1. Understand the Problem Statement & Import Packages and Datasets :

```
# Warning Libraries :
import warnings
warnings.filterwarnings("ignore")

# Scientific and Data Manipulation Libraries :
import pandas as pd
import numpy as np
import math
import gc
import os

# ML Libraries :
from sklearn.preprocessing import OneHotEncoder,LabelEncoder,StandardScaler,MinMaxScaler
from sklearn.model_selection import KFold,train_test_split,cross_val_score,StratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier,VotingClass
from sklearn.metrics import f1_score,confusion_matrix,classification_report
# Boosting Algorithms :
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier

# Data Visualization Libraries :
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.io as pio
import plotly.graph_objects as go
import plotly.express as px



train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')


train.head()
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	
1	65141	Operations	region_22	Bachelor's	m	other	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	
		Sales &					

```
display(test.head())
```

```
display(test.head())
```

	employee_id	department	region	education	gender	recruitment_channel	no_of
0	8724	Technology	region_26	Bachelor's	m	sourcing	
1	74430	HR	region_4	Bachelor's	f	other	
2	72255	Sales & Marketing	region_13	Bachelor's	m	other	
3	38562	Procurement	region_2	Bachelor's	f	other	

2. Perform EDA (Exploratory Data Analysis) - Understanding the Datasets :

```
train.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
```

```
train.describe()
```

	employee_id	no_of_trainings	age	previous_year_rating	length_of_s

```
# unique values

def unique_values(data):
    for column in data.columns :

        print("No. of Unique Values in "+column+" Column are : "+str(data[column].nunique(
        print("Actual Unique Values in "+column+" Column are : "+str(data[column].sort_val
        print("")

unique_values(train)
```

```
No. of Unique Values in employee_id Column are : 54808
Actual Unique Values in employee_id Column are : [    1     2     4 ... 78296 78297

No. of Unique Values in department Column are : 9
Actual Unique Values in department Column are : ['Analytics' 'Finance' 'HR' 'Legal'
 'Sales & Marketing' 'Technology']

No. of Unique Values in region Column are : 34
Actual Unique Values in region Column are : ['region_1' 'region_10' 'region_11' 'reg
 'region_15' 'region_16' 'region_17' 'region_18' 'region_19' 'region_2'
 'region_20' 'region_21' 'region_22' 'region_23' 'region_24' 'region_25'
 'region_26' 'region_27' 'region_28' 'region_29' 'region_3' 'region_30'
 'region_31' 'region_32' 'region_33' 'region_34' 'region_4' 'region_5'
 'region_6' 'region_7' 'region_8' 'region_9']

No. of Unique Values in education Column are : 3
Actual Unique Values in education Column are : ["Bachelor's" 'Below Secondary' "Mast

No. of Unique Values in gender Column are : 2
Actual Unique Values in gender Column are : ['f' 'm']

No. of Unique Values in recruitment_channel Column are : 3
Actual Unique Values in recruitment_channel Column are : ['other' 'referred' 'sourci

No. of Unique Values in no_of_trainings Column are : 10
Actual Unique Values in no_of_trainings Column are : [ 1  2  3  4  5  6  7  8  9 10]

No. of Unique Values in age Column are : 41
Actual Unique Values in age Column are : [20 21 22 23 24 25 26 27 28 29 30 31 32 33
 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60]

No. of Unique Values in previous_year_rating Column are : 5
Actual Unique Values in previous_year_rating Column are : [ 1.  2.  3.  4.  5. nan]

No. of Unique Values in length_of_service Column are : 35
Actual Unique Values in length_of_service Column are : [ 1  2  3  4  5  6  7  8  9 1
 25 26 27 28 29 30 31 32 33 34 37]

No. of Unique Values in KPIs_met >80% Column are : 2
Actual Unique Values in KPIs_met >80% Column are : [0 1]

No. of Unique Values in awards_won? Column are : 2
Actual Unique Values in awards_won? Column are : [0 1]

No. of Unique Values in avg_training_score Column are : 61
```

```
Actual Unique Values in avg_training_score Column are : [39 40 41 42 43 44 45 46 47
 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
 87 88 89 90 91 92 93 94 95 96 97 98 99]

No. of Unique Values in is_promoted Column are : 2
Actual Unique Values in is_promoted Column are : [0 1]
```

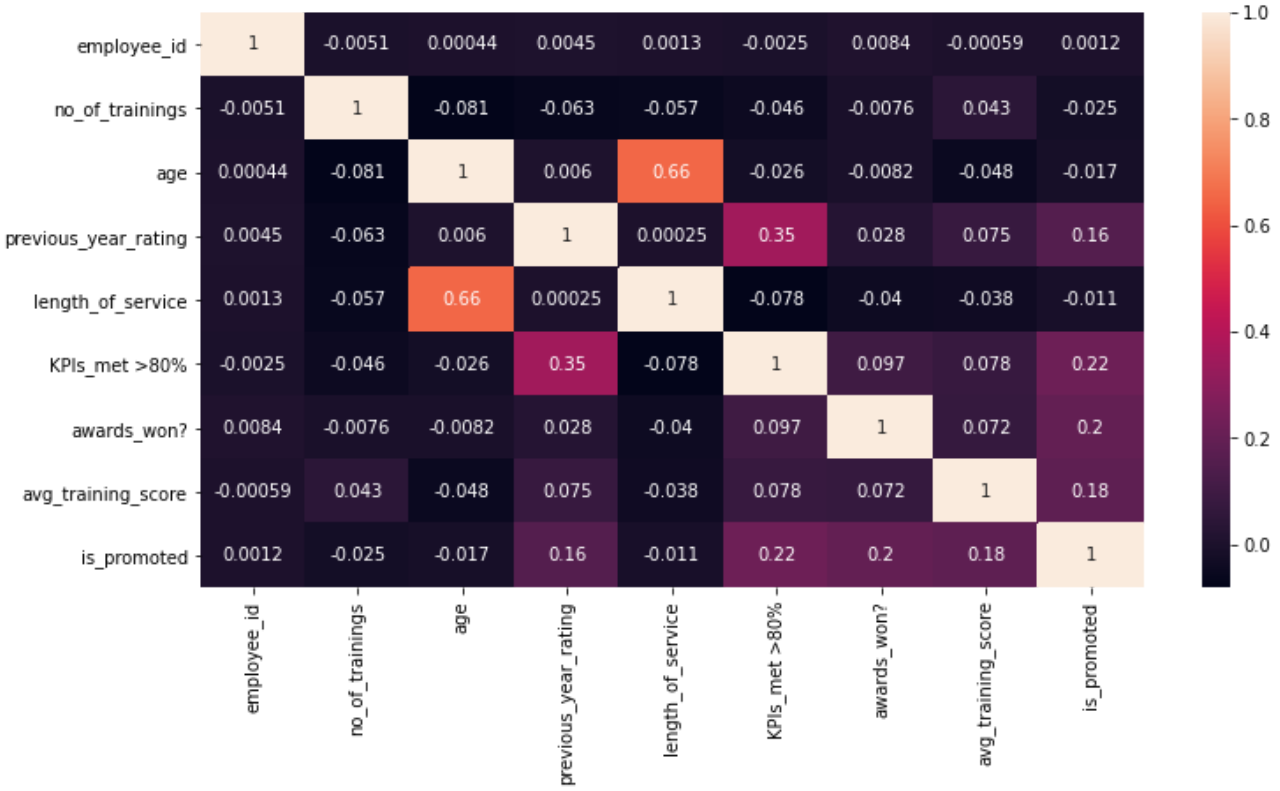
```
unique_values(test)
```

```
No. of Unique Values in employee_id Column are : 23490
Actual Unique Values in employee_id Column are : [    3     6    11 ... 78284 78293

No. of Unique Values in department Column are : 9
Actual Unique Values in department Column are : ['Analytics' 'Finance' 'HR' 'Legal'
 'Sales & Marketing' 'Technology']

No. of Unique Values in region Column are : 34
Actual Unique Values in region Column are : ['region_1' 'region_10' 'region_11' 'reg
 'region_15' 'region_16' 'region_17' 'region_18' 'region_19' 'region_2'
 'region_20' 'region_21' 'region_22' 'region_23' 'region_24' 'region_25'
 'region_26' 'region_27' 'region_28' 'region_29' 'region_3' 'region_30'
 'region_31' 'region_32' 'region_33' 'region_34' 'region_4' 'region_5'
 'region_6' 'region_7' 'region_8' 'region_9']

No. of Unique Values in education Column are : 3
Actual Unique Values in education Column are : ["Bachelor's" 'Below Secondary' "Mast

No. of Unique Values in gender Column are : 2
Actual Unique Values in gender Column are : ['f' 'm']

No. of Unique Values in recruitment_channel Column are : 3
Actual Unique Values in recruitment_channel Column are : ['other' 'referred' 'sourci

No. of Unique Values in no_of_trainings Column are : 9
Actual Unique Values in no_of_trainings Column are : [1 2 3 4 5 6 7 8 9]

No. of Unique Values in age Column are : 41
Actual Unique Values in age Column are : [20 21 22 23 24 25 26 27 28 29 30 31 32 33
 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60]

No. of Unique Values in previous_year_rating Column are : 5
Actual Unique Values in previous_year_rating Column are : [ 1.  2.  3.  4.  5. nan]

No. of Unique Values in length_of_service Column are : 34
Actual Unique Values in length_of_service Column are : [ 1  2  3  4  5  6  7  8  9 1
 25 26 27 28 29 30 31 32 33 34]

No. of Unique Values in KPIs_met >80% Column are : 2
Actual Unique Values in KPIs_met >80% Column are : [0 1]

No. of Unique Values in awards_won? Column are : 2
Actual Unique Values in awards_won? Column are : [0 1]

