WORKFORCE ANALYSIS

Employee Performance Analysis

Mentored by:

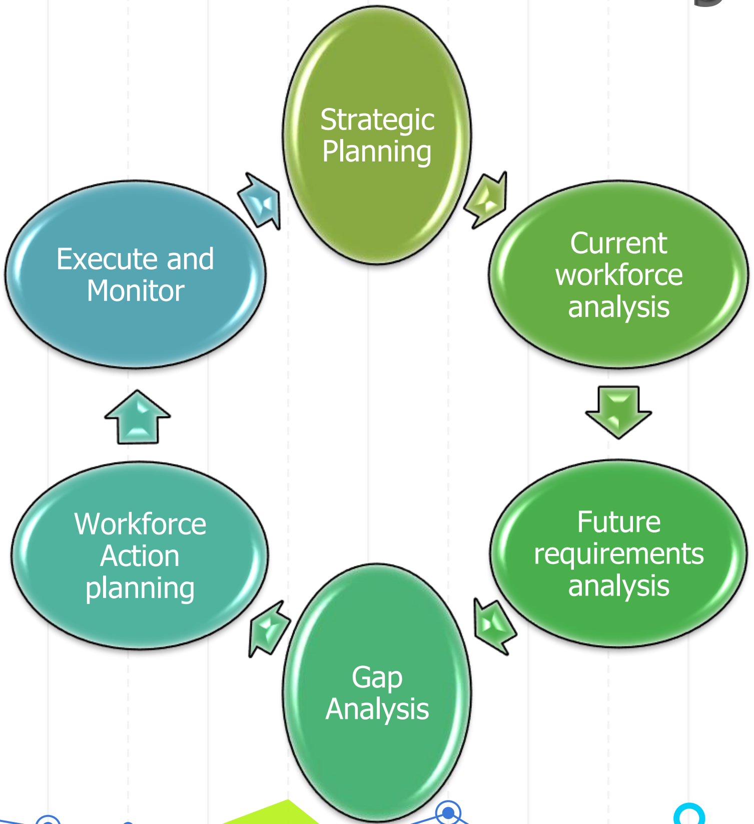
Mr Muppidi Srikar

Prepared by :

Yash Agarwal
Aman kumar
Ashwin Yenigalla
Sara Faruqui

Index

Strategic Workforce Planning Model



Project on Employee Performance Analysis

Objective: The performance of various employees in an organisation varies and so is the probability of each employee getting promoted. Not getting promoted could have a direct bearing against employee attrition and hence the HR department would like to know the probability that an employee will get promoted. The objective of this project is to predict whether an employee will get promoted or not and also understand the factors which impact the promotion. This helps HR team to plan for back up resources prior to rating cycle against the resources who have high changes of not getting promoted.

1

Explanation of Dataset

The dataset consists of the following information of 54808 employees:

Variables in the Dataset

X-Variable

employee_id	Unique employee ID
department:	Department in which the employee works
Region	Employee region
Education	Education level of the employee
Gender	Gender of the employee
recruitment_channel	Channel through which employee was recruited
no_of_trainings:	of training programs the employee has undergone
Age	Age of the employee
previous_year_rating	Performance rating of the employee in the previous year
length_of_service:	Experience of the employee
KPIs_met >80%:	Has the employee met more than 80% of the KPIs. 0-No;1-Yes
awards_won	Has the employee won any awards? 0-No;1-Yes
avg_training_score:	Average training score of the employee

Target-Variable

is_promoted	Y variable – 0-Not promoted; 1-Promoted

Project Instruction

- Perform the required data pre-processing to treat for missing values and outliers
- Perform exploratory data analysis to visualise the spread of each of the X variables and the relationship between the various X variables and the Y variable
- Use the data provided to create employee segments using clustering and visually explore the % of employees promoted in each segment.
- Divide the given data into train and test sets
- Build a model to predict whether an employee will get promoted or not
- Evaluate the model based on model performance measures for classification and recommend the most suitable model.
- Come up with recommendations / actionable insights based on feature importance scores derived from the model.

Dataset Information

Dataset has 54808 instances (rows) and 14 attributes (columns).

❑ Number of Attributes

- 14 Columns : 3 Continuous and 11 Categorical.

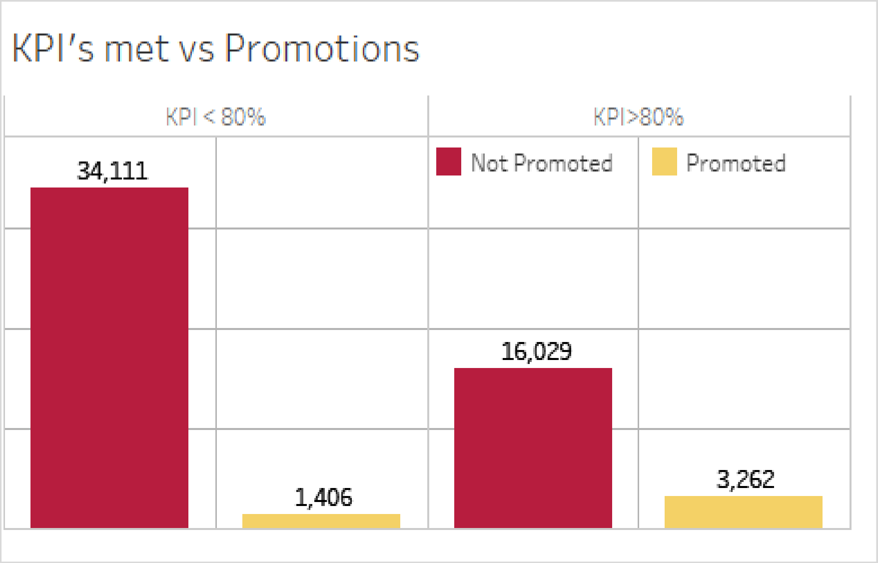
❑ Attributes Information

- Attribute 1 : employee_id with continuous figures.
- Attribute 2 : Department (Categorical) with 9 different department names.
- Attribute 3 : Region (Categorical) with 34 different regions.
- Attribute 4 : Education (Categorical) with 4 different degree names.
- Attribute 5 : Gender (Categorical) with Male and Female.
- Attribute 6 : Recruitment Channel (Categorical) with 3 different channels.
- Attribute 7 : no_of_trainings (Categorical) with 10 different training.
- Attribute 8 : age (Continuous) varies as Upper Limit – 60 and lower limit -20.
- Attribute 9 : previous_year_rating (Categorical) varies from 0-5.
- Attribute 10 : length_of_service (Categorical) time spent varies from 1-37 years in Company.
- Attribute 11 : KPIs_met >80% (Categorical) with 0 and 1.
- Attribute 12 : awards_won (Categorical) with 0 and 1.
- Attribute 13 : avg_training_score (Continuous) with 61 different averages.
- Attribute 14 : is_promoted (Categorical) with 0 and 1 as targets.

