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

%cd "C:\Users\Guruji\Videos\ML class\HR Analytics"

C:\Users\Guruji\Videos\ML class\HR Analytics

trainhr=pd.read_csv('train_LZdllcl.csv')

trainhr.head()

   employee_id         department     region         education gender
\
0        65438  Sales & Marketing   region_7  Master's & above      f

1        65141         Operations  region_22        Bachelor's      m

2         7513  Sales & Marketing  region_19        Bachelor's      m

3         2542  Sales & Marketing  region_23        Bachelor's      m

4        48945         Technology  region_26        Bachelor's      m


  recruitment_channel  no_of_trainings  age  previous_year_rating  \
0           sourcing                1   35                   5.0
1              other                1   30                   5.0
2           sourcing                1   34                   3.0
3              other                2   39                   1.0
4              other                1   45                   3.0

   length_of_service  KPIs_met >80%  awards_won?
avg_training_score  \
0                  8              1            0                  49

1                  4              0            0                  60

2                  7              0            0                  50

3                 10              0            0                  50

4                  2              0            0                  73


   is_promoted
0            0
1            0
2            0
```

```
3            0
4            0
```

```
testhr=pd.read_csv('test_2umaH9m.csv')
testhr.head()
```

```
   employee_id         department     region   education gender  \
0         8724         Technology  region_26  Bachelor's      m
1        74430                 HR   region_4  Bachelor's      f
2        72255  Sales & Marketing  region_13  Bachelor's      m
3        38562        Procurement   region_2  Bachelor's      f
4        64486            Finance  region_29  Bachelor's      m

  recruitment_channel  no_of_trainings  age  previous_year_rating  \
0           sourcing                1   24                   NaN
1              other                1   31                   3.0
2              other                1   31                   1.0
3              other                3   31                   2.0
4           sourcing                1   30                   4.0

   length_of_service  KPIs_met >80%  awards_won?  avg_training_score
0                  1              1            0                  77
1                  5              0            0                  51
2                  4              0            0                  47
3                  9              0            0                  65
4                  7              0            0                  61
```

```
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
```

```
print(trainhr.isnull().sum())
print(testhr.isnull().sum())
```

```
employee_id              0
department               0
region                   0
education                0
gender                   0
recruitment_channel      0
no_of_trainings          0
age                      0
previous_year_rating     0
length_of_service        0
KPIs_met >80%            0
awards_won?              0
avg_training_score       0
is_promoted              0
dtype: int64
no_of_trainings             0
age                         0
length_of_service           0
avg_training_score          0
department_Analytics        0
                           ..
previous_year_rating_5.0    0
KPIs_met >80%_0             0
KPIs_met >80%_1             0
```

```
awards_won?_0                0
awards_won?_1                0
Length: 64, dtype: int64
```

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

```
education
Bachelor's          36669
Master's & above    14925
NaN                  2409
Below Secondary       805
Name: count, dtype: int64
```

```
trainhr.education=trainhr.education.fillna("Bachelor's")
```

```
trainhr.education.value_counts()
```

```
education
Bachelor's          39078
Master's & above    14925
Below Secondary       805
Name: count, dtype: int64
```

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

```
previous_year_rating
3.0    18618
5.0    11741
4.0     9877
1.0     6223
2.0     4225
NaN     4124
Name: count, dtype: int64
```

```
trainhr.previous_year_rating=trainhr.previous_year_rating.fillna(3.0)
```

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

```
previous_year_rating
3.0    22742
5.0    11741
4.0     9877
1.0     6223
2.0     4225
Name: count, dtype: int64
```

test

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

```
education
Bachelor's          15578
Master's & above     6504
NaN                  1034
Below Secondary       374
Name: count, dtype: int64
```

```
testhr.education=testhr.education.fillna("Bachelor's")
```

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

```
education
Bachelor's          16612
Master's & above     6504
Below Secondary       374
Name: count, dtype: int64
```

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

```
previous_year_rating
3.0    7921
5.0    5097
4.0    4249
1.0    2680
NaN    1812
2.0    1731
Name: count, dtype: int64
```

```
testhr.previous_year_rating=testhr.previous_year_rating.fillna(3.0)
```

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

```
previous_year_rating
3.0    9733
5.0    5097
4.0    4249
1.0    2680
2.0    1731
Name: count, dtype: int64
```

```
#describe of no_of_trainings,age,length_of_service,avg_training_score
trainhr.no_of_trainings.describe()
```

```
count    54808.000000
mean         1.253011
std          0.609264
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
```

```
max         10.000000
Name: no_of_trainings, dtype: float64
```

```
trainhr.age.describe()
```

```
count    54808.000000
mean        34.803915
std          7.660169
min         20.000000
25%         29.000000
50%         33.000000
75%         39.000000
max         60.000000
Name: age, dtype: float64
```

```
trainhr.length_of_service.describe()
```

```
count    54808.000000
mean         5.865512
std          4.265094
min          1.000000
25%          3.000000
50%          5.000000
75%          7.000000
max         37.000000
Name: length_of_service, dtype: float64
```

```
trainhr.avg_training_score.describe()
```

```
count    54808.000000
mean        63.386750
std         13.371559
min         39.000000
25%         51.000000
50%         60.000000
75%         76.000000
max         99.000000
Name: avg_training_score, dtype: float64
```

```
print(trainhr.isnull().any())
print(testhr.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
```

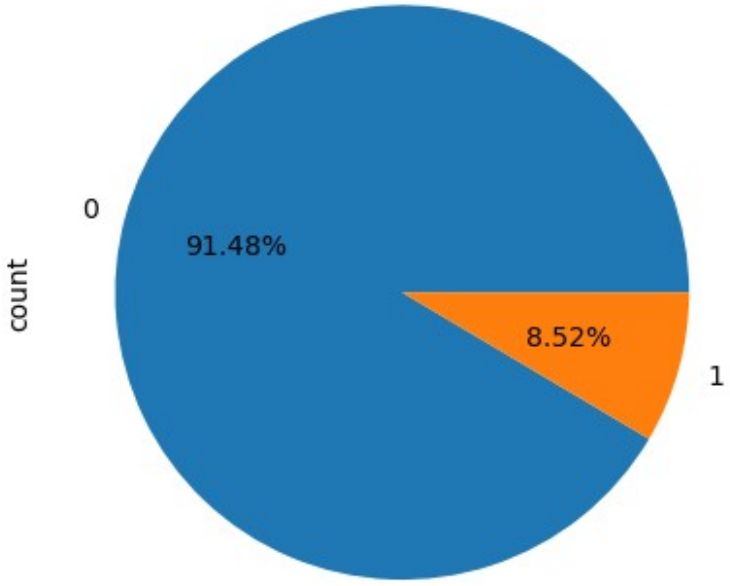
```
length_of_service       False
KPIs_met >80%           False
awards_won?             False
avg_training_score      False
is_promoted             False
dtype: bool
no_of_trainings             False
age                         False
length_of_service           False
avg_training_score          False
department_Analytics        False
                            ...
previous_year_rating_5.0    False
KPIs_met >80%_0             False
KPIs_met >80%_1             False
awards_won?_0               False
awards_won?_1               False
Length: 64, dtype: bool
```