No. of Unique Values in avg_training_score Column are : 61
Actual Unique Values in avg_training_score Column are : [39 40 41 42 43 44 45 46 47
 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
 87 88 89 90 91 92 93 94 95 96 97 98 99]
```

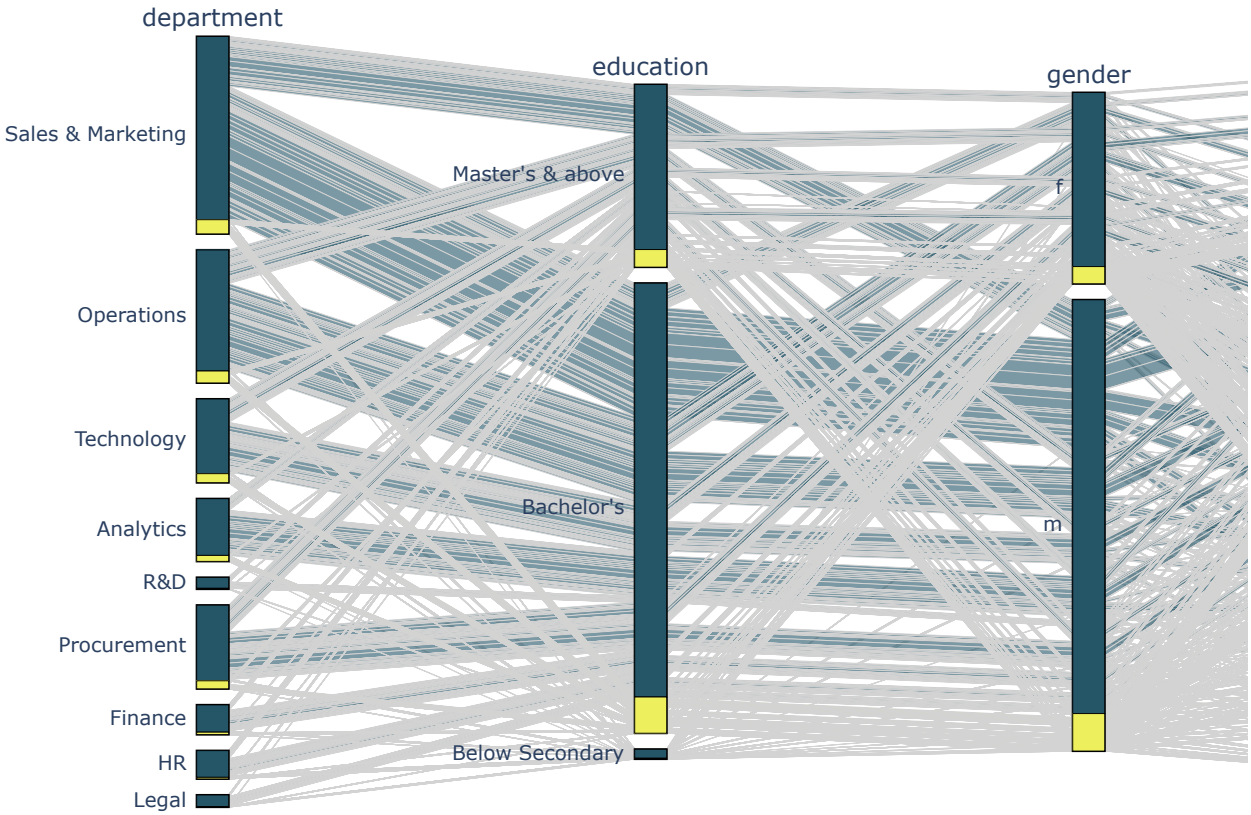
```
train.corr()['is_promoted']
```

```
employee_id             0.001206
no_of_trainings        -0.024896
age                    -0.017166
previous_year_rating    0.159320
length_of_service      -0.010670
KPIs_met >80%           0.221582
awards_won?             0.195871
avg_training_score      0.181147
is_promoted             1.000000
Name: is_promoted, dtype: float64
```

```
plt.figure(figsize=(12,6))
sns.heatmap(train.corr(),annot=True,fmt='.2g')
plt.show()
```




```
# Display How Each Feature is related to Target Variable in a Flow :
fig=px.parallel_categories(train[['department','education','gender','previous_year_rating'
                                  'recruitment_channel',
                                 'is_promoted']],
                           color='is_promoted',
                           color_continuous_scale=px.colors.sequential.Aggrnyl)
fig.show()
```




```
<Figure size 432x288 with 0 Axes>
```

3. Check for Duplicate Rows from Train Data if present : De-Duping

```
print(train.duplicated().sum())
```

```
0
```

```
print(test.duplicated().sum())
```

```
0
```

4. Fill/Impute Missing Values :

```
train.isna().mean()*100
```

```
employee_id             0.000000
department              0.000000
region                  0.000000
education               4.395344
```

```
gender                  0.000000
recruitment_channel     0.000000
no_of_trainings         0.000000
age                     0.000000
previous_year_rating    7.524449
length_of_service       0.000000
KPIs_met >80%           0.000000
awards_won?             0.000000
avg_training_score      0.000000
is_promoted             0.000000
dtype: float64
```

education and previous_year_rating contain null values

```
test.isna().mean()*100
```

```
employee_id             0.000000
department              0.000000
region                  0.000000
education               4.401873
gender                  0.000000
recruitment_channel     0.000000
no_of_trainings         0.000000
age                     0.000000
previous_year_rating    7.713921
length_of_service       0.000000
KPIs_met >80%           0.000000
awards_won?             0.000000
avg_training_score      0.000000
dtype: float64
```

```
train.previous_year_rating.value_counts()
```

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

```
temp=train[train.previous_year_rating.isna()]
temp['length_of_service'].value_counts()
```

```
1    4124
Name: length_of_service, dtype: int64
```

```
# Why is Data Missing in Column "previous_year_rating" ?
# 1. Data was not entered Because those employees were Freshers (i.e) length_of_service =
# 2. No Data would have been there in the Data Source itself for these employees.
# Logically we are imputing with "0" as Freshers with 1 Year Experince may not have previc

# Filling Missing Values in Train and Test :
train["previous_year_rating"] = train["previous_year_rating"].fillna(0)
test["previous_year_rating"] = test["previous_year_rating"].fillna(0)
```

```
train['is_Fresher']=train['previous_year_rating'].apply(lambda x: 1 if x==0 else 0)
test['is_Fresher']=test['previous_year_rating'].apply(lambda x: 1 if x==0 else 0)


display( train['is_Fresher'].value_counts())
```

```
0    50684
1     4124
Name: is_Fresher, dtype: int64
```

```
# Display Missing Values in Train and Test data :
display("Train : ", train.isnull().sum())
display("Test : ",test.isnull().sum())
```

```
'Train : '
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
is_Fresher              0
dtype: int64
'Test : '
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_Fresher              0
dtype: int64
```

```
train[train.education.isna()]
```

	employee_id	department	region	education	gender	recruitment_channel	nc
10	29934	Technology	region_23	NaN	m	sourcing	
21	33332	Operations	region_15	NaN	m	sourcing	
32	35465	Sales & Marketing	region_7	NaN	f	sourcing	
43	17423	Sales & Marketing	region_2	NaN	m	other	
82	66013	Sales & Marketing	region_2	NaN	m	sourcing	
...	...	...	...	...	...	...	
54692	14821	Sales & Marketing	region_2	NaN	f	sourcing	
54717	7684	Analytics	region_2	NaN	m	sourcing	
54729	1797	HR	region_2	NaN	f	other	
54742	38935	Sales & Marketing	region_31	NaN	m	other	

```
# Filling with Mode and New Category called "Others" are Most commonly Used Techniques whi
# So we can assume that while Collecting Data Relevant Members Data were collected Close t

train["education"] = train["education"].ffill(axis = 0)
train["education"] = train["education"].bfill(axis = 0)

test["education"] = test["education"].ffill(axis = 0)
test["education"] = test["education"].bfill(axis = 0)


# Display Missing Values in Train and Test data :
display("Train : ", train.isnull().sum())
display("Test : ",test.isnull().sum())
```

```
'Train : '
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
```

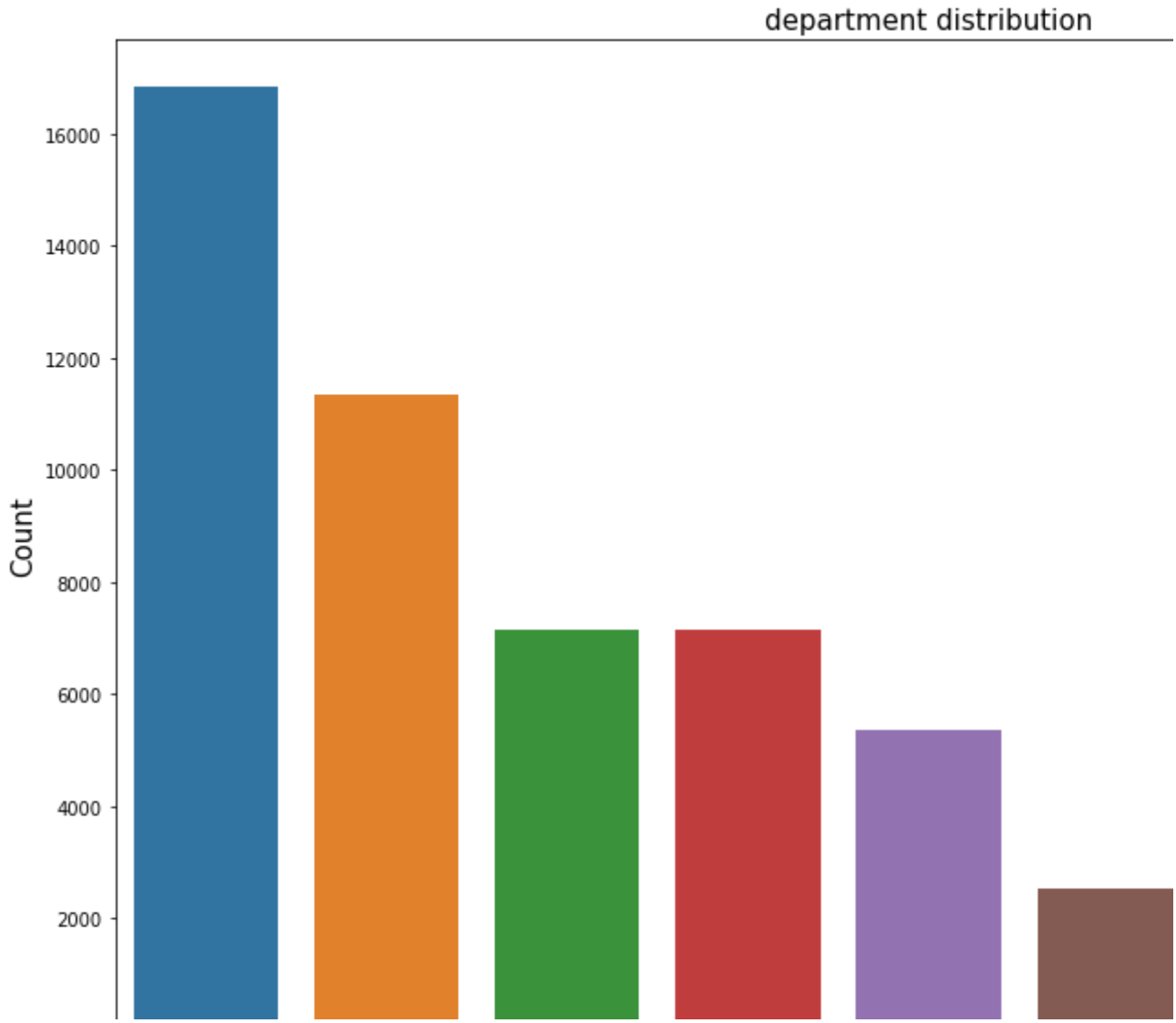
```
train.head()
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	
1	65141	Operations	region_22	Bachelor's	m	other	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	
4	48945	Technology	region_26	Bachelor's	m	other	

```
dtype: int64
```

```
def count_plot(df,col,rotation=90):
  plt.figure(figsize=(16,10))
  sns.countplot(train[str(col)],order=(train[str(col)].value_counts().index))
  plt.ylabel('Count',fontsize=15)
  plt.xlabel('{}'.format(str(col)), fontsize =15)
  plt.title('{} distribution'.format(str(col)),fontsize=15)
  plt.xticks(rotation=rotation,fontsize=20)
  plt.show()
```

```
count_plot(train,'department')
```