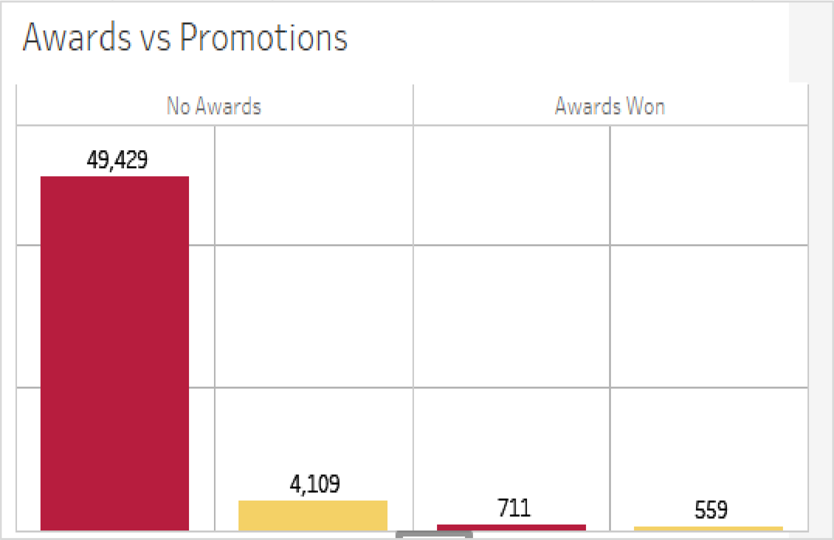
Data Visualisation with Tableau

2

Performance Metrics with Variables

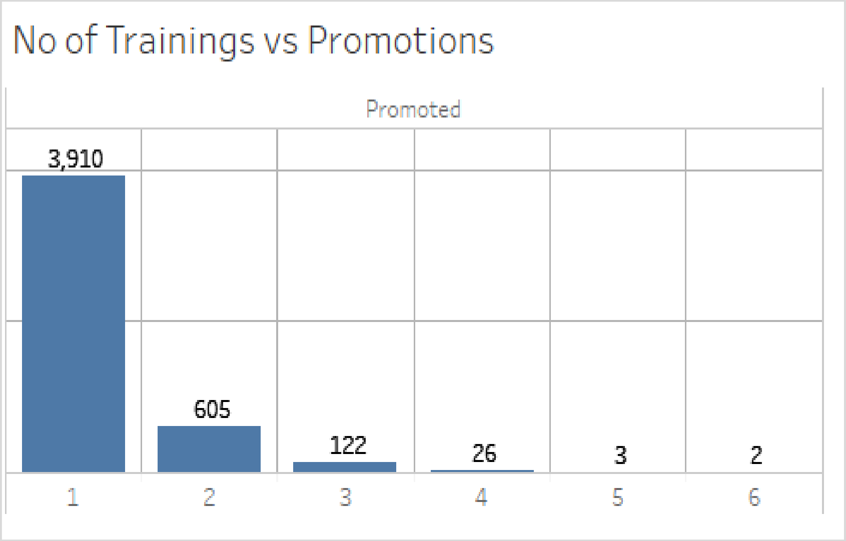


Insights : More people got promoted in case of KPI>80%

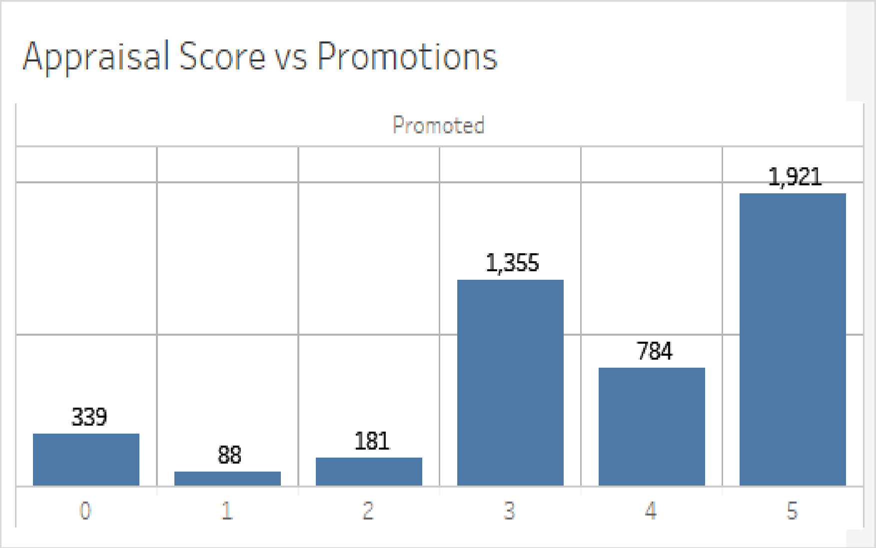


Insights : Very few employees were able to win awards.

Performance Metrics with Variables

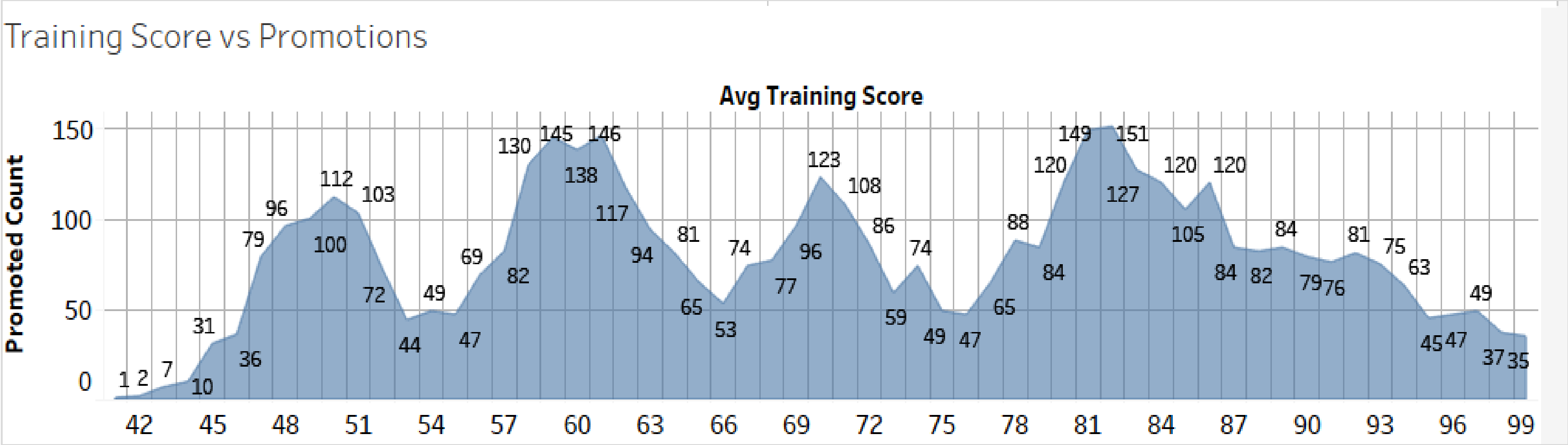


Insights : Maximum number of promotions were given on single no. of training.