```
#frequency counts of is_promoted,gender,previous_year_ratnig,with pie diagram
trainhr.is_promoted.value_counts()
```

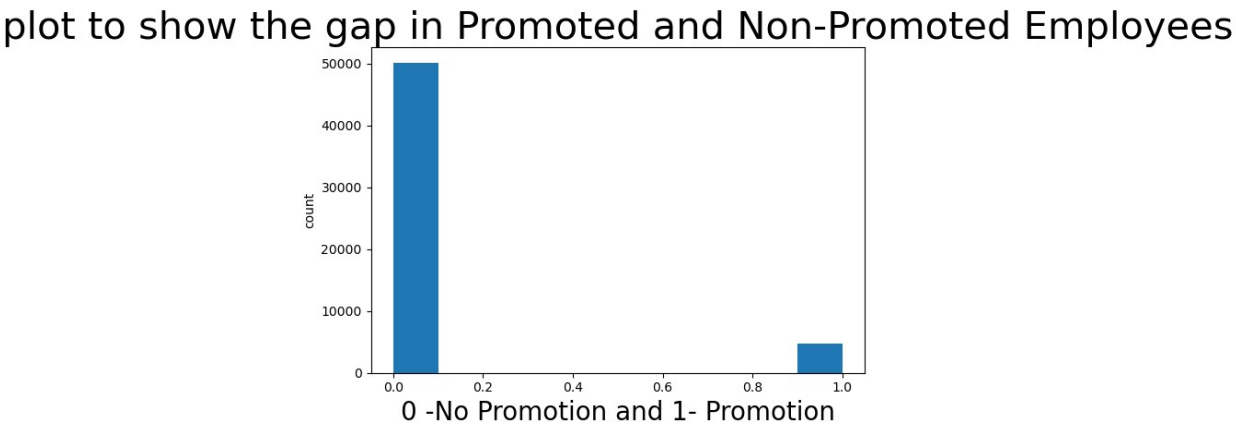
```
is_promoted
0    50140
1     4668
Name: count, dtype: int64
```

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

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




```
#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()
```



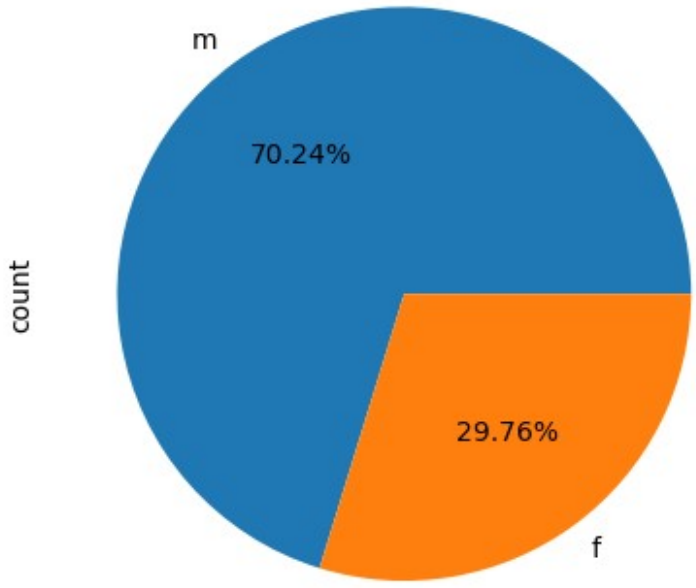

```
trainhr.gender.value_counts()
```

```
gender
m    38496
```

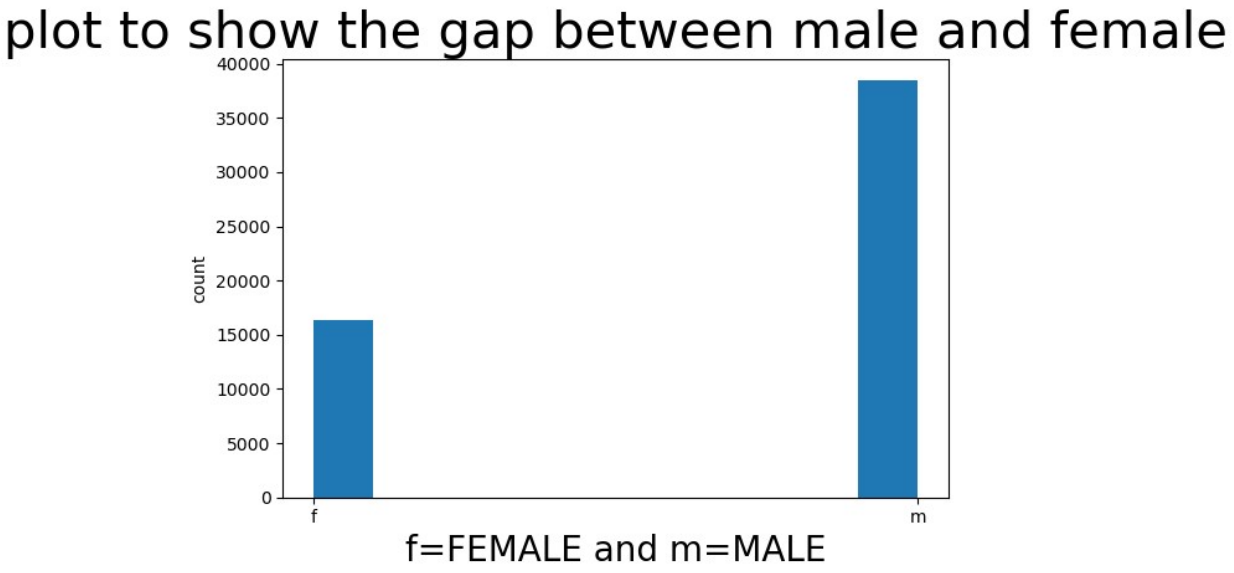
```
f    16312
Name: count, dtype: int64
```

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

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




```
#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()
```

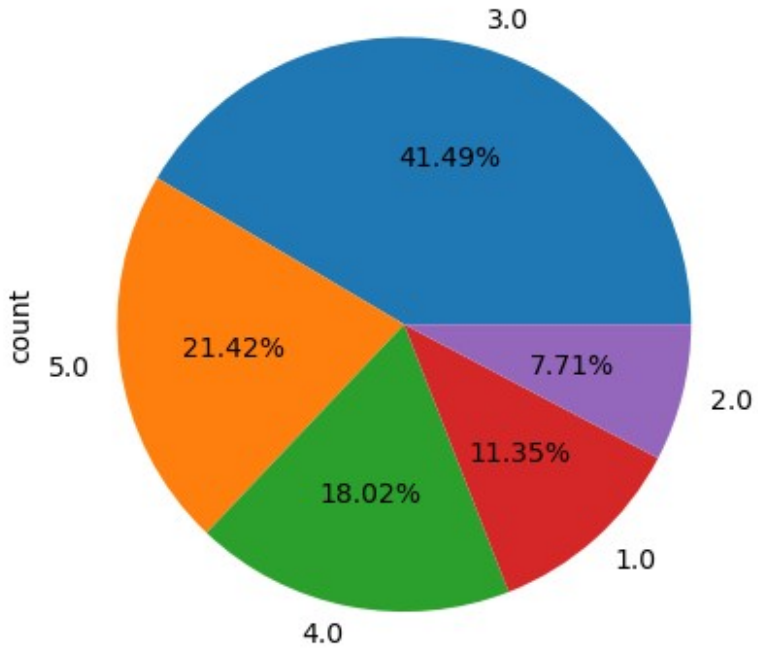