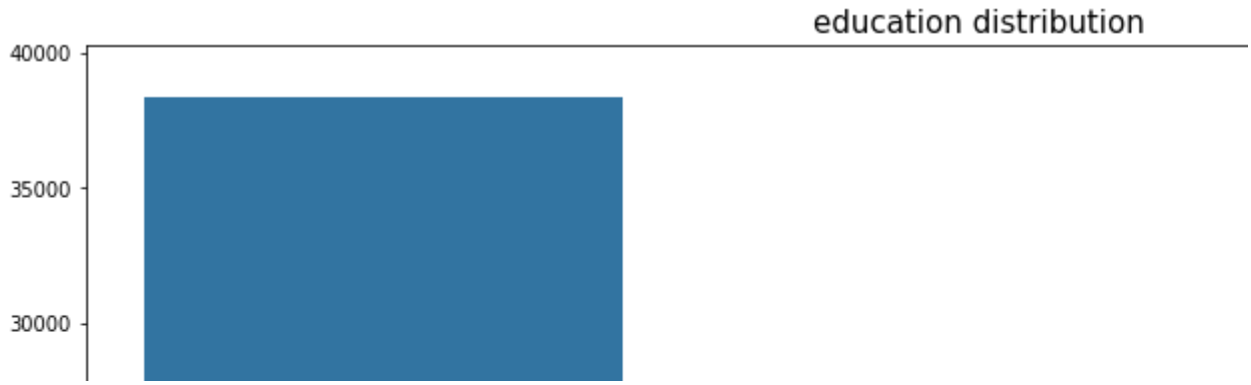


most people work in Sales & Marketing department

```
count_plot(train,'region')
```

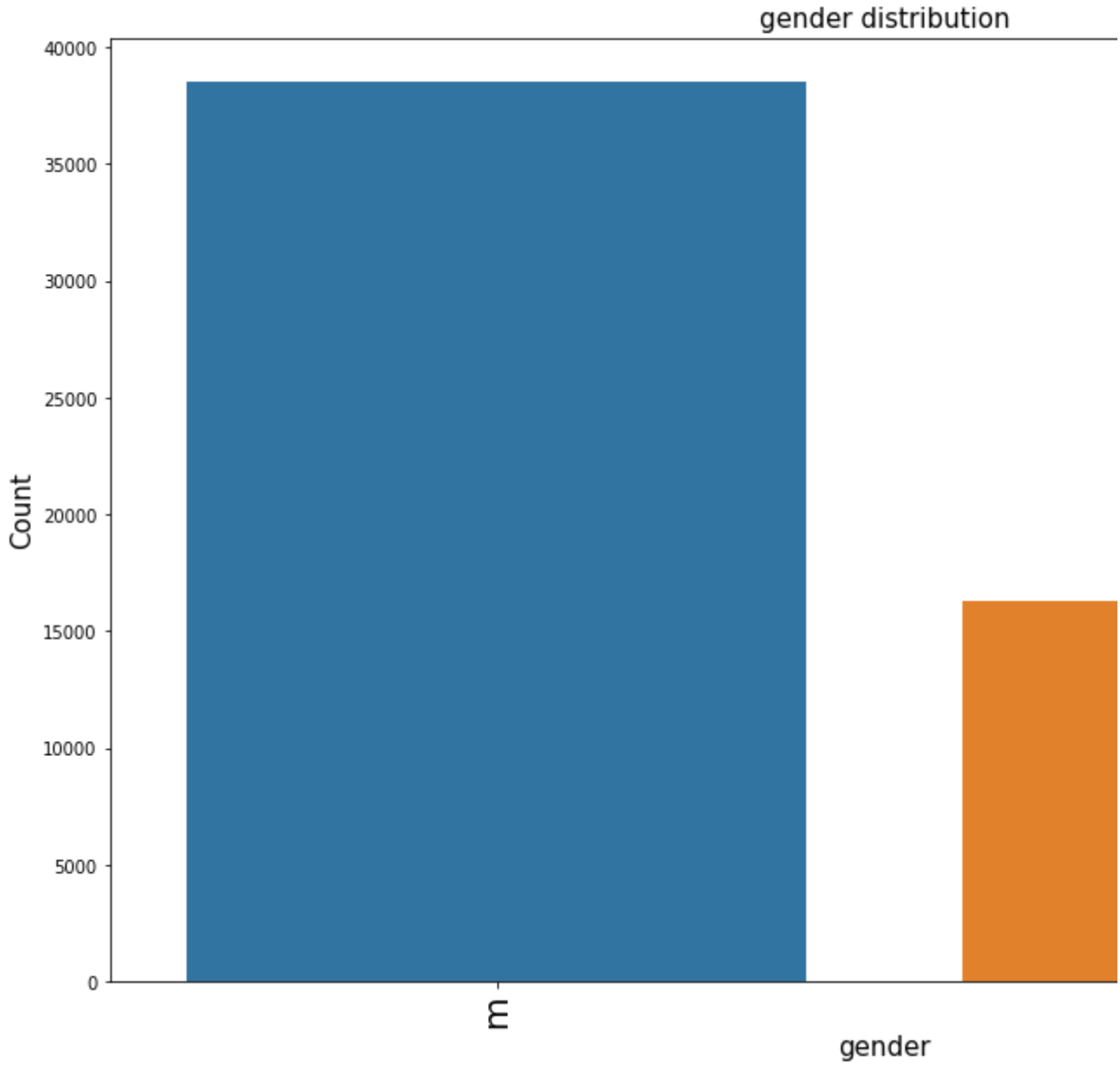
most people work from region 2

```
count_plot(train,'education')
```



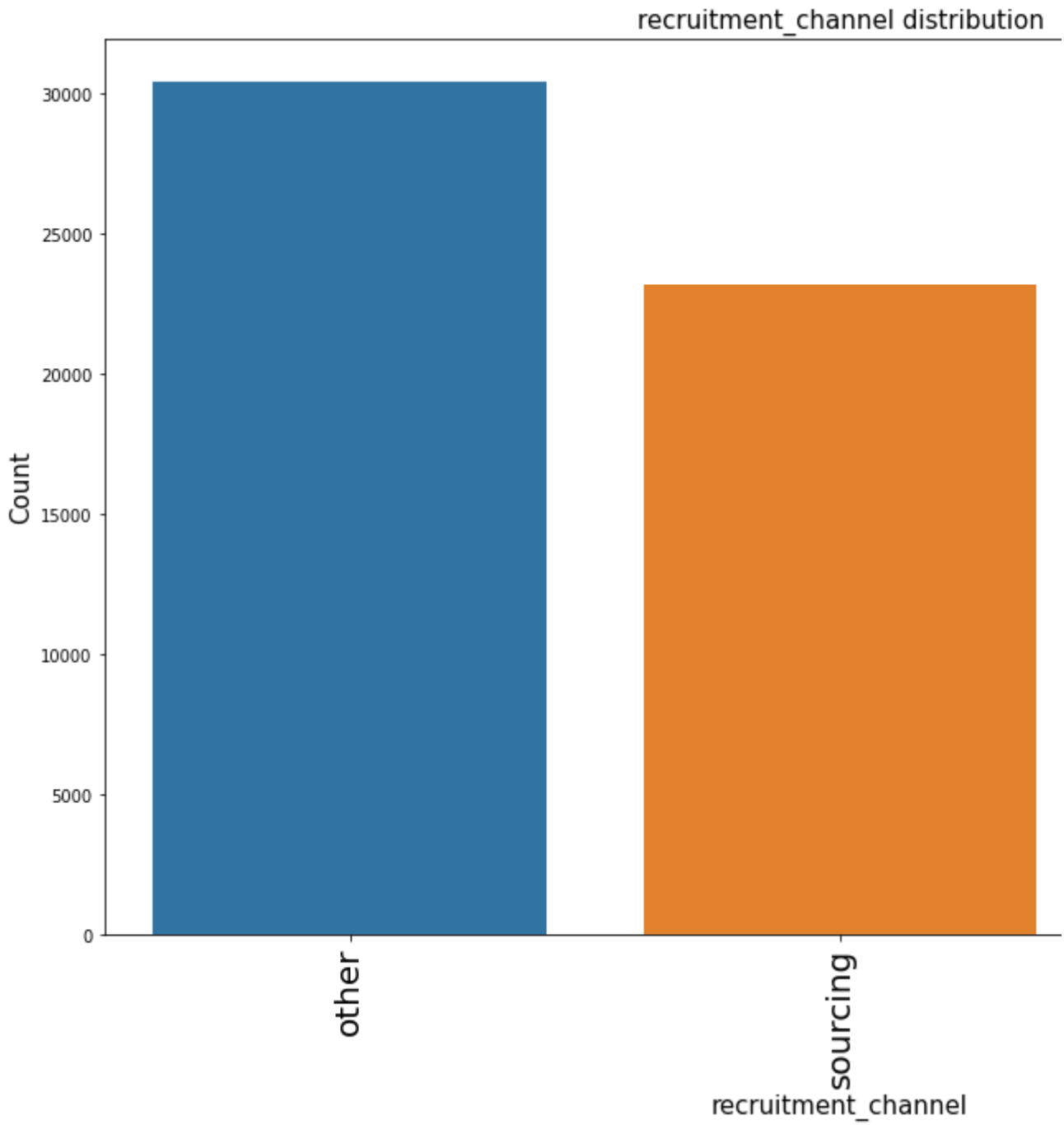
Most people possess bachelors' degrees

```
count_plot(train,'gender')
```



male employees are more than twice the female employees

```
count_plot(train,'recruitment_channel')
```

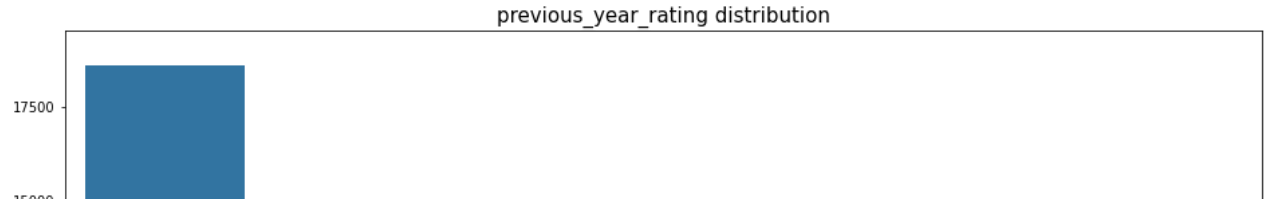


most people are hired through other sources i.e, there are so many job portals nowadays