Insights: Employees who scored more in Appraisal had more chances of promotion

Visualisation of Training Score VS Promotion



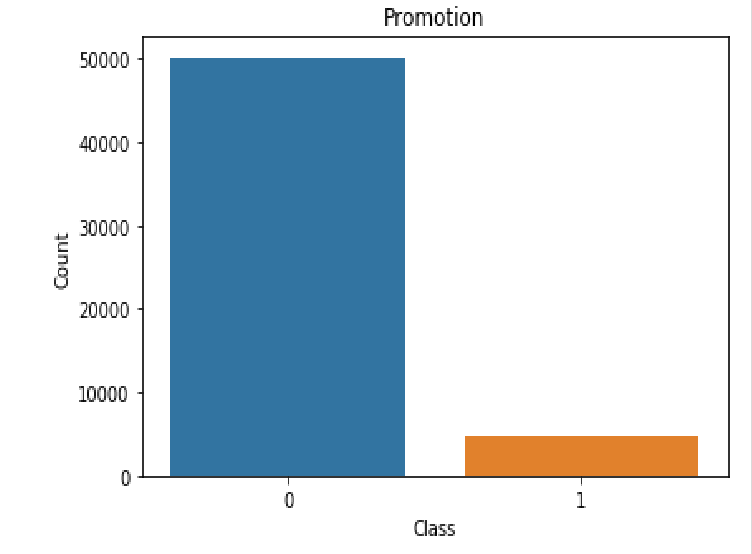
Exploratory Data Analysis

3

Univariate Analysis (Categorical):

Target Variable : is promoted

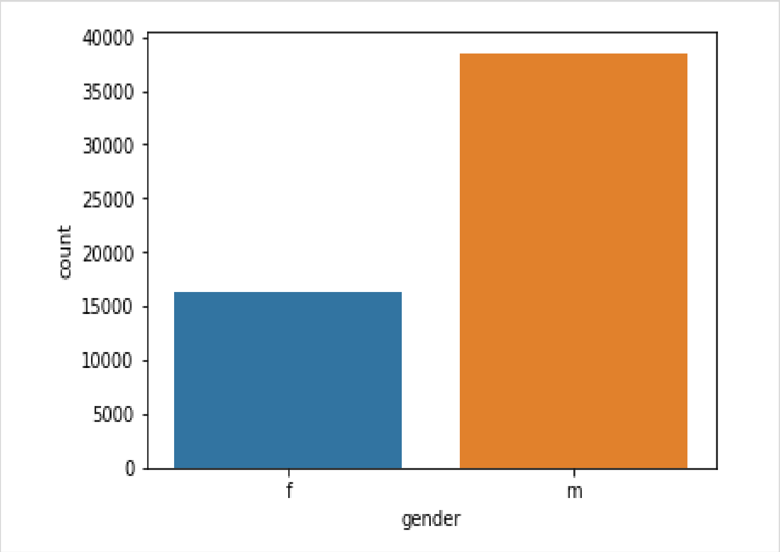
```
# Target Variable
grph = sns.countplot(employee.is_promoted)
grph.set(xlabel = "Class", ylabel = 'Count',title = 'Promotion')
plt.tight_layout()
plt.show()
```



Insights : Target Variable has highly Imbalanced data, Very few promotions are given**.**

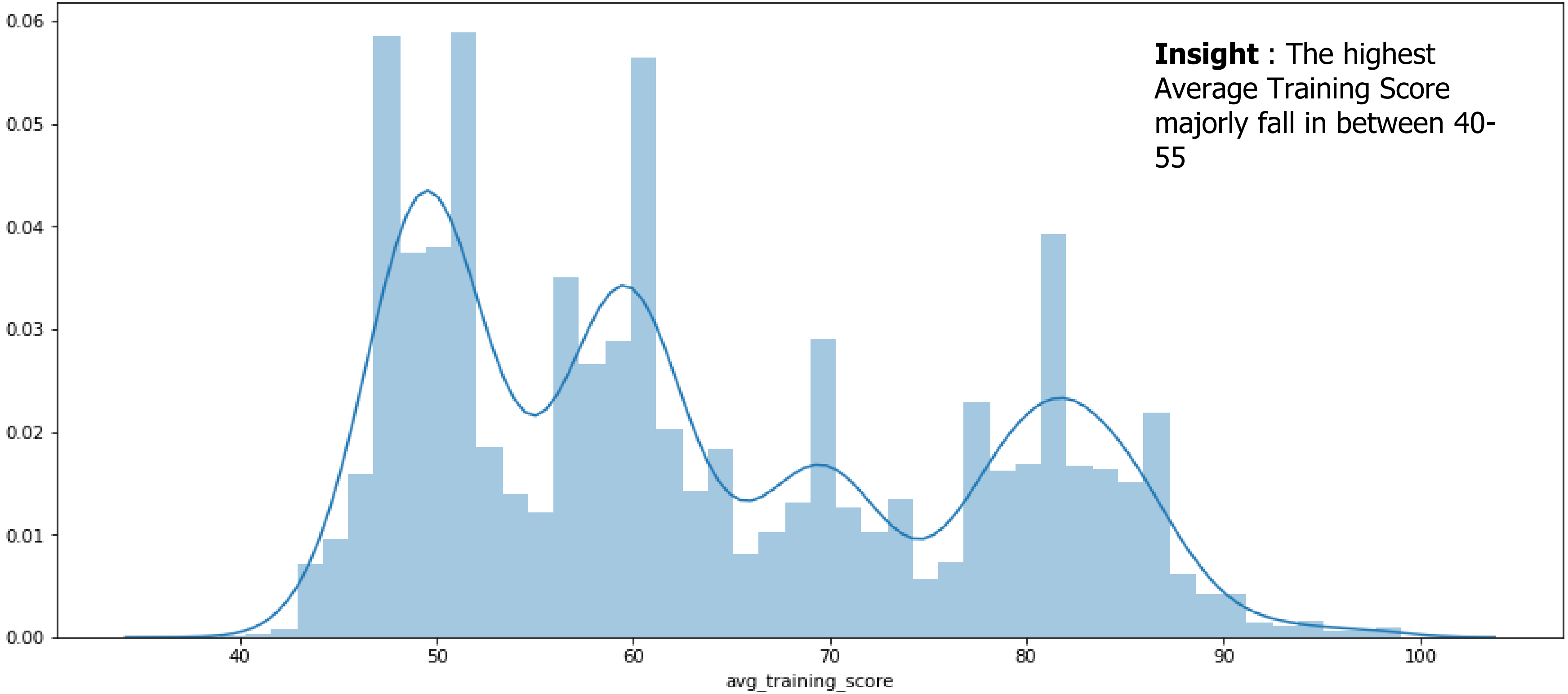
Attribute: Gender

```
sns.countplot(x = 'gender',data = employee)  # Male
plt.show()
```



Insights : The Data has been dominated by Males.