```
trainhr.previous_year_rating.value_counts()
```

```
previous_year_rating
3.0    22742
5.0    11741
4.0     9877
1.0     6223
2.0     4225
Name: count, dtype: int64
```

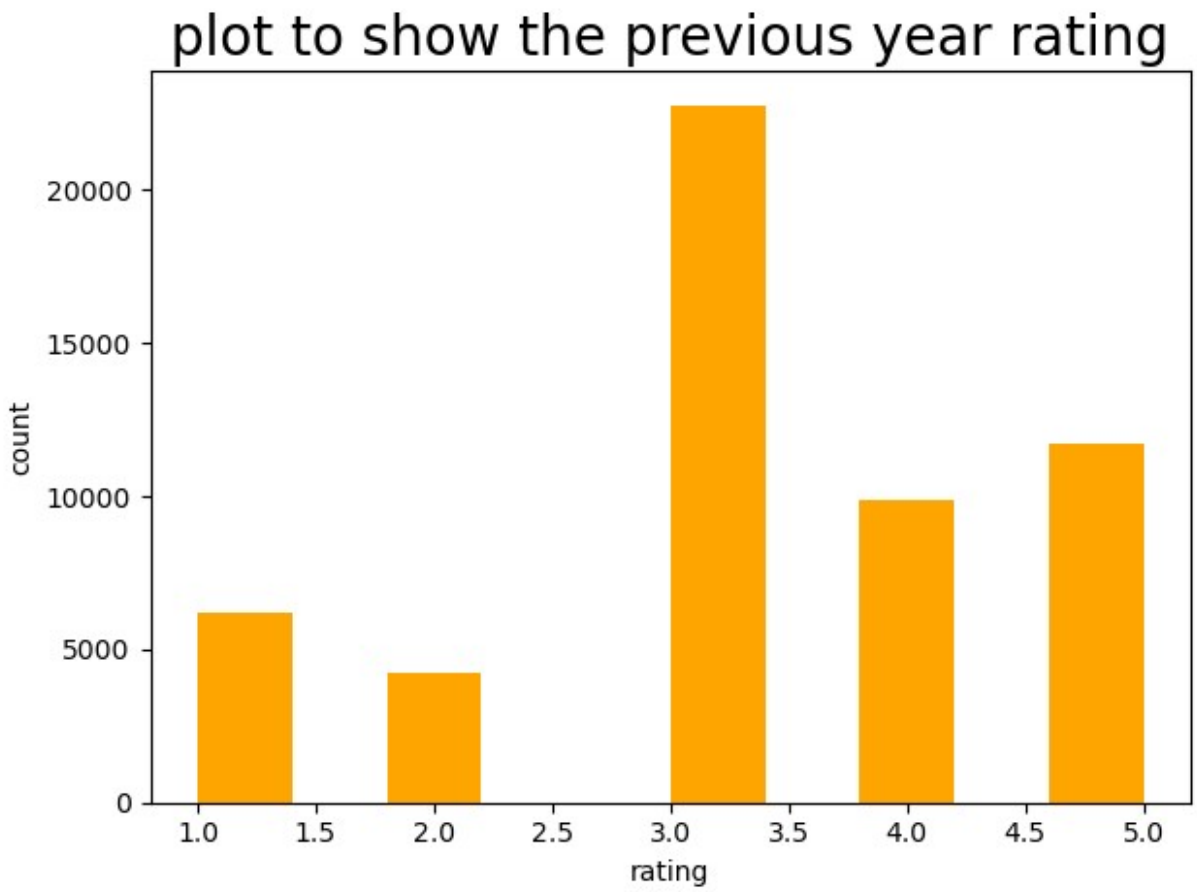
```
trainhr.previous_year_rating.value_counts().plot(kind='pie',autopct='%0.2f%%')
```

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



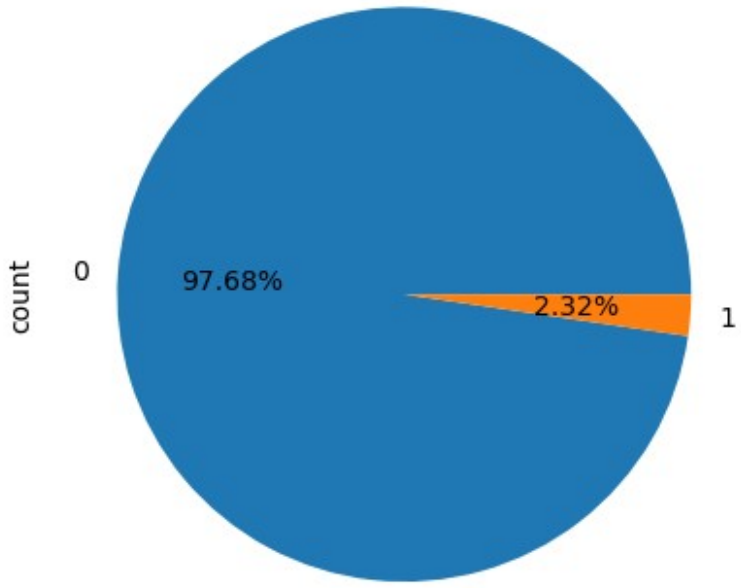