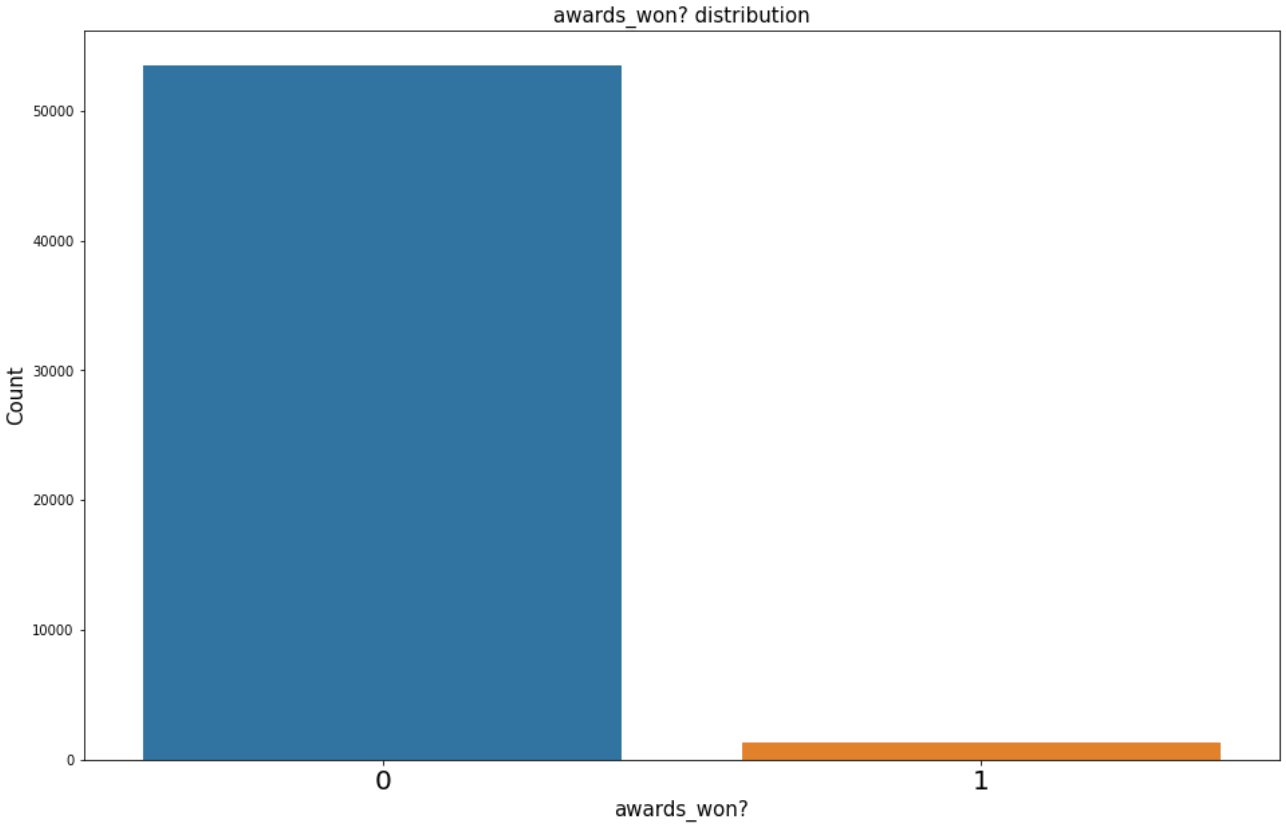
```
count_plot(train,'no_of_trainings',rotation=0)
```

most people have done 1 training only

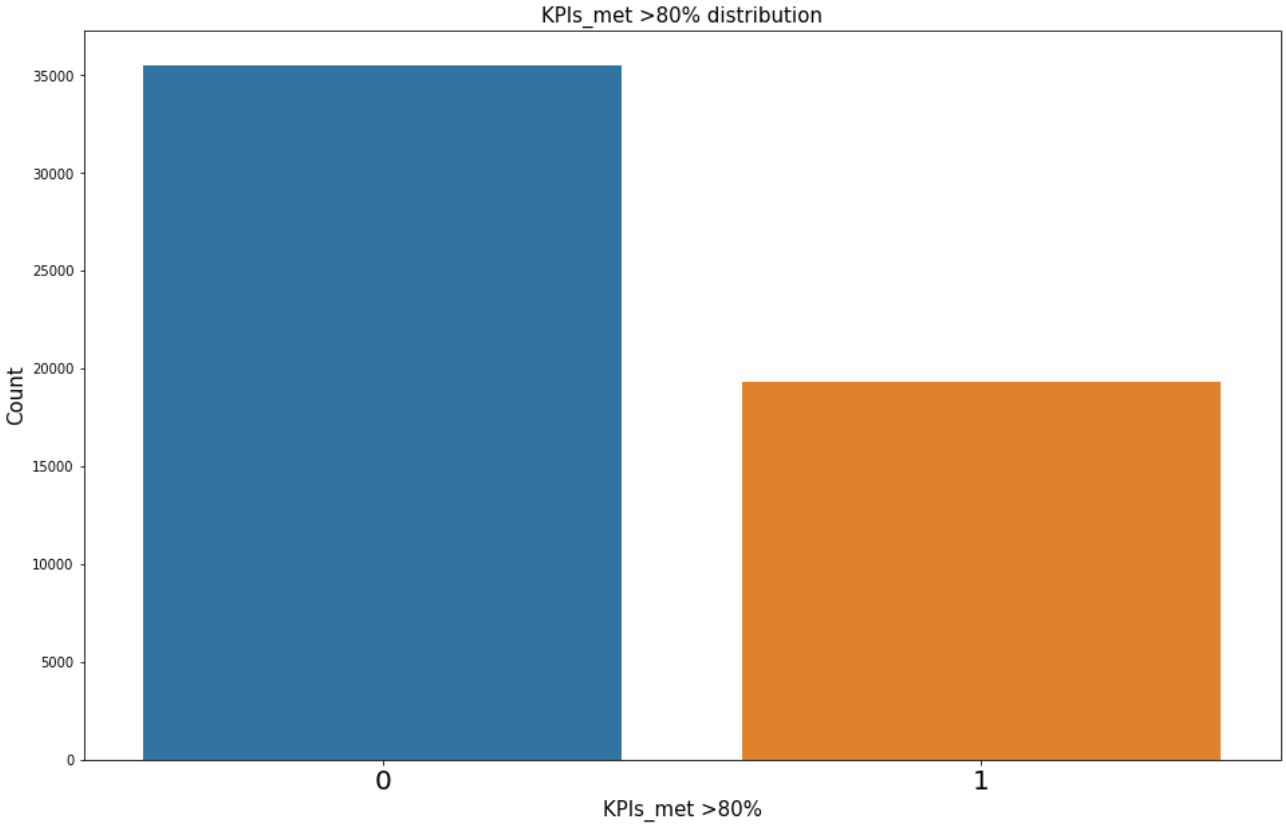

```
count_plot(train,'previous_year_rating',rotation=0)
```



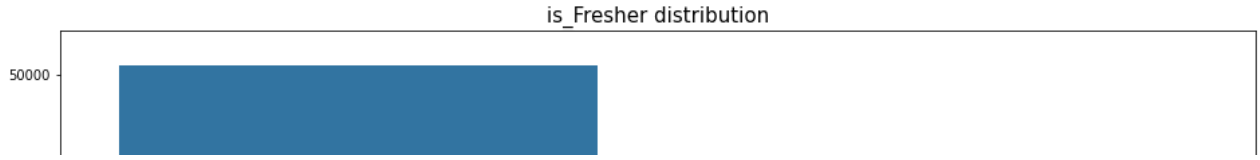

```
count_plot(train,'awards_won?',rotation=0)
```



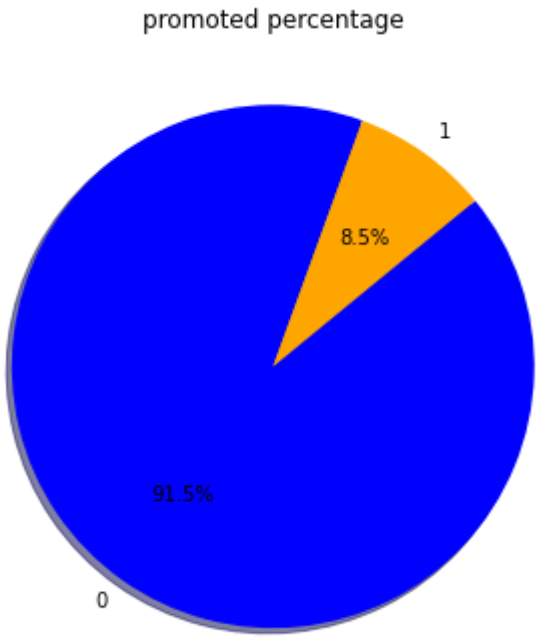

```
count_plot(train,'KPIs_met >80%',rotation=0)
```




```
count_plot(train,'is_Fresher',rotation=0)
```

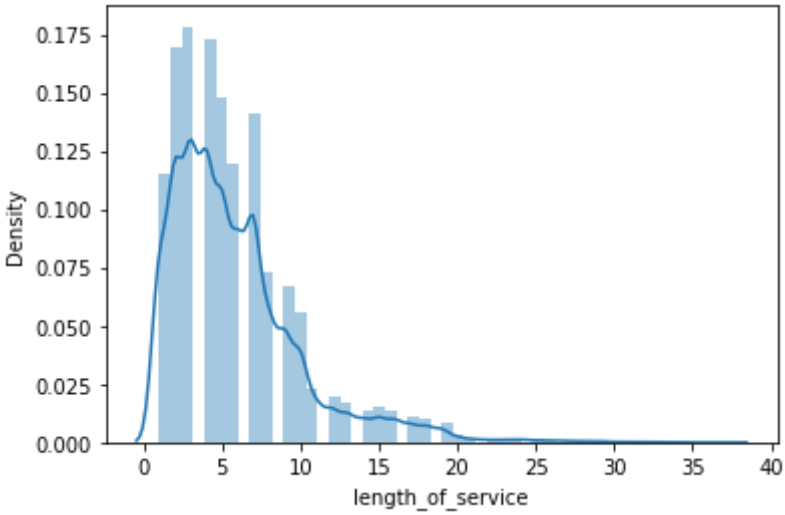



```
vc=train.is_promoted.value_counts()
vc.index
plt.figure(figsize=(10,6))
x=np.array(vc.index)
colors=['blue','orange']
y=(np.array(vc/vc.sum()))*100
plt.pie(y, labels=x, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=70)
plt.title("promoted percentage")
x=np.array(vc.index)
plt.show()
```