Attribute : Average Training Score

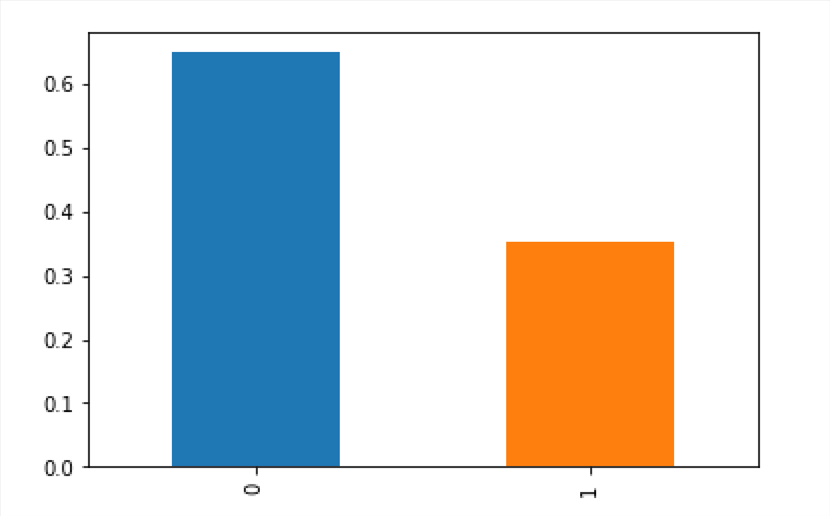


Insight : The highest Average Training Score majorly fall in between 40-55

Attribute : KPIs met >80%

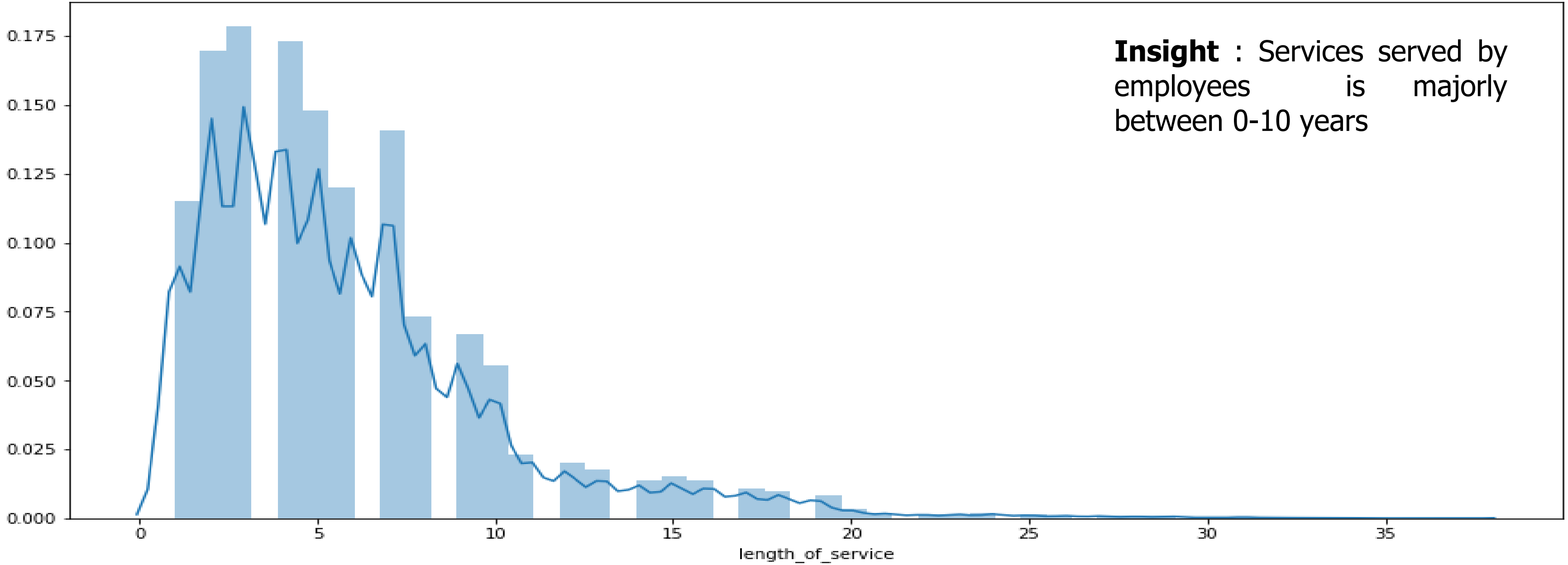
```
employee['KPIs_met >80%'].value_counts(normalize = True)
```

```
0    0.648026
1    0.351974
Name: KPIs_met >80%, dtype: float64
```



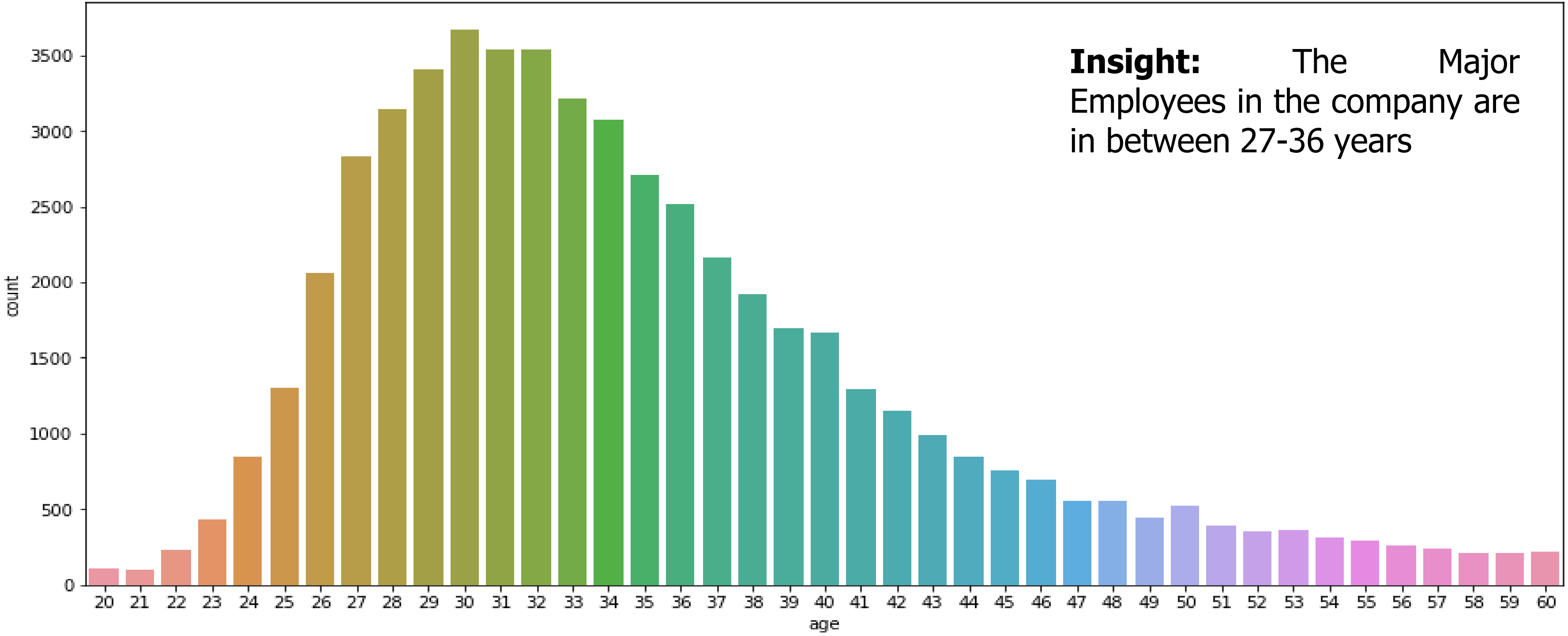
Insights : KPI with <80% are just double of KPI with >80%