```
#plotting a scatter plot

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()
```

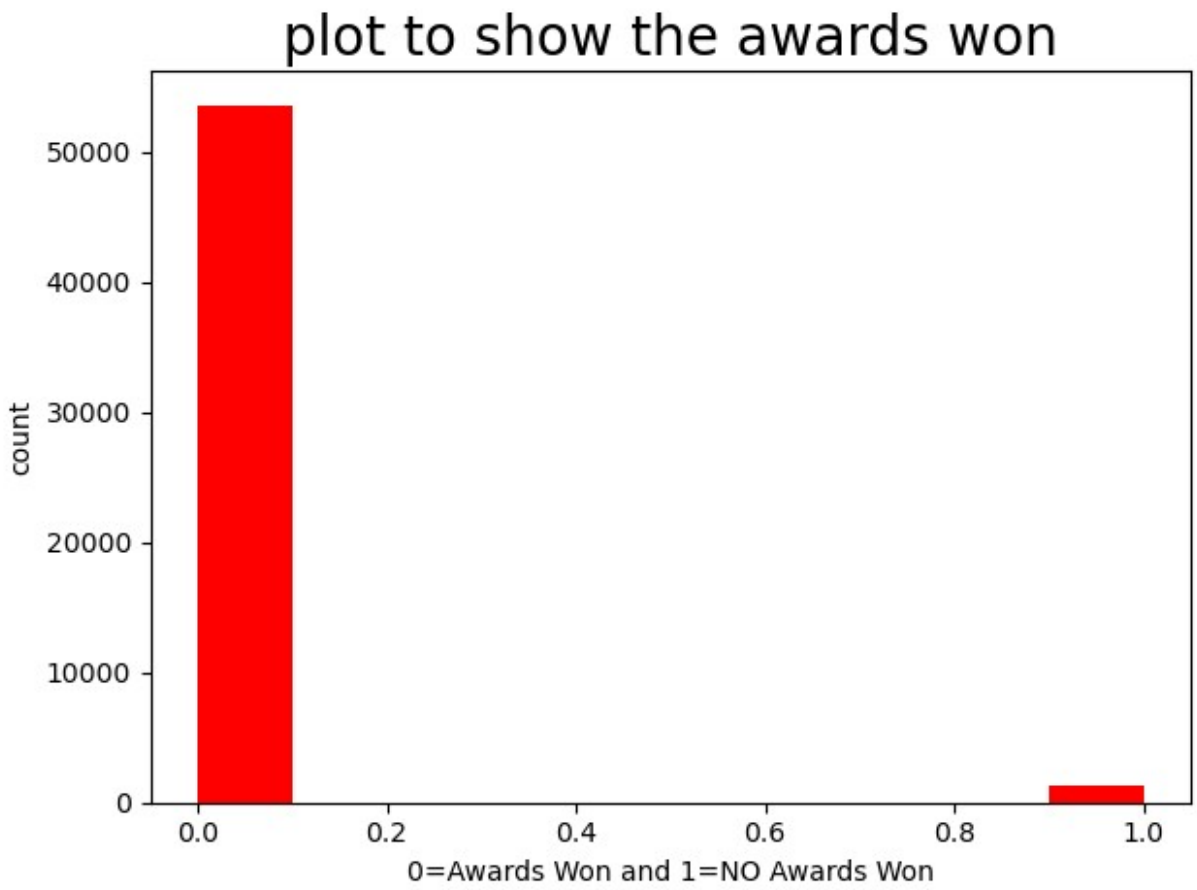



```
trainhr['awards_won?'].value_counts().plot(kind='pie',autopct='%0.2f%
%')
```

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

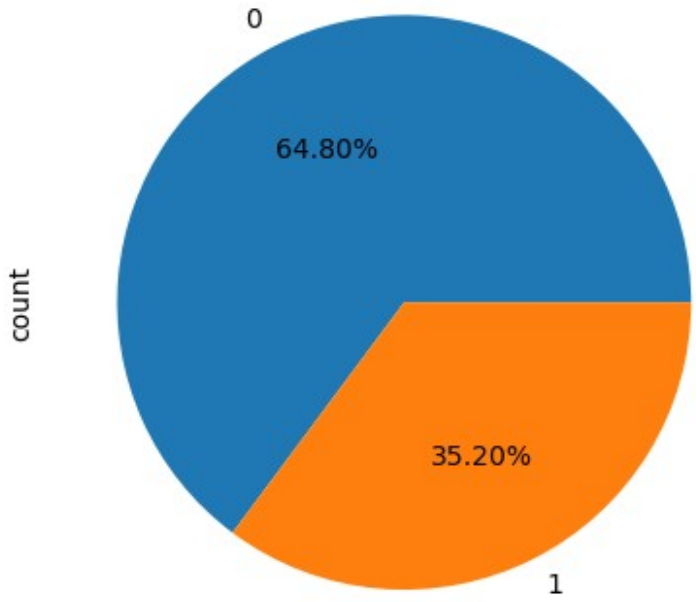



```
#plotting a scatter plot
plt.hist(trainhr['awards_won?'],color='red')
plt.title('plot to show the awards won', fontsize = 20)
plt.xlabel('0=Awards Won and 1=NO Awards Won', fontsize = 10)
plt.ylabel('count')
plt.tight_layout()
plt.show()
```

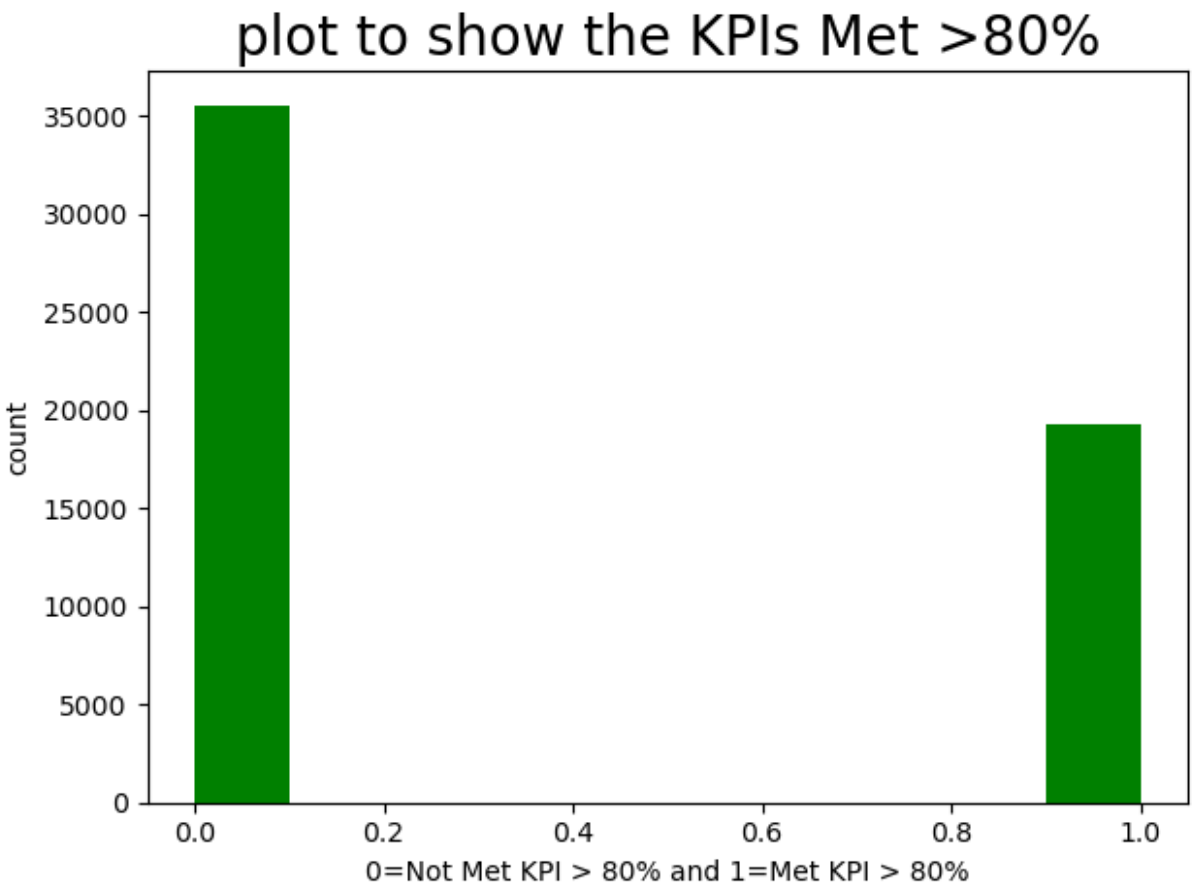