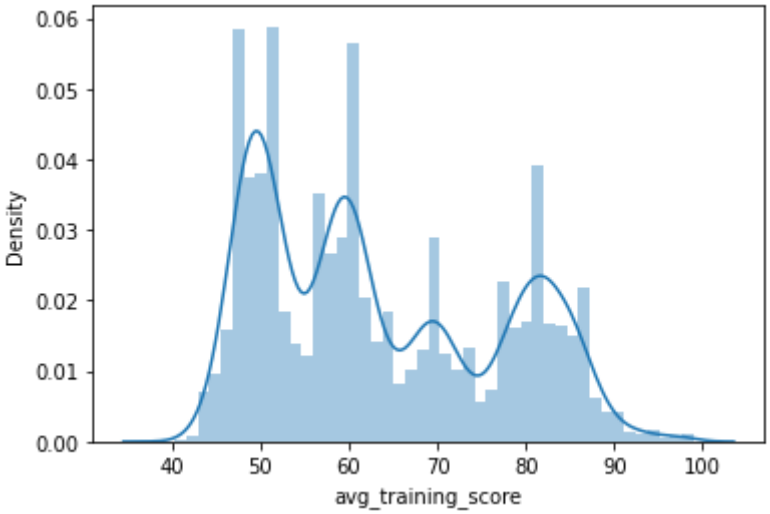



```
# displot -> plot a univariate(Single Feature) distribution of observations.

sns.distplot(train['age'])

train['age'] = pd.cut( x=train['age'], bins=[20, 29, 39, 49], labels=['20', '30', '40'] )
test['age']  = pd.cut( x=test['age'], bins=[20, 29, 39, 49],  labels=['20', '30', '40'] )
```

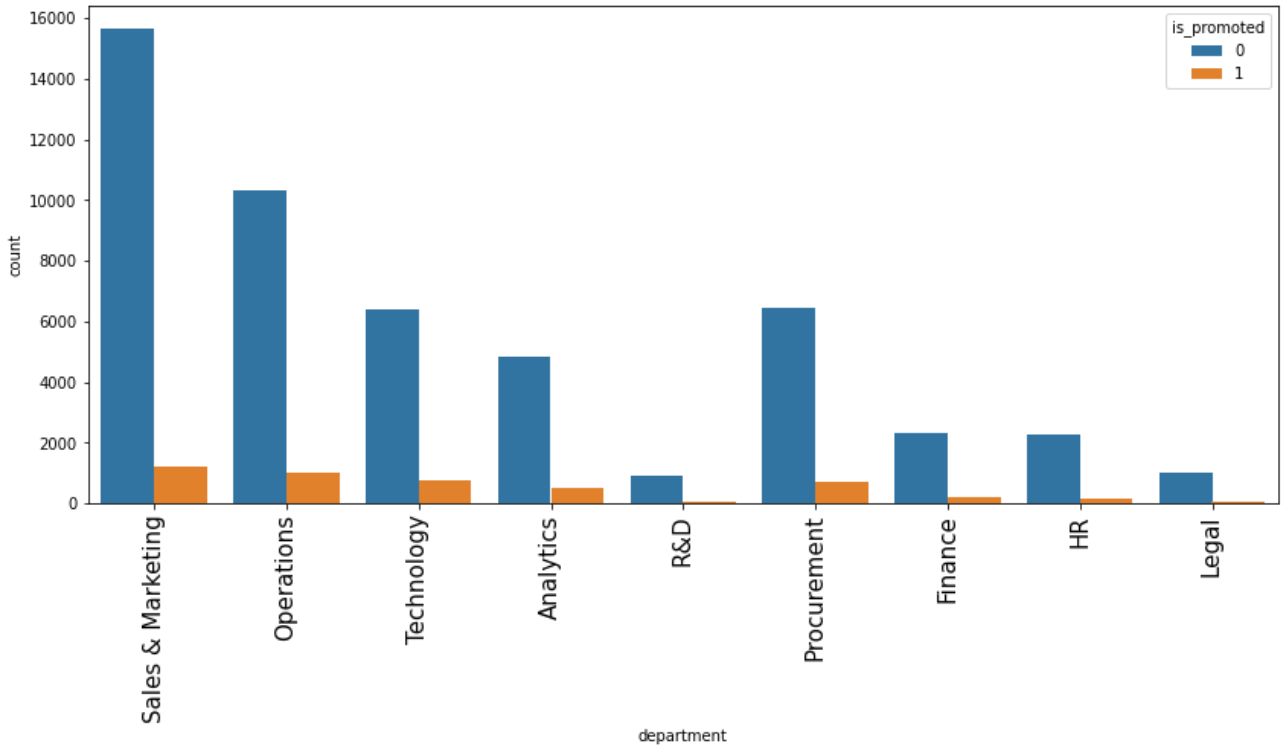

```
# displot -> plot a univariate(Single Feature) distribution of observations.

sns.distplot(train['length_of_service'])

train['length_of_service'] = pd.cut( x=train['length_of_service'], bins=[20, 29, 39, 49],
test['length_of_service']  = pd.cut( x=test['length_of_service'], bins=[20, 29, 39, 49],
```




```
# displot -> plot a univariate(Single Feature) distribution of observations.

sns.distplot(train['avg_training_score'])

train['avg_training_score'] = pd.cut( x=train['avg_training_score'], bins=[20, 29, 39, 49]
test['avg_training_score']  = pd.cut( x=test['avg_training_score'], bins=[20, 29, 39, 49],
```

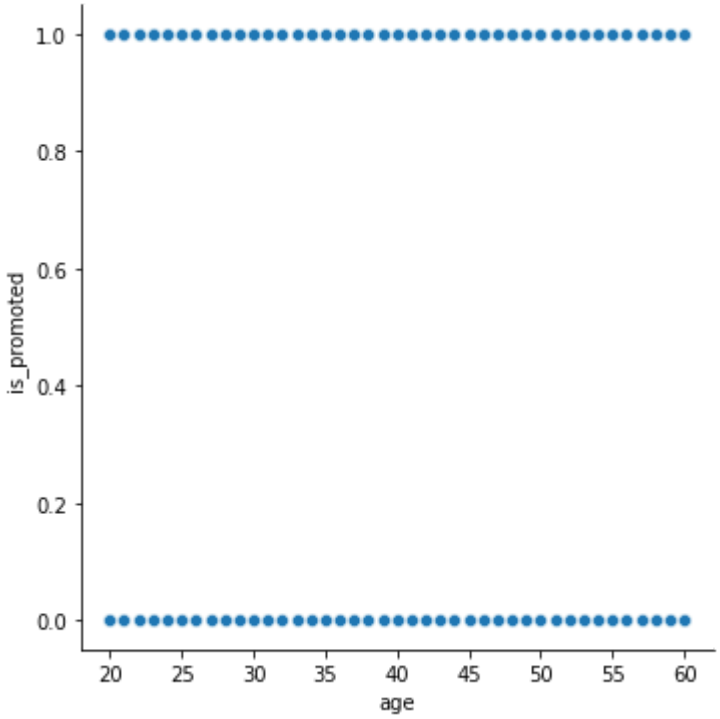



```
plt.figure(figsize=(14,6))
sns.countplot(x='department', hue='is_promoted', data=train)
plt.xticks(rotation=90,fontsize=15)
plt.show()
```



ratio of people promoted in technology and procurement is higher

```
sns.relplot( x='age',y='is_promoted',data=train)
plt.show()
```




```
pd.crosstab(index=train['recruitment channel'],columns='count')
```

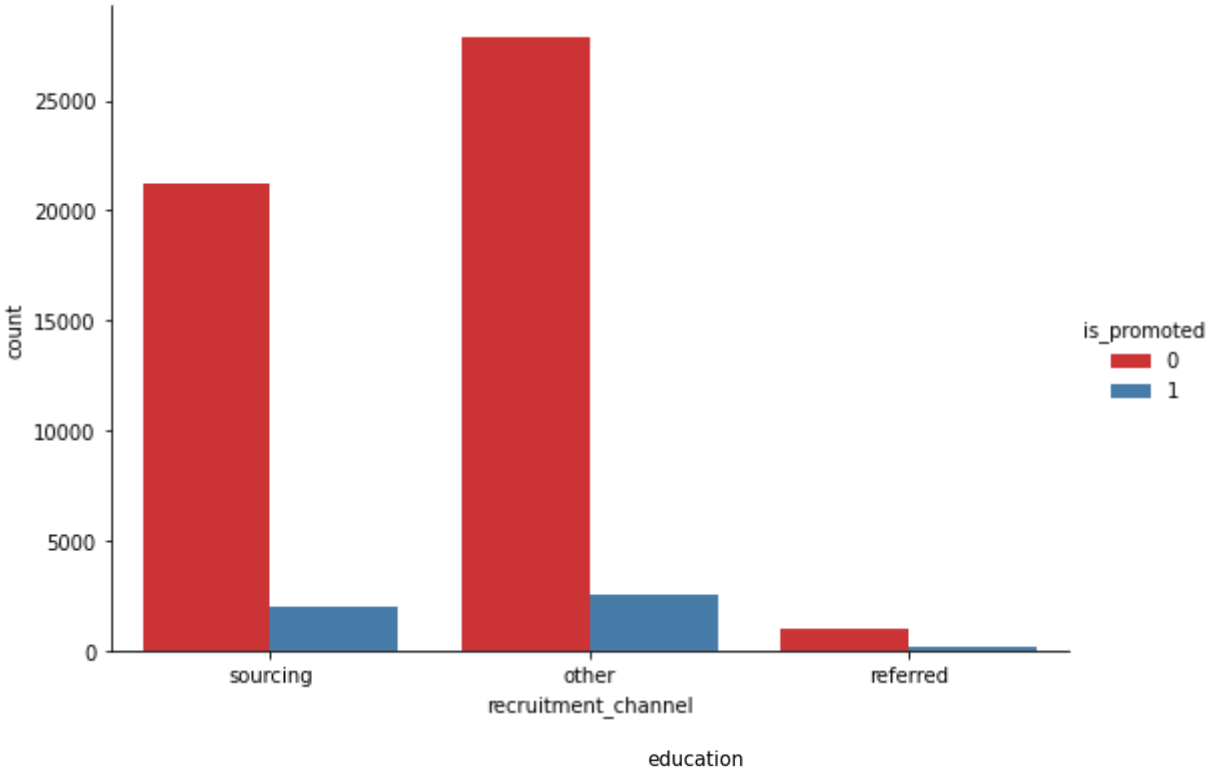
col_0	count
recruitment_channel	
other	30446
referred	1142
sourcing	23220

```
plt.figure(figsize=(16,10))
sns.countplot(x='region',hue='is_promoted',data=train,palette='Greens')
plt.xlabel('region',fontsize = 15)
plt.legend(loc=5,fontsize=20)
plt.ylabel('Count',fontsize=15)
plt.xticks(rotation=90,fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```