Attribute : Length of Service



Insight : Services served by employees is majorly between 0-10 years

Attribute : Age



Insight: The Major Employees in the company are in between 27-36 years

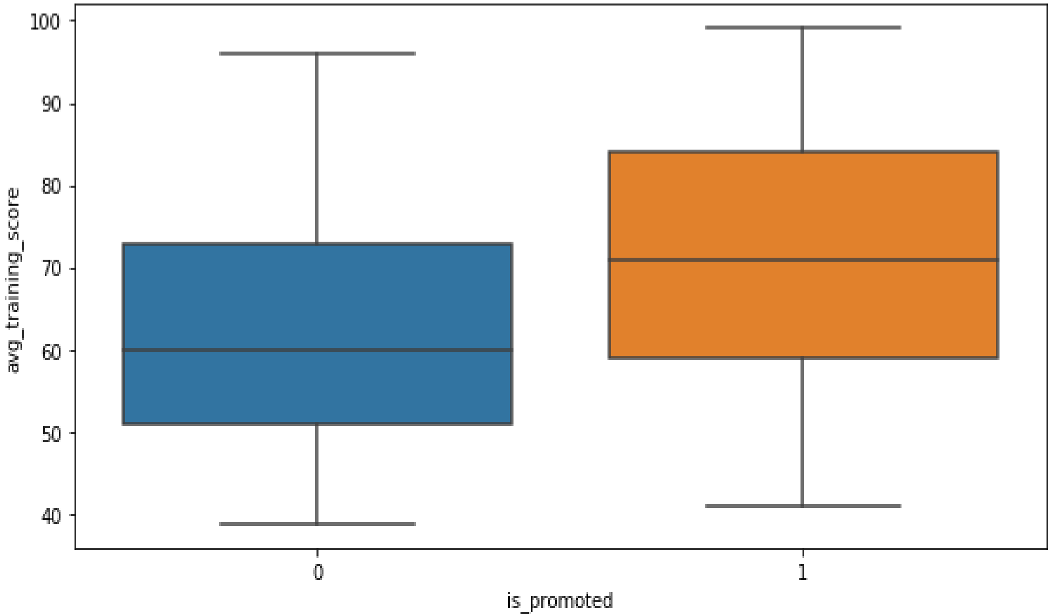
Bivariate Analysis:

- Dependent Variables
- Dependent Variable Vs Independent Variable

Attribute : x =[Promoted] y =[Average Training Score]

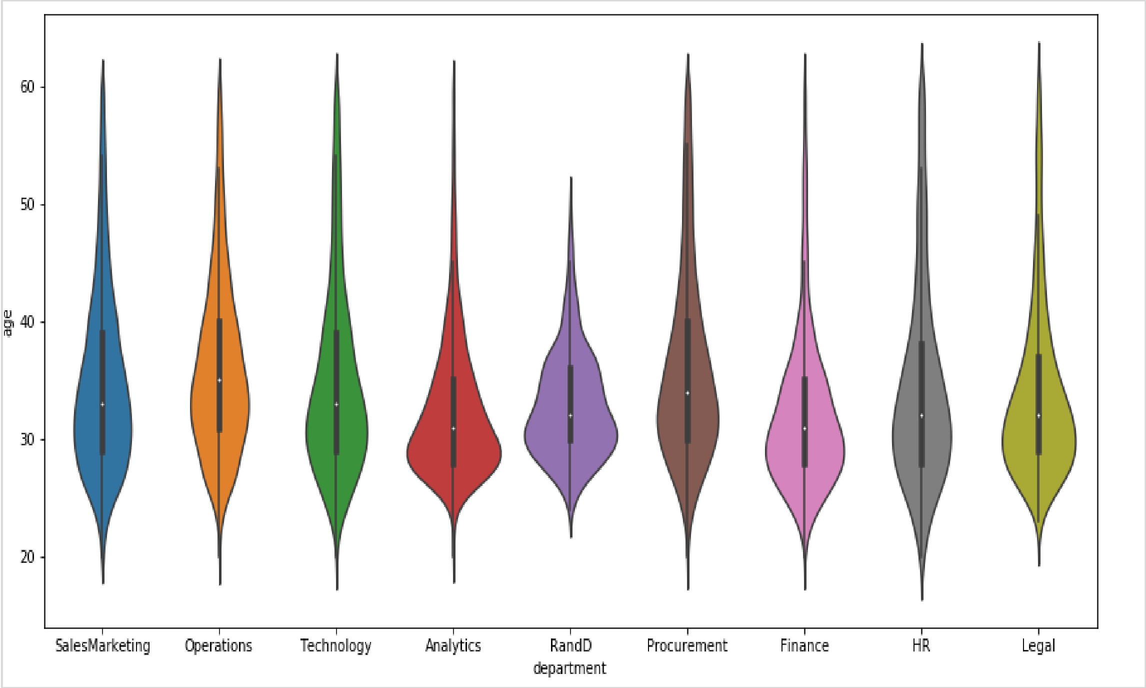
Insights :

1. The employee which are promoted have more Average Training Score than who are not promoted. This means More Training Score has more chance to get promote



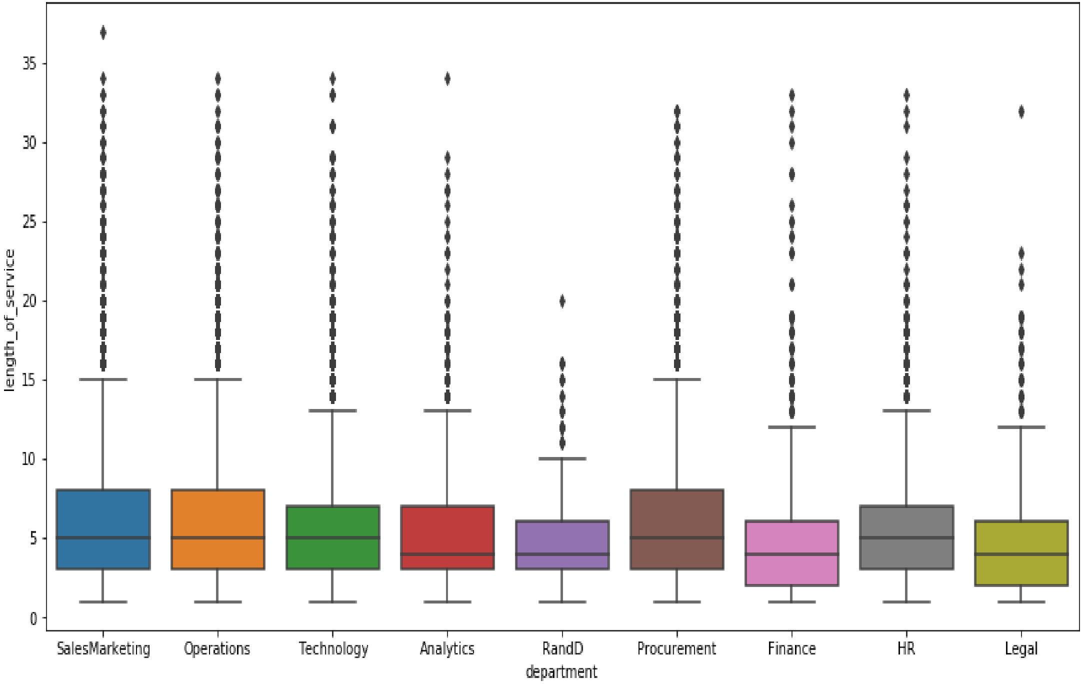
Attribute : x =[Department] y =[Age]