```
trainhr['KPIs_met >80%'].value_counts().plot(kind='pie',autopct='%0.2f
%%')
```

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




```
#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
import seaborn as sns
sns.distplot(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()
```

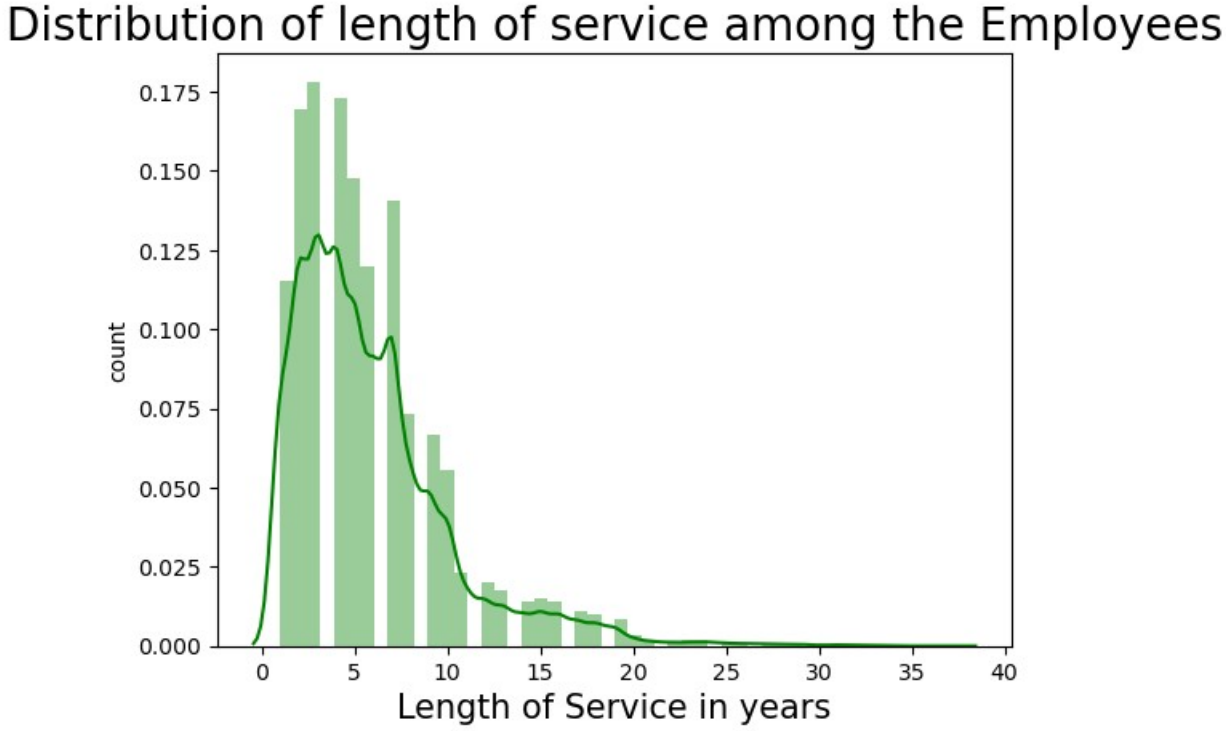
```
C:\Users\Guruji\AppData\Local\Temp\ipykernel_12492\306004379.py:3:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