```
plt.figure(figsize=(16,10))
sns.countplot(x='education',hue='is_promoted',data=train,palette='Greens')
plt.xlabel('education',fontsize = 15)
plt.legend(loc=5,fontsize=20)
plt.ylabel('Count',fontsize=15)
plt.xticks(rotation=90,fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```


```
sns.catplot(hue='is_promoted',x='recruitment_channel',data=train,kind='count',aspect=1.5,
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4f911fa20>
```

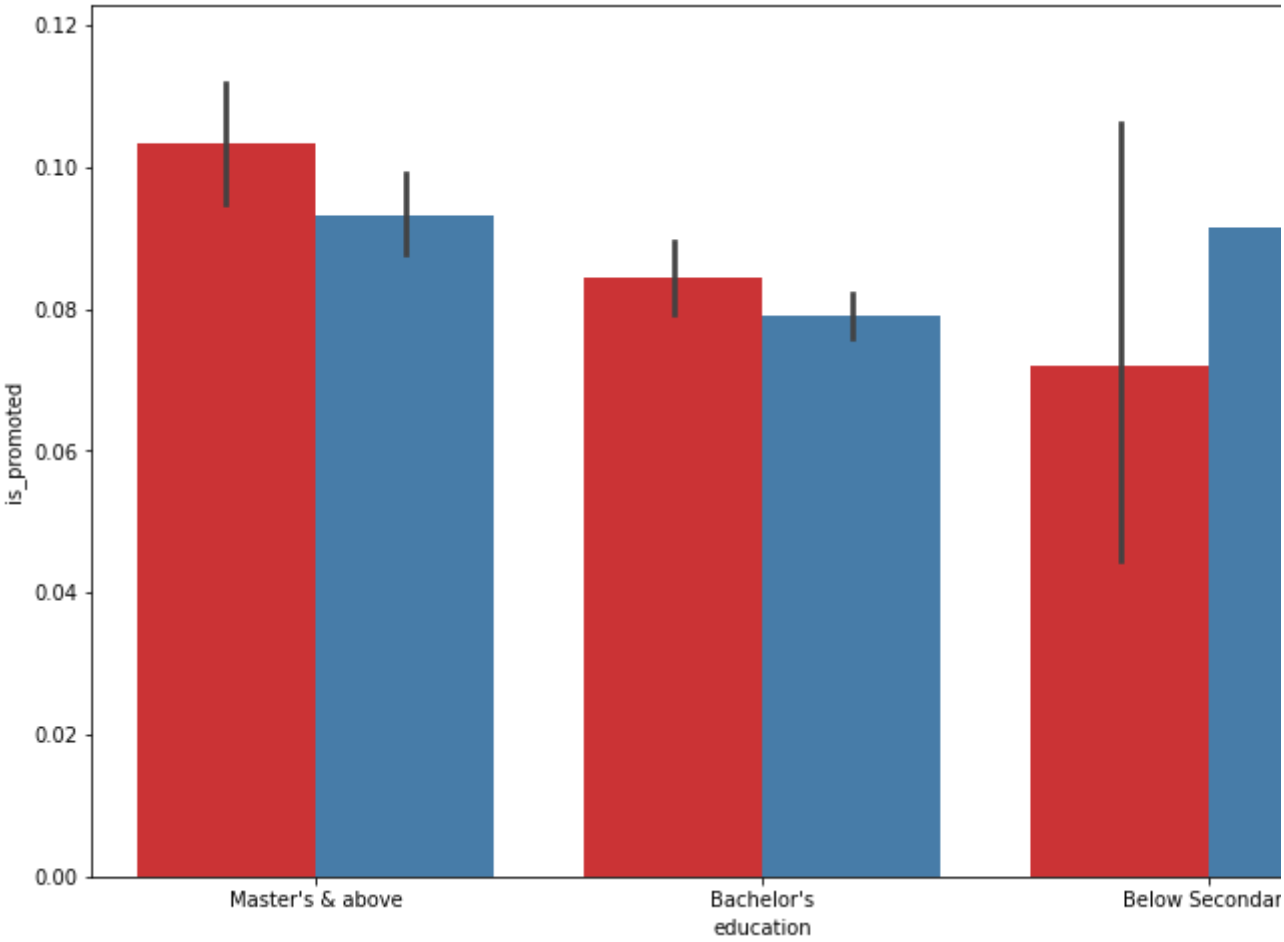



```
plt.figure(figsize=[12,8])
sns.barplot(x='department',y='is_promoted',hue='gender',data=train, palette="Set1")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4f8f31fd0>
```

```
plt.figure(figsize=[12,8])
sns.barplot(x='education',y='is_promoted',hue='gender',data=train, palette="Set1")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4f8f10b38>
```

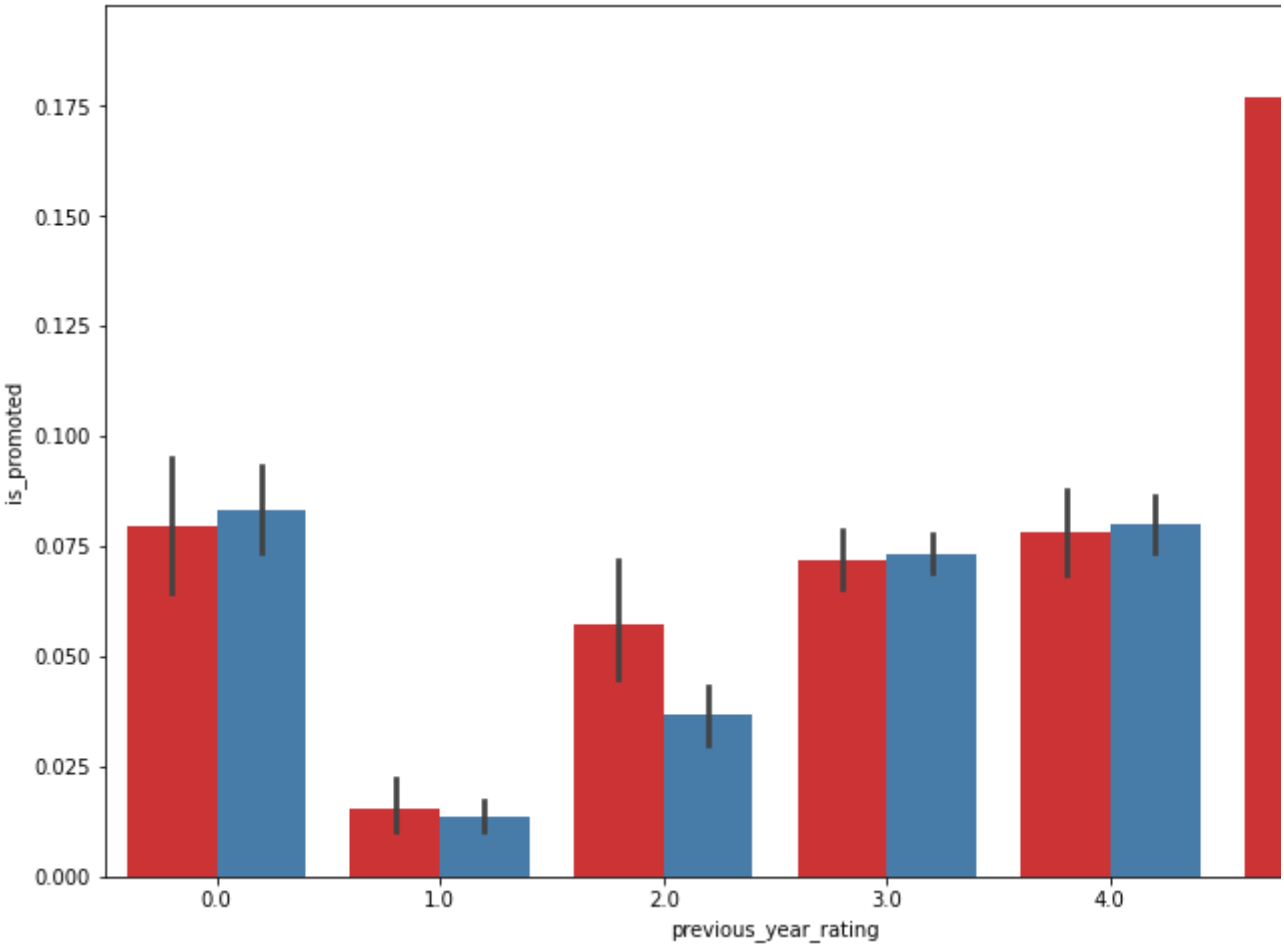



```
plt.figure(figsize=[12,8])
sns.barplot(x='recruitment_channel',y='is_promoted',hue='gender',data=train, palette="Set1
```

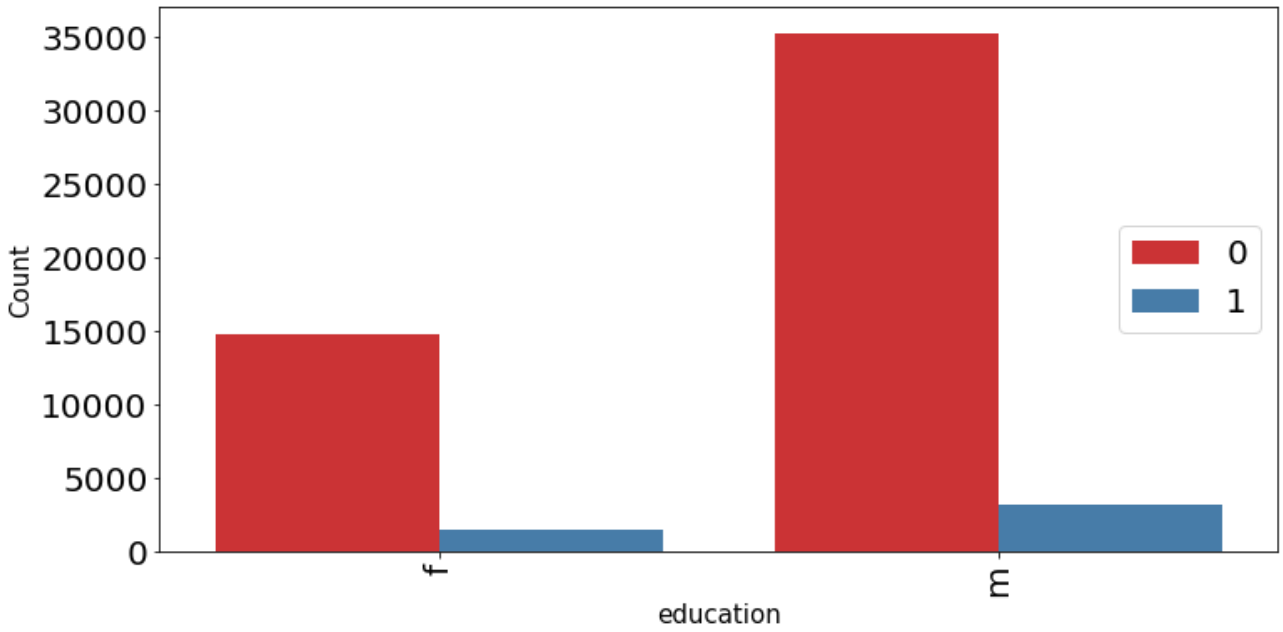
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4f985bac8>
```