Insights:
1- Average age of people working in different departments are in range 32- 36.
2- Interesting part is R&D dept has quite younger people.(with analytics finance and legal.)
3- operations, procurement, technology, sales and marketing and HR have more people above 50.(especially compared to analytics and R&D)



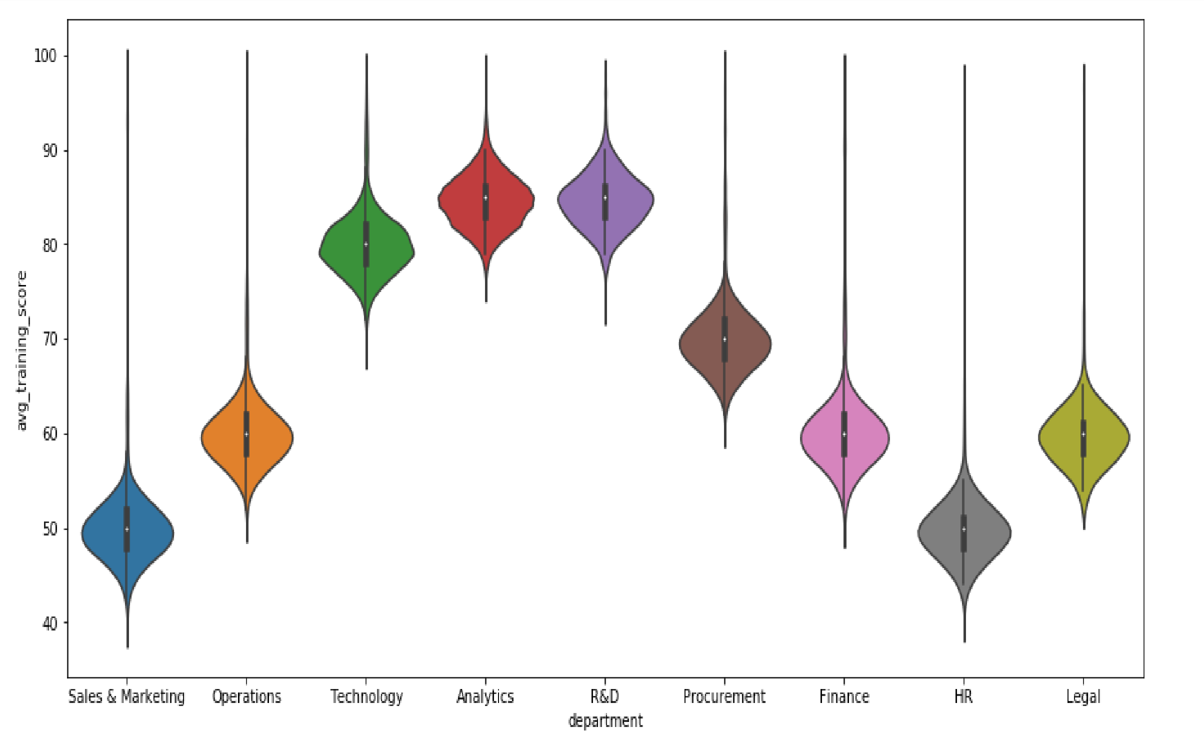
Attribute : x =[Department] y=[Length of Service]

Insights:
1. All departments have average length of service between 3-8 years.
2. Analytics and finance department has mostly less experienced (more young professionals) than compared to others.
3. Sales & marketing, Technology, Operations and HR has decent mix of experienced professionals (a good range of experience).



Attribute : x =[Department] y=[Average Training Score]

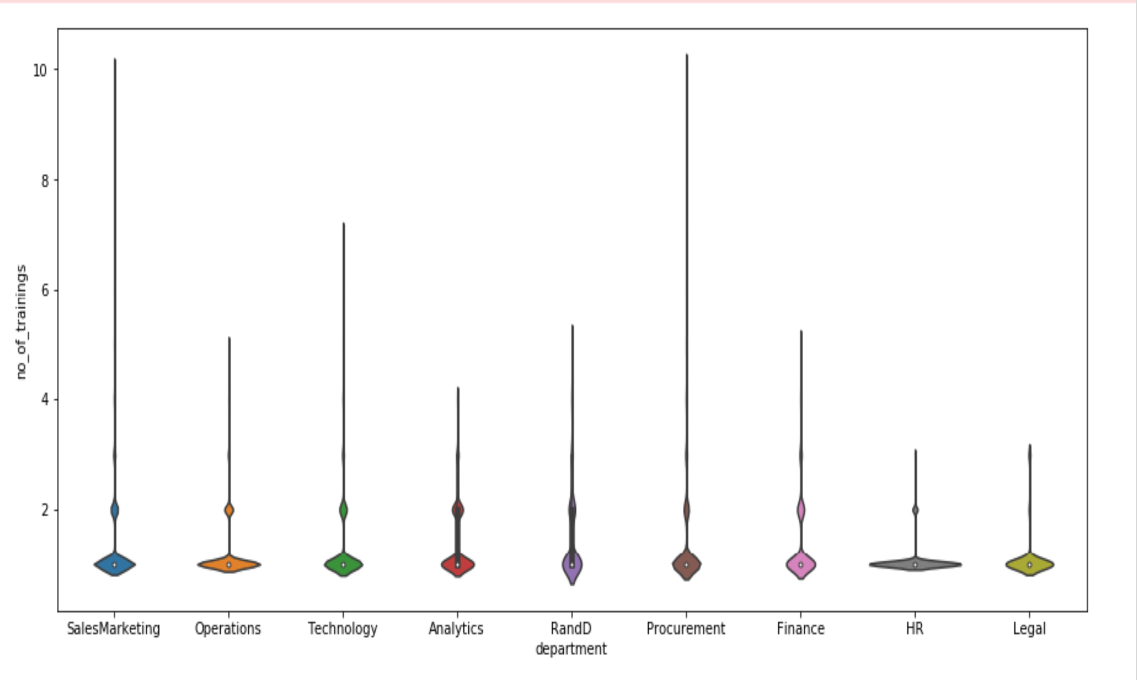
Insights:
1. Analytics and R&D dept scores highest in company training performance scores.
2. Sales & Marketing and HR dept who are huge in numbers scores the least.



Attribute x =[Department]   y =[No of trainings]

Insights:

1. Sales & Marketing, Procurement are good dept in terms of training as they are huge in numbers and they have done the maximum trainings.
2. HR is the worst dept in terms of training attended.
3. Analytics and R&D although less in number but have done good amount of 2 trainings than compared to every other who have done mostly just 1.