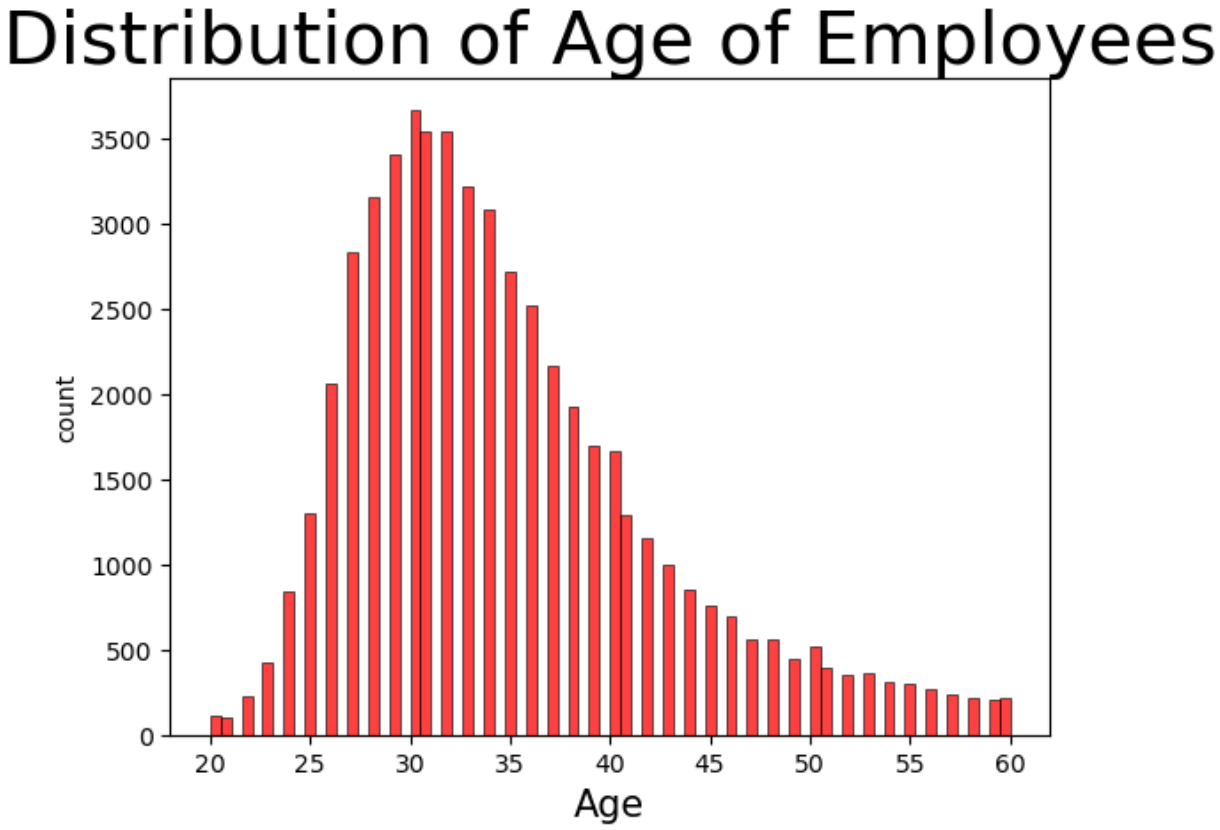
```
  sns.distplot(trainhr['length_of_service'], color = 'green')
c:\Users\Guruji\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\Guruji\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



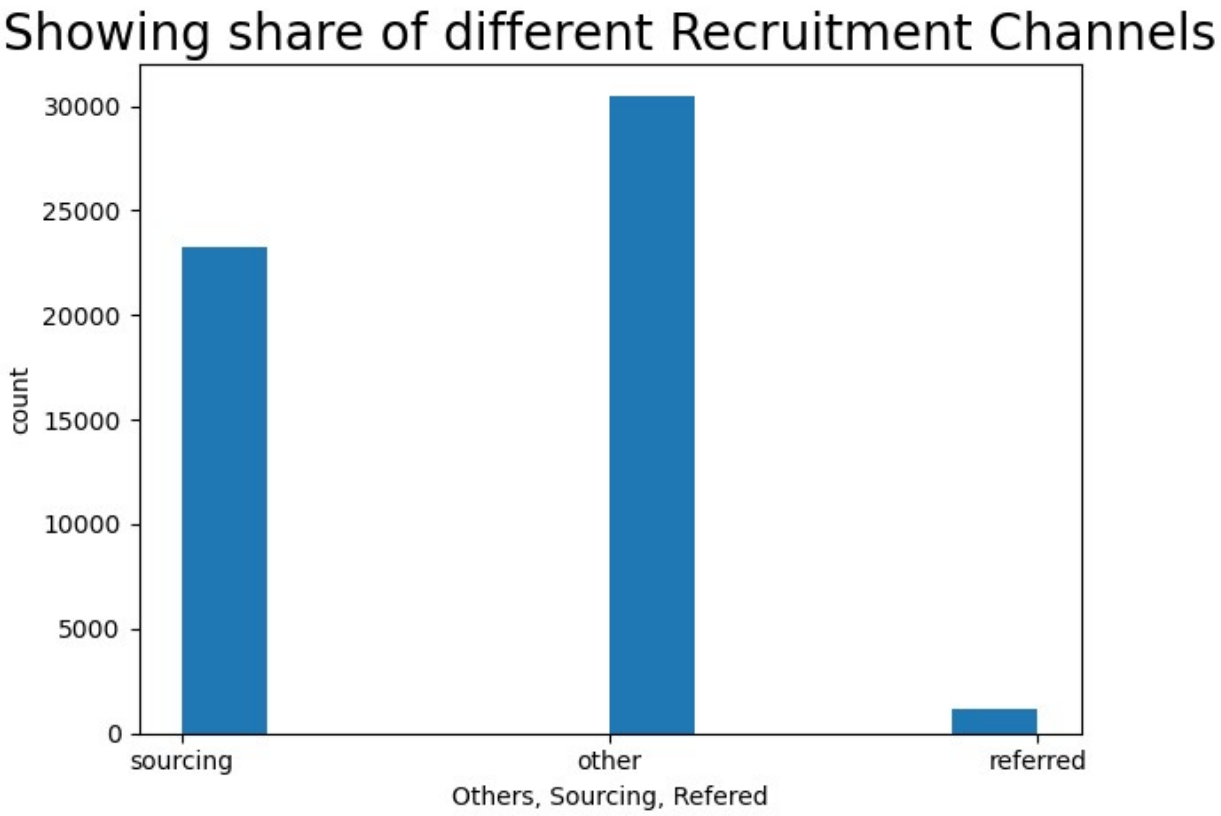

```
# checking the distribution of age of Employees in the company
sns.histplot(trainhr['age'], color = 'red')
plt.title('Distribution of Age of Employees', fontsize = 30)
plt.xlabel('Age', fontsize = 15)
plt.ylabel('count')
plt.show()

c:\Users\Guruji\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\Guruji\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
```

```
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
#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()
```



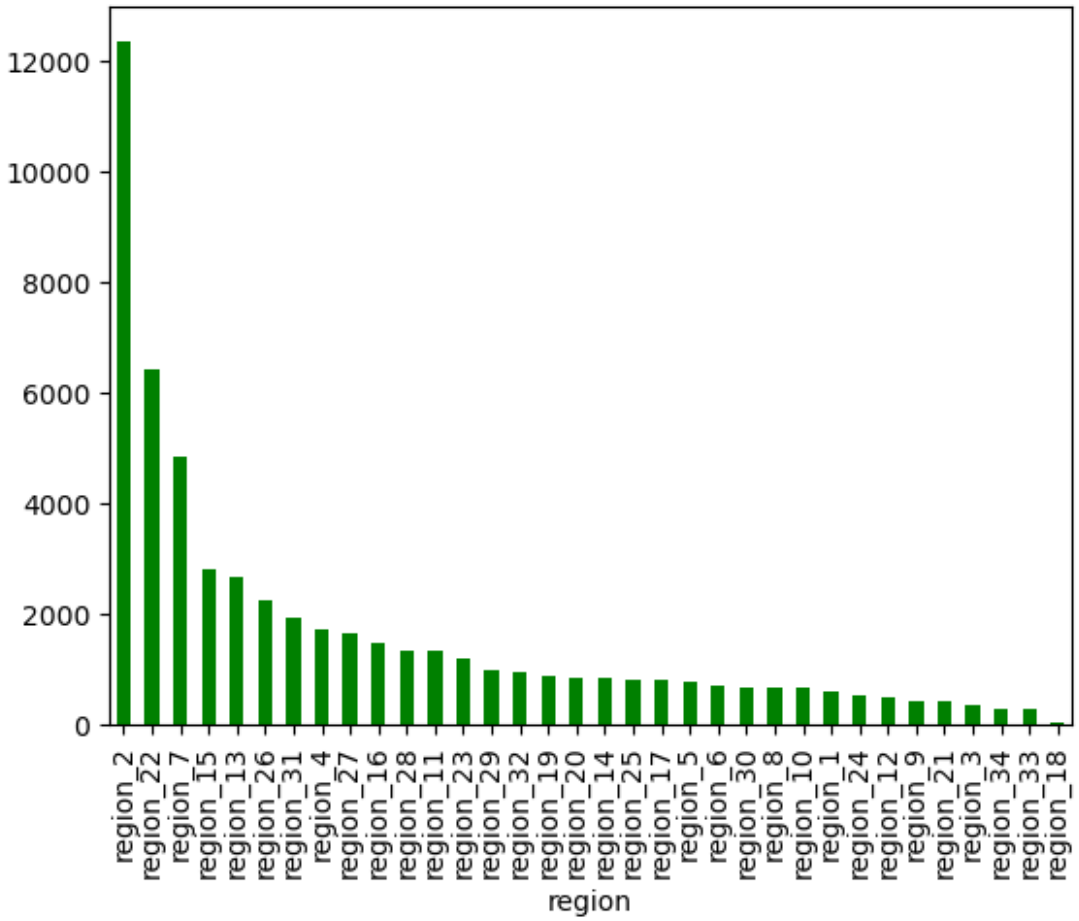