```
plt.figure(figsize=[12,8])
sns.barplot(x='previous_year_rating',y='is_promoted',hue='gender',data=train, palette="Set
```

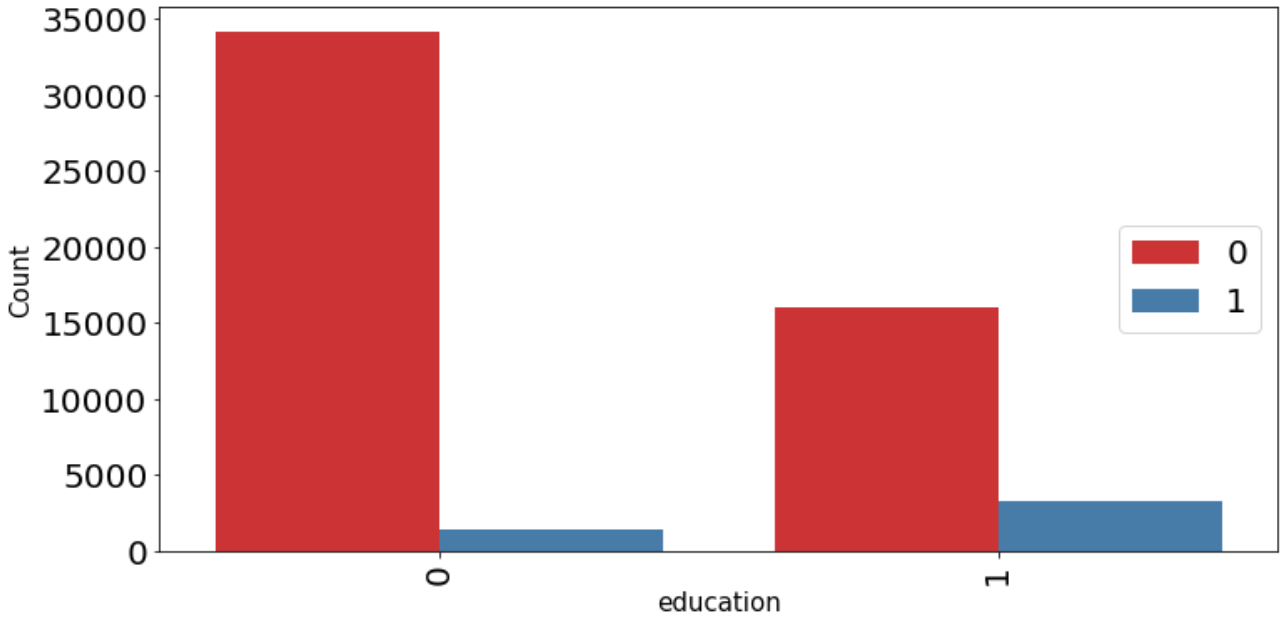
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4f8c062e8>
```




```
plt.figure(figsize=(12,6))
sns.countplot(x='gender',hue='is_promoted',data=train,palette='Set1')
plt.xlabel('education',fontsize = 15)
plt.legend(loc=5,fontsize=20)
plt.ylabel('Count',fontsize=15)
plt.xticks(rotation=90,fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```

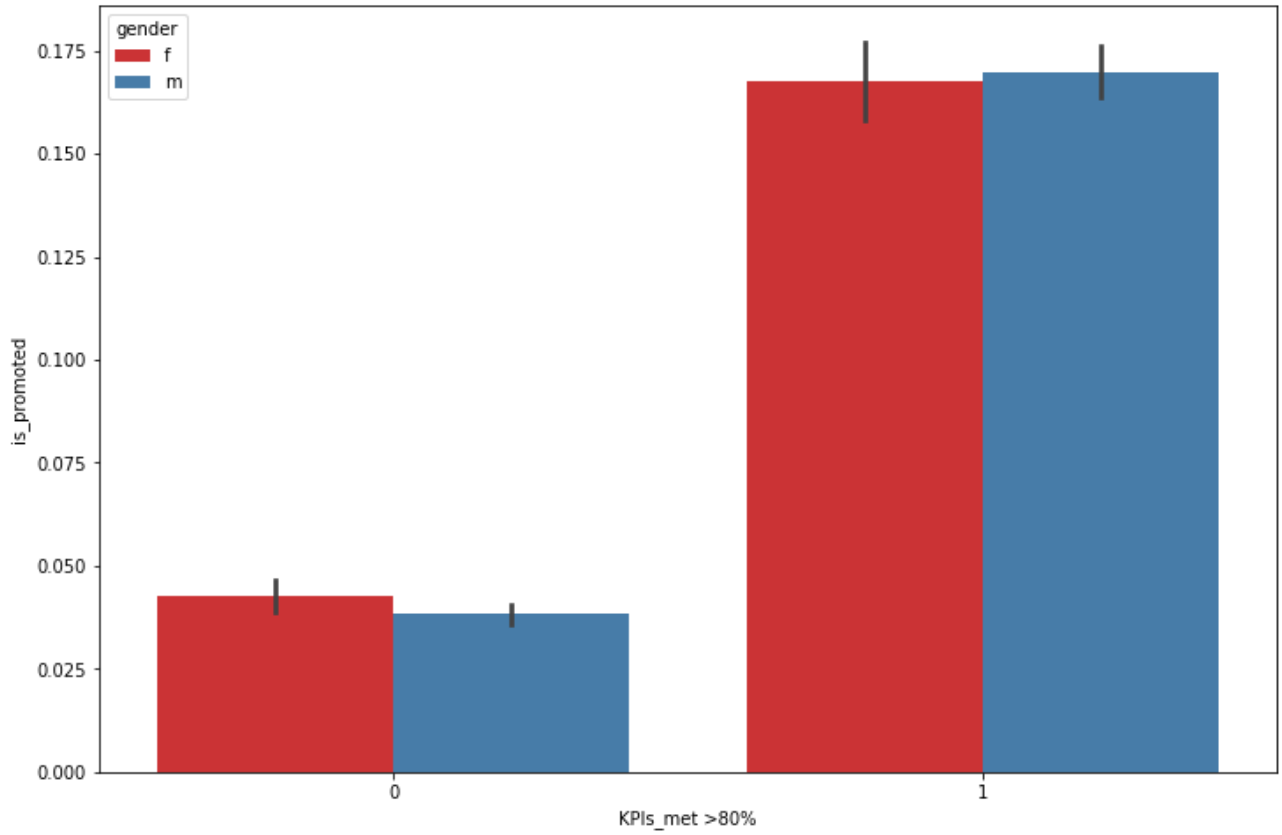



```
plt.figure(figsize=(12,6))
sns.countplot(x='KPIs_met >80%',hue='is_promoted',data=train,palette='Set1')
plt.xlabel('education',fontsize = 15)
plt.legend(loc=5,fontsize=20)
plt.ylabel('Count',fontsize=15)
plt.xticks(rotation=90,fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```




```
plt.figure(figsize=[12,8])
sns.barplot(x='KPIs_met >80%',y='is_promoted',hue='gender',data=train, palette="Set1")
```

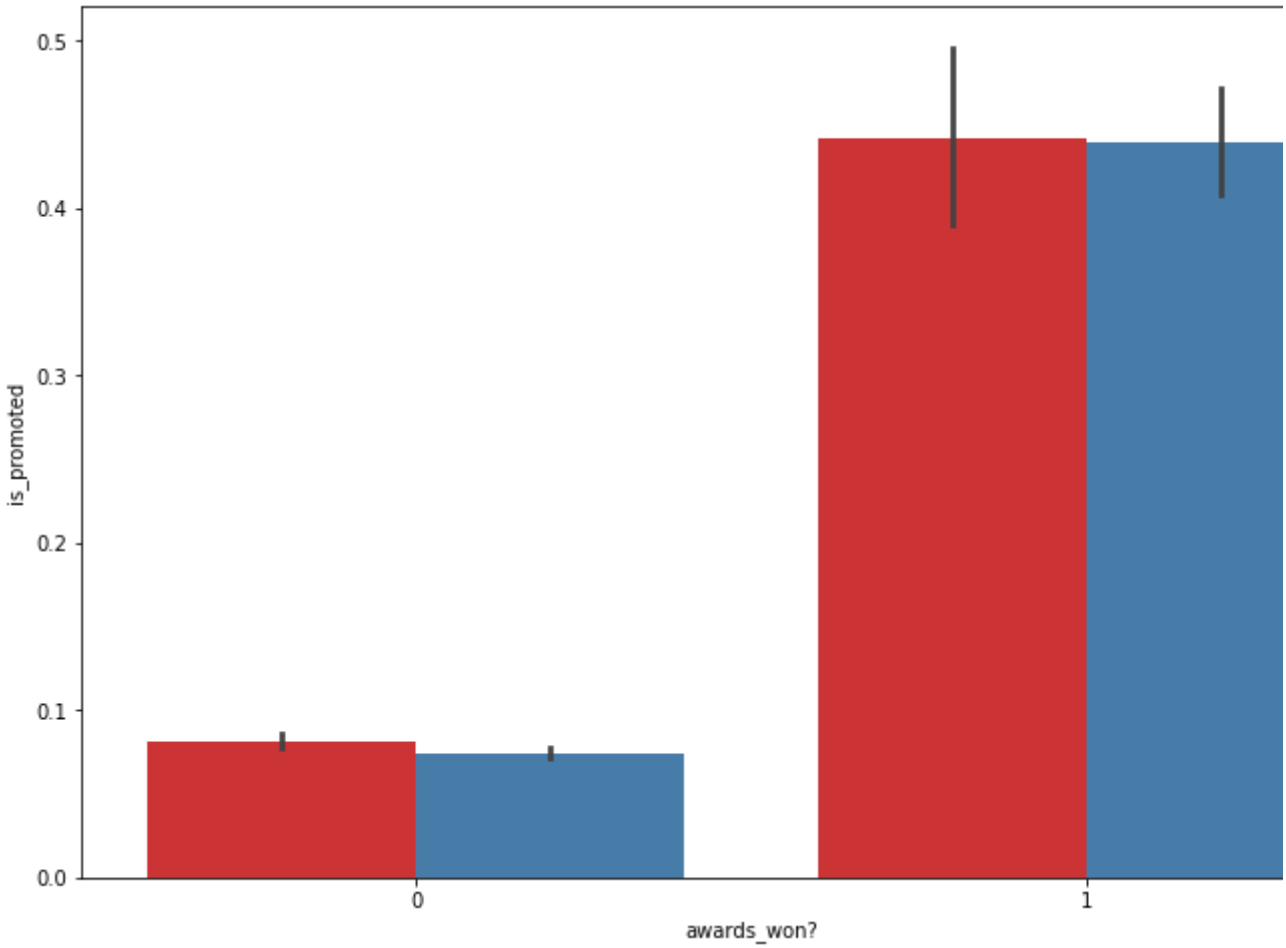
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4f91457f0>
```