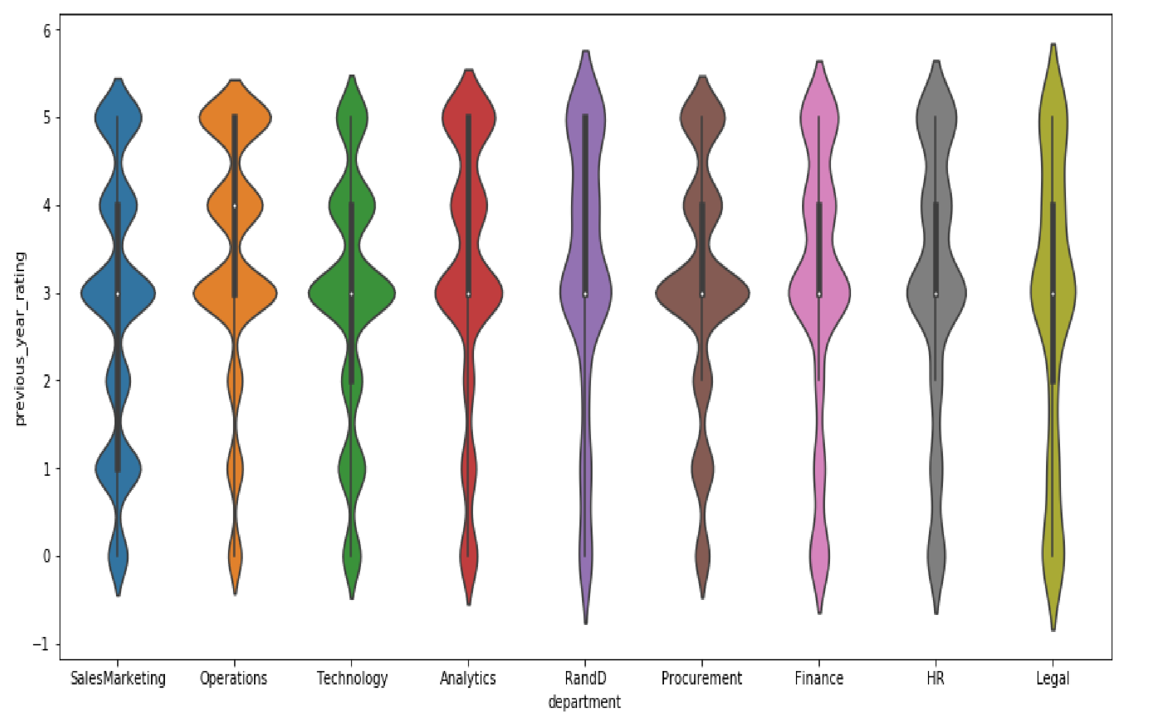


Attribute : x =[Department] y=[Previous Year Rating]

Insights:

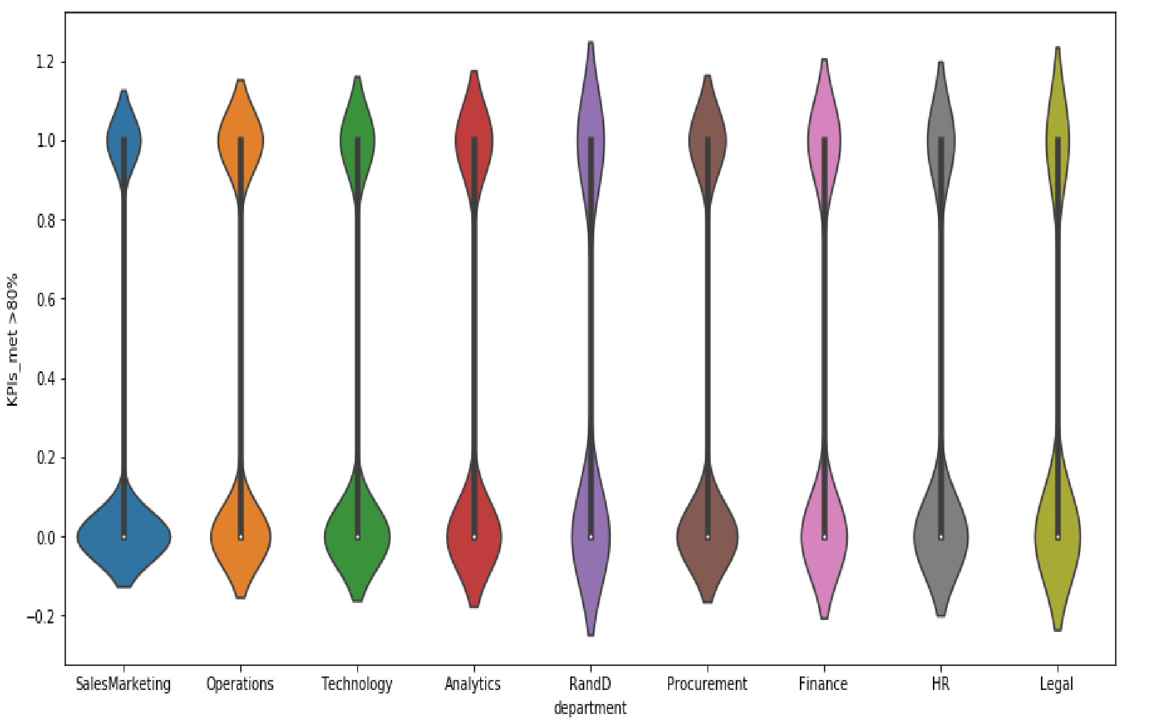
1. Almost every dept has on an average previous year rating of 3.