```
# checking the different types of recruitment channels for the company
trainhr['region'].value_counts().sort_values(ascending=False)
```

```
region
region_2     12343
region_22     6428
region_7      4843
region_15     2808
region_13     2648
region_26     2260
region_31     1935
region_4      1703
region_27     1659
region_16     1465
region_28     1318
region_11     1315
region_23     1175
region_29      994
region_32      945
region_19      874
region_20      850
region_14      827
region_25      819
region_17      796
```

```
region_5        766
region_6        690
region_30       657
region_8        655
region_10       648
region_1        610
region_24       508
region_12       500
region_9        420
region_21       411
region_3        346
region_34       292
region_33       269
region_18        31
Name: count, dtype: int64
```

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

```
<Axes: xlabel='region'>
```



#CROSSTAB

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

awards_won?	0	1
is_promoted		
0	49429	711
1	4109	559

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

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

is_promoted	0	1
age		
20	109	4
21	93	5
22	213	18
23	394	34
24	775	70
25	1230	69
26	1897	163
27	2566	261
28	2839	308
29	3111	294
30	3341	324
31	3224	310
32	3237	297
33	2891	319
34	2790	286
35	2428	283
36	2309	208
37	1981	184

```
38            1739  184
39            1534  161
40            1536  127
41            1185  104
42            1058   91
43             915   77
44             777   70
45             712   48
46             654   43
47             530   27
48             513   44
49             408   33
50             490   31
51             355   34
52             323   28
53             350   14
54             289   24
55             276   18
56             243   21
57             233    5
58             197   16
59             192   17
60             203   14
```

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.department,trainhr['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.gender,trainhr['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.avg_training_score.groupby(trainhr.is_promoted).mean()
```

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

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

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

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

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

```
one=trainhr[trainhr.is_promoted==1]
zero=trainhr[trainhr.is_promoted==0]

#Null-
#Alt-

from scipy.stats import ttest_ind
ttest_ind(one.avg_training_score,zero.avg_training_score,equal_var=Fal
se)
#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.avg_training_score.groupby(trainhr.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[trainhr.previous_year_rating==1.0]
two=trainhr[trainhr.previous_year_rating==2.0]
three=trainhr[trainhr.previous_year_rating==3.0]
four=trainhr[trainhr.previous_year_rating==4.0]
five=trainhr[trainhr.previous_year_rating==5.0]
```

```
from scipy.stats import f_oneway
f_oneway(ones.avg_training_score,two.avg_training_score,three.avg_trai
ning_score,four.avg_training_score,five.avg_training_score)
#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.shape)
print(testhr.shape)

(54808, 14)
(23490, 13)

trainhr.columns

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

X=pd.get_dummies(trainhr,columns=['department', 'region', 'education',
                                  'gender','recruitment_channel',
                                  'previous_year_rating','KPIs_met
>80%',
                                  'awards_won?'])

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

y=X.is_promoted
X=X.drop(['employee_id','is_promoted'],axis=1)

testhr=testhr.drop('employee_id',axis=1)

print(X.shape)
print(testhr.shape)

(54808, 64)
(23490, 64)

from sklearn.preprocessing import LabelEncoder

y=LabelEncoder().fit_transform(y)

from sklearn.linear_model import LogisticRegression

logit=LogisticRegression(max_iter=3000)

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

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

```
0.9317617866004962
```

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

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

```
col_0      0     1
row_0
0      49832   308
1       3432  1236
```

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

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

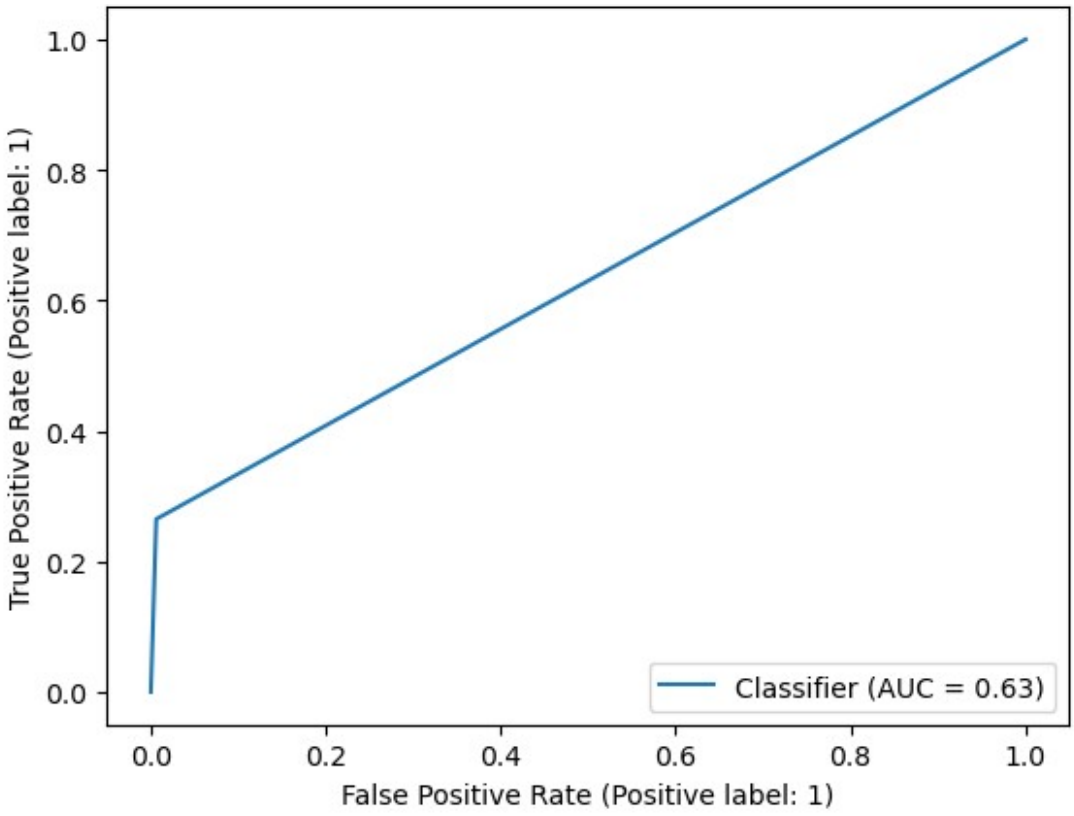
```
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     50140
           1       0.80      0.26      0.40      4668

    accuracy                           0.93     54808
   macro avg       0.87      0.63      0.68     54808
weighted avg       0.92      0.93      0.92     54808
```

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

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1c156687a90>
```




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

```
array([0.93267652, 0.93021346, 0.93048714, 0.93321777, 0.93066326])
```

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

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

Decission Tree

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

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

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

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

```
0.9420887461684425
```

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

```
array([0.93778508, 0.93742018, 0.93851487, 0.93841803, 0.9383268 ])
```

```
np.mean([0.9378763 , 0.93632549, 0.93887977, 0.93850926, 0.93787063])
```

```
0.93789229
```

