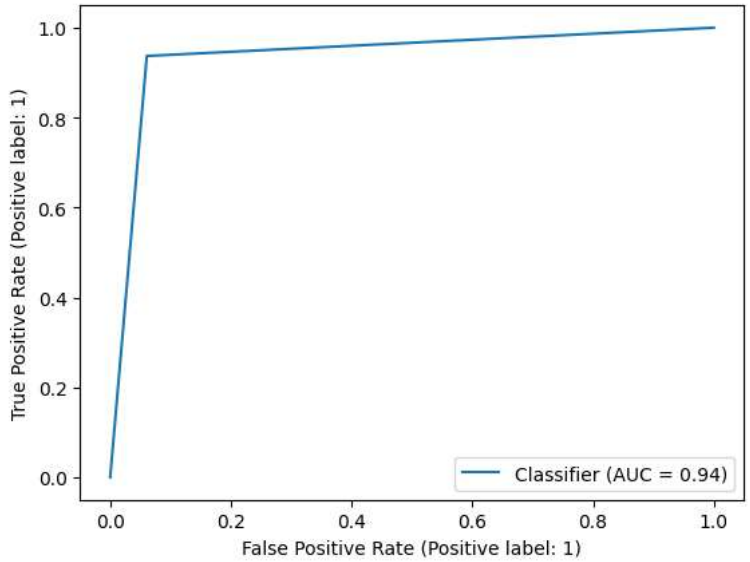
col_0	0	1
row_0		
0	47099	3041
1	3156	46984

```
print(classification_report(y,svcpredict))
```

```
                  precision    recall  f1-score   support

               0       0.94      0.94      0.94     50140
               1       0.94      0.94      0.94     50140

        accuracy                           0.94    100280
       macro avg       0.94      0.94      0.94    100280
    weighted avg       0.94      0.94      0.94    100280
```

```
RocCurveDisplay.from_predictions(y,svcpredict)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7d7d021ceb90>
```




```
cross_val_score(svc,X,y)
```

```
array([0.82977663, 0.9486438 , 0.9512864 , 0.95243319, 0.94934184])
```

```
np.mean([0.82977663, 0.9486438 , 0.9512864 , 0.95243319, 0.94934184])
```

```
0.926296372
```

```
svcpredict=svcmodel.predict(testhr_df)
```

```
pd.DataFrame(svcpredict).to_csv('svc1.csv')
```

Validating the Score in analytics vidya

Submitted at	Submission message	Score	Code File	Solution File	Final Solution
Wed, Nov-08-2023, 11:57:20 PM	SVM	0.437931034482759	Download	Download	○
Wed, Nov-08-2023, 11:56:41 PM	RF	0.309446254071661	Download	Download	○
Wed, Nov-08-2023, 11:55:45 PM	Gaussian	0.188826331745344	Download	Download	○
Wed, Nov-08-2023, 11:54:43 PM	Gradient Boosting Machine	0.468085106382979	Download	Download	○
Wed, Nov-08-2023, 11:52:40 PM	# Bernoulli's	0.298701298701299	Download	Download	○
Wed, Nov-08-2023, 11:44:48 PM	# Naive Bayes	0.303833145434047	Download	Download	○
Wed, Nov-08-2023, 11:37:01 PM	log	0.390961639516553	Download	Download	○
Sat, Nov-04-2023, 05:26:42 PM	gbc	0.485099337748344	Download	Download	○