```
plt.figure(figsize=(12,6))
sns.countplot(x='awards_won?',hue='is_promoted',data=train,palette='Set1')
plt.xlabel('education',fontsize = 15)
plt.legend(loc=5,fontsize=20)
plt.ylabel('Count',fontsize=15)
plt.xticks(rotation=90,fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```


```
plt.figure(figsize=[12,8])
sns.barplot(x='awards_won?',y='is_promoted',hue='gender',data=train, palette="Set1")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4f9998080>
```



5. Feature Engineering :

```
train.head(3)
```

	employee_id	department	region	education	gender	recruitment_channel	no_of_
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	
1	65141	Operations	region_22	Bachelor's	m	other	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	

```
train.region.value_counts()
```

```
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: region, dtype: int64
```

6. Split Train Data into Predictors(Independent) & Target(Dependent) :

```
X_train=train.drop('is_promoted',axis=1)
y_train=train['is_promoted']

X_test=test
```

7.1 Data Encoding : Label Encoding, OneHot Encoding :

```
def data_encoding( encoding_strategy , encoding_data , encoding_columns ):

    if encoding_strategy == "LabelEncoding":

        Encoder = LabelEncoder()
        for column in encoding_columns :
            print("column",column )
            encoding_data[ column ] = Encoder.fit_transform(tuple(encoding_data[ column ])
```

```
    elif encoding_strategy == "OneHotEncoding":
        encoding_data = pd.get_dummies(encoding_data)

    dtypes_list =['float64','float32','int64','int32']
    encoding_data.astype( dtypes_list[0] ).dtypes

    return encoding_data


encoding_columns  = [ "region", "age","department", "education", "gender", "recruitment_ch
encoding_strategy = [ "LabelEncoding", "OneHotEncoding"]


X_train_encode = data_encoding( encoding_strategy[1] , X_train , encoding_columns )
X_test_encode =  data_encoding( encoding_strategy[1] , X_test  , encoding_columns )


display(X_train_encode.head())
```

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_
0	65438	1	35	5.0	8	
1	65141	1	30	5.0	4	
2	7513	1	34	3.0	7	
3	2542	2	39	1.0	10	
4	48945	1	45	3.0	2	

```
display(X_test_encode.head())
```

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_
0	8724	1	24	0.0	1	
1	74430	1	31	3.0	5	
2	72255	1	31	1.0	4	
3	38562	3	31	2.0	9	
4	64486	1	30	4.0	7	

7.2 Data Scaling : StandardScaler, MinMaxScaler

```
def data_scaling( scaling_strategy , scaling_data , scaling_columns ):

    if    scaling_strategy =="RobustScaler" :
        scaling_data[scaling_columns] = RobustScaler().fit_transform(scaling_data[scaling_
```

```
    elif  scaling_strategy =="StandardScaler" :
        scaling_data[scaling_columns] = StandardScaler().fit_transform(scaling_data[scalir

    else:
        scaling_data[scaling_columns] = MinMaxScaler().fit_transform(scaling_data[scaling_


    return scaling_data


# RobustScaler is better in handling Outliers :

# QUOTED FOR DOCUMENTATION : https://scikit-learn.org/stable/modules/generated/sklearn.pre
scaling_strategy = ["StandardScaler","MinMaxScaler"]
X_train_scale = data_scaling( scaling_strategy[0] , X_train_encode , X_train_encode.columr
X_test_scale  = data_scaling( scaling_strategy [0] , X_test_encode  , X_test_encode.columr

# Display Scaled Train and Test Features :

display(X_train_scale.head())
display(X_train_scale.head())
```

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service
0	1.161858	-0.415276	0.025598	1.283878	0.500460
1	1.148709	-0.415276	-0.627135	1.283878	-0.437395
2	-1.402741	-0.415276	-0.104948	-0.052623	0.265996
3	-1.622829	1.226063	0.547785	-1.389124	0.969387
4	0.431639	-0.415276	1.331064	-0.052623	-0.906322

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service
0	1.161858	-0.415276	0.025598	1.283878	0.500460
1	1.148709	-0.415276	-0.627135	1.283878	-0.437395
2	-1.402741	-0.415276	-0.104948	-0.052623	0.265996
3	-1.622829	1.226063	0.547785	-1.389124	0.969387
4	0.431639	-0.415276	1.331064	-0.052623	-0.906322

8. Create Baseline ML Model for Binary Classification Problem :

```
# Create a Dictionary (Key->Value Pairs) for "ML Model Name"-> "ML Model Functions with Hy

Classifiers = {'XGBoost' : XGBClassifier(learning_rate =0.10,
                                         n_estimators=500,
                                         max_depth=5,
```

```
                                        subsample = 0.8,
                                        verbosity = 1,
                                        scale_pos_weight = 2.1,
                                        updater ="grow_histmaker",
                                        base_score  = 0.25),

              'CatBoost' : CatBoostClassifier(learning_rate=0.20,
                                              n_estimators=500,
                                              subsample=0.0015,

                                              max_depth=5,
                                              scale_pos_weight=2.1),

              'LightGBM' : LGBMClassifier(subsample_freq = 2,
                                          objective ="binary",
                                          importance_type = "gain",
                                          verbosity = 1,
                                          max_bin = 60,
                                          num_leaves = 400,
                                          boosting_type = 'dart',
                                          learning_rate=0.20,
                                          n_estimators=500,
                                          max_depth=5,
                                          scale_pos_weight=2.1)
 }
```

9. Improve ML Model with Voting Classifier with MODEL Evaluation METRIC - "F1" and Predict Target "is_promoted" :

```
voting=VotingClassifier(estimators=Classifiers.items(),voting='soft',weights=[4,4,4.1])
```

```
voting.fit(X_train_scale,y_train)
```

```
     425:    learn: 0.2350271        total: 8.92s    remaining: 1.55s
     426:    learn: 0.2349391        total: 8.94s    remaining: 1.53s
     427:    learn: 0.2348907        total: 8.96s    remaining: 1.51s
     428:    learn: 0.2347753        total: 8.98s    remaining: 1.49s
     429:    learn: 0.2347566        total: 9s       remaining: 1.47s
     430:    learn: 0.2347406        total: 9.02s    remaining: 1.44s
     431:    learn: 0.2346953        total: 9.04s    remaining: 1.42s
     432:    learn: 0.2346544        total: 9.06s    remaining: 1.4s
     433:    learn: 0.2346036        total: 9.08s    remaining: 1.38s
     434:    learn: 0.2345705        total: 9.1s     remaining: 1.36s
     435:    learn: 0.2345235        total: 9.12s    remaining: 1.34s
     436:    learn: 0.2344286        total: 9.14s    remaining: 1.32s
     437:    learn: 0.2342816        total: 9.17s    remaining: 1.3s
     438:    learn: 0.2342452        total: 9.19s    remaining: 1.28s
     439:    learn: 0.2342196        total: 9.21s    remaining: 1.25s
     440:    learn: 0.2341382        total: 9.23s    remaining: 1.23s
     441:    learn: 0.2341019        total: 9.25s    remaining: 1.21s
     442:    learn: 0.2340856        total: 9.27s    remaining: 1.19s
     443:    learn: 0.2340454        total: 9.29s    remaining: 1.17s
```

```
444:    learn: 0.2339660        total: 9.31s    remaining: 1.15s
445:    learn: 0.2339458        total: 9.33s    remaining: 1.13s
446:    learn: 0.2338822        total: 9.35s    remaining: 1.11s
447:    learn: 0.2338365        total: 9.38s    remaining: 1.09s
448:    learn: 0.2338056        total: 9.4s     remaining: 1.07s
449:    learn: 0.2337521        total: 9.42s    remaining: 1.05s
450:    learn: 0.2337453        total: 9.44s    remaining: 1.02s
451:    learn: 0.2336671        total: 9.45s    remaining: 1s
452:    learn: 0.2336496        total: 9.47s    remaining: 983ms
453:    learn: 0.2336435        total: 9.49s    remaining: 961ms
454:    learn: 0.2336062        total: 9.51s    remaining: 940ms
455:    learn: 0.2335613        total: 9.52s    remaining: 919ms
456:    learn: 0.2333339        total: 9.54s    remaining: 898ms
457:    learn: 0.2333112        total: 9.57s    remaining: 877ms
458:    learn: 0.2332515        total: 9.59s    remaining: 857ms
459:    learn: 0.2332208        total: 9.61s    remaining: 836ms
460:    learn: 0.2331775        total: 9.63s    remaining: 815ms
461:    learn: 0.2331271        total: 9.65s    remaining: 794ms
462:    learn: 0.2330226        total: 9.67s    remaining: 773ms
463:    learn: 0.2330022        total: 9.69s    remaining: 752ms
464:    learn: 0.2328919        total: 9.71s    remaining: 731ms
465:    learn: 0.2328179        total: 9.73s    remaining: 710ms
466:    learn: 0.2327958        total: 9.75s    remaining: 689ms
467:    learn: 0.2327556        total: 9.77s    remaining: 668ms
468:    learn: 0.2326993        total: 9.79s    remaining: 647ms
469:    learn: 0.2326733        total: 9.82s    remaining: 627ms
470:    learn: 0.2326521        total: 9.84s    remaining: 606ms
471:    learn: 0.2326251        total: 9.86s    remaining: 585ms
472:    learn: 0.2325921        total: 9.88s    remaining: 564ms
473:    learn: 0.2324572        total: 9.9s     remaining: 543ms
474:    learn: 0.2323826        total: 9.92s    remaining: 522ms
475:    learn: 0.2323490        total: 9.94s    remaining: 501ms
476:    learn: 0.2323463        total: 9.96s    remaining: 480ms
477:    learn: 0.2322885        total: 9.98s    remaining: 460ms
478:    learn: 0.2322324        total: 10s      remaining: 439ms
479:    learn: 0.2321740        total: 10s      remaining: 418ms
480:    learn: 0.2321368        total: 10s      remaining: 397ms
481:    learn: 0.2320152        total: 10.1s    remaining: 376ms
482:    learn: 0.2317415        total: 10.1s    remaining: 355ms
```

```
voting_pred=voting.predict_proba(X_test_scale)[::,1]
```

10. Result Submission, Check Leaderboard & Improve "F1" Score :

```
# Round off the Probability Results :
predictions = [int(round(value)) for value in voting_pred]

# Create a Dataframe Table for Submission Purpose :
final = pd.DataFrame({'employee_id': test["employee_id"], 'is_promoted' : predictions})
final.to_csv('submission_HR_Analytics.csv',index=False)
```