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

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

non parametric Algorothms -Treebased models-muti tree models

ensemble methods-1) bagging method or boot strap aggregating algorithm in bagging -random forest

Random Forest is a multi tree model or esemble bagging algorithm and random(sampling metod) and Forest(Multiple Trees)

step-1 specify the numberof trees to be built (n_estimators=1000) step-2 create 1000 samples ofdata from original data each will 65% randomly selected observations and sqrt(numof variables) randomly selected per sample. Data is 10000 observaions and 36 variables. Each sample will have 6500 observations and 6 variables. Bootstrap sampling is where observations and variables appear in multiple samples with multiple combinations there by appearing in multiple trees.data with replacemnet method

step-3 upon completion of sampling,for each sample one decision tree isbuilt and also predictions done for each tree. step-4 Aggreagte all the predictions and if it is classification problem- majority voting is done Regresion problem averaging is done

Random Forest algorithm isdesignd to overcome the overfitting problem in dcision tree and alsoto improve the predictive accuracy ofmodel.

there is no fixed formula for deciding howmany trees to be specified Trial &Error orgrid searchAlgorithm.

Grid search algorithms are used in hyper parameter tuning where multiple parameters given and grid searchalgorithms run all possible combination of parameters and identifies best fit model.

grid search parameters like n_estimators=[200,300,500,1000] max_depth=[8,10,12,16]

Random Forest requires computing power and sampling process requires computing power.

parallel building oftrees as each sample is independent.

Random Forest

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

```
RF=RandomForestClassifier(n_estimators=3000)
```

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

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

```
0.999744562837542
```

```
from sklearn.model_selection import cross_val_score
```

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

```
array([0.93532202, 0.93103448, 0.93367999, 0.93376517, 0.9304808 ])
```

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

```
print(classification_report(y,rfpredict_X))
```

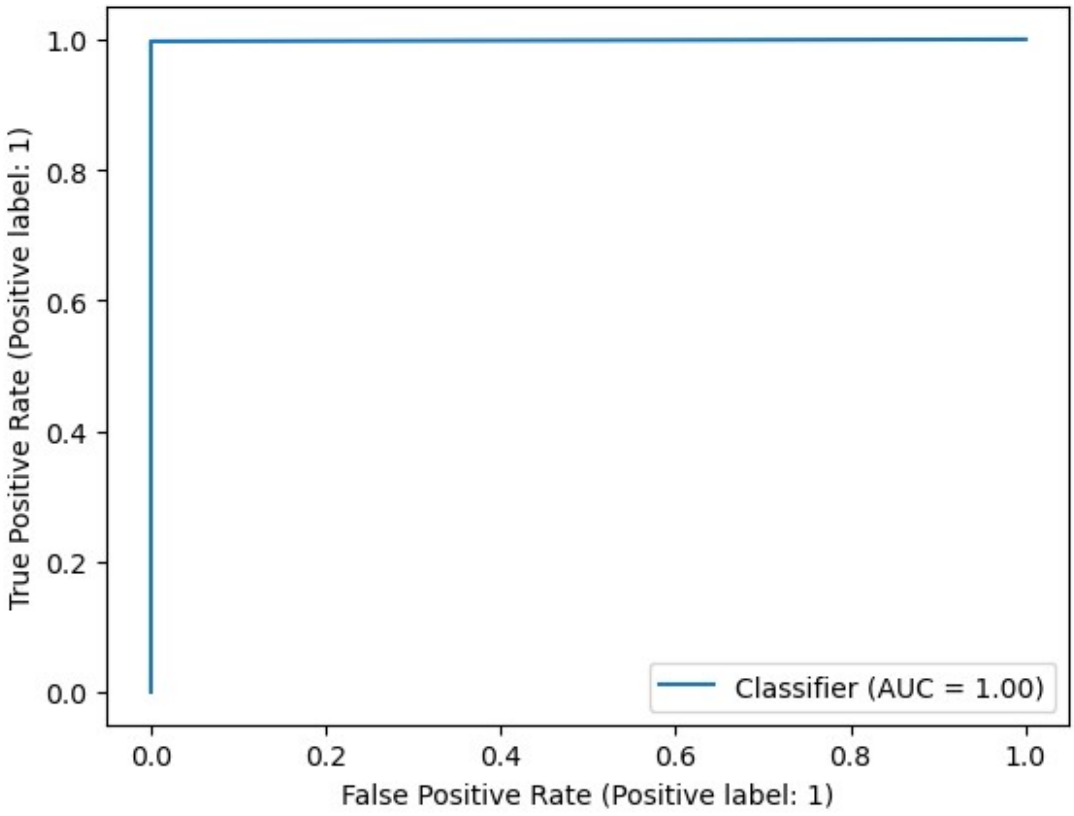
```
RocCurveDisplay.from_predictions(y,rfpredict_X)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     50140
           1       1.00      1.00      1.00      4668

    accuracy                           1.00     54808
   macro avg       1.00      1.00      1.00     54808
weighted avg       1.00      1.00      1.00     54808


<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1c1595321d0>
```




```
pd.DataFrame(rfpredict_X).to_csv('RF.csv')
```

Gradient Boosting Machine

```
from sklearn.ensemble import GradientBoostingClassifier
gbc=GradientBoostingClassifier(max_depth=5,n_estimators=2000)
gbcmodel=gbc.fit(X,y)
gbcmodel.score(X,y)
0.973398044081156
cross_val_score(gbc,X,y)
array([0.93167305, 0.93240285, 0.93358876, 0.93449503, 0.93349147])
np.mean([0.93942711, 0.93814997, 0.94006568, 0.94124624, 0.93951282])
0.939680364
gbcpredict=gbcmodel.predict(testhr)
```

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

Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
mul=MultinomialNB()
mulmodel=mul.fit(X,y)
mulmodel.score(X,y)
0.8943767333236023
cross_val_score(mul,X,y)
array([0.89326765, 0.8934501 , 0.88907134, 0.89672475, 0.90010036])
np.mean([0.89326765, 0.8934501 , 0.88907134, 0.89672475, 0.90010036])
0.8945228399999999
mulpredict=mulmodel.predict(testhr)
pd.DataFrame(mulpredict).to_csv('mul.csv')
from sklearn.naive_bayes import GaussianNB
nbg=GaussianNB()
nbgmodel=nbg.fit(X,y)
nbgmodel.score(X,y)
0.6400890380966282
from sklearn.naive_bayes import BernoulliNB
bb=BernoulliNB()
bbmodel=bb.fit(X,y)
bbmodel.score(X,y)
0.8741789519778135
bbpredict=bbmodel.predict(testhr)
pd.DataFrame(bbpredict).to_csv('bb.csv')
```

SVC

```
from sklearn.svm import SVC
svc=SVC(C=3)
svcmodel=svc.fit(X,y)
svcmodel.score(X,y)
0.9210334257772588
svcpredict=svcmodel.predict(testhr)
pd.DataFrame(svcpredict).to_csv('svc.csv')
```