2. Analytics, operations and R&D dept has good amount of 5 rating.

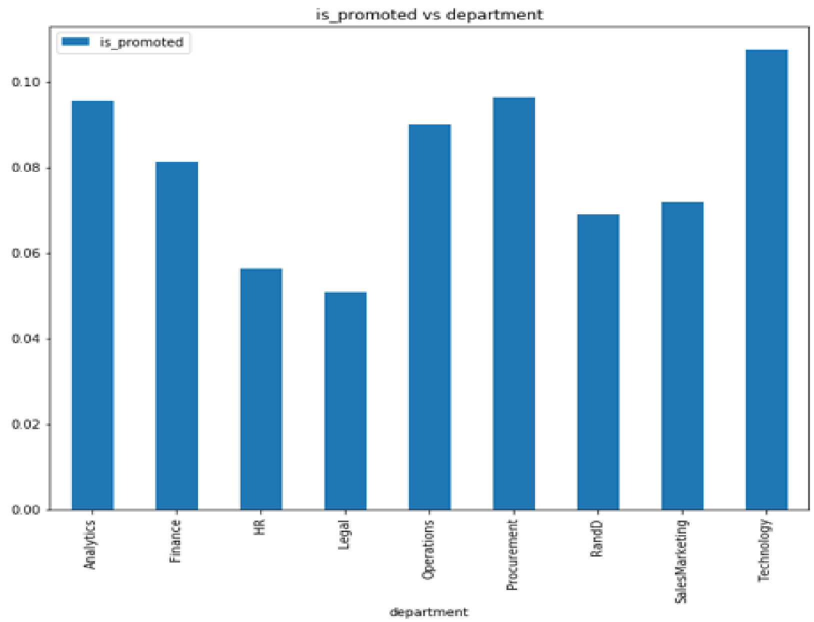


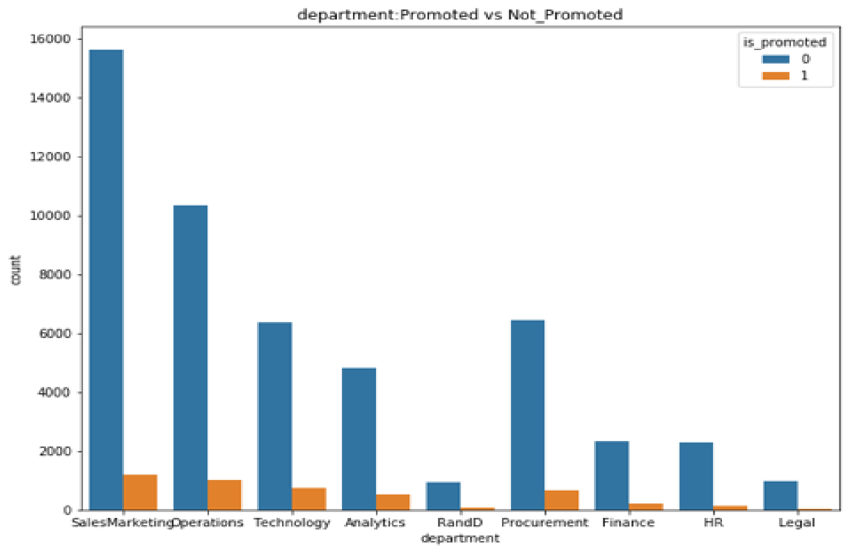
Attribute : x =[Department] y =[KPIs met >80%]

Insights:
1. Every department has good mix blend of KPI scores.
2. Analytics can be seen much better among all of them.



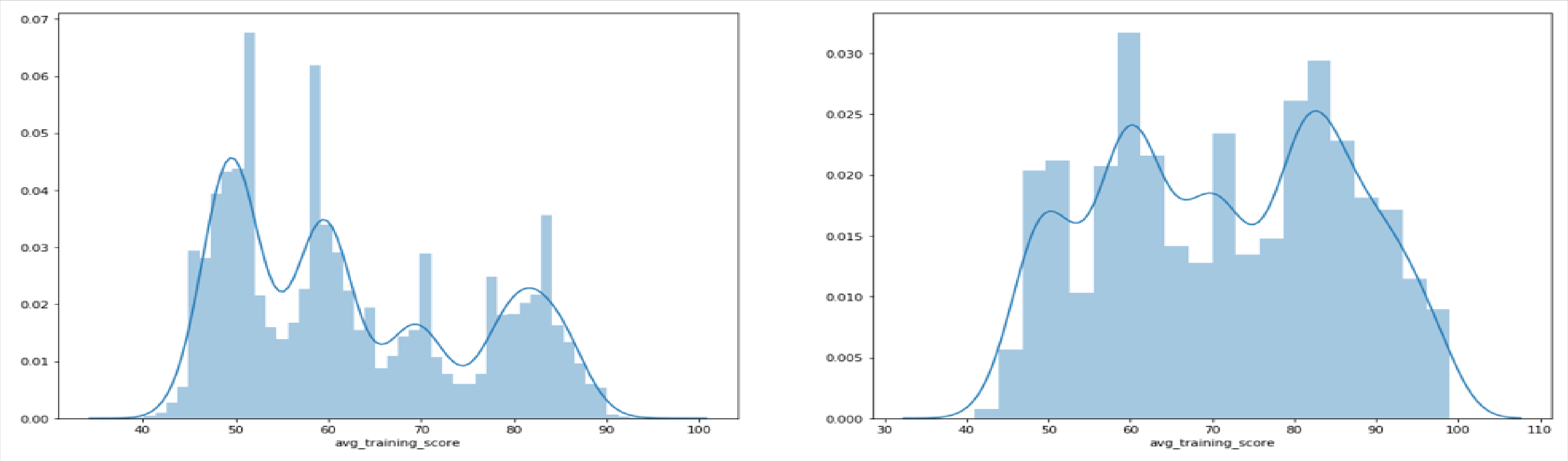
Department VS Promotion





Insights : From Sales & Marketing department, we have seen maximum employees get promoted.

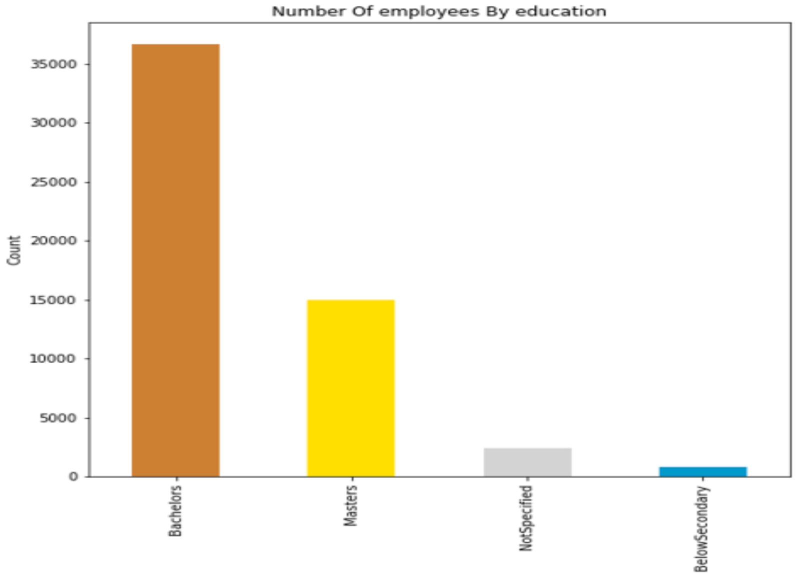
Average Training Score VS Promotion

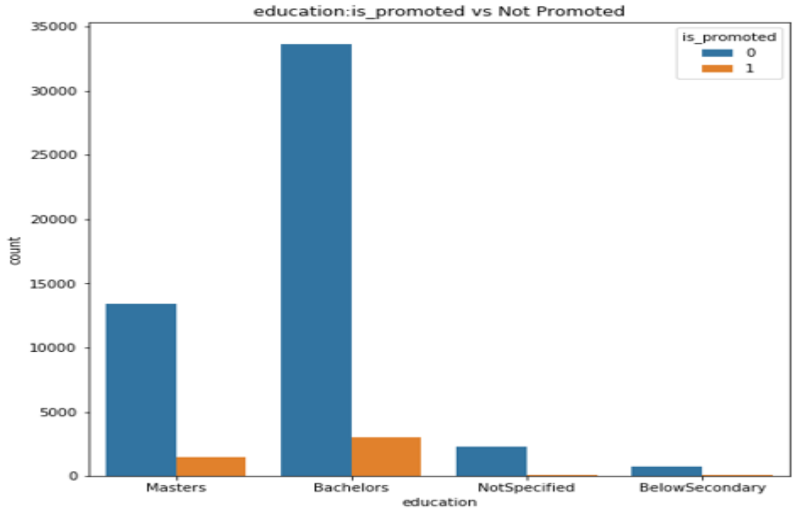


Insights :

1. Highest Average Training Score was: 99
2. Lowest Average Training Score was: 39
3. Average of all employee Average Training Score was: 63.38675010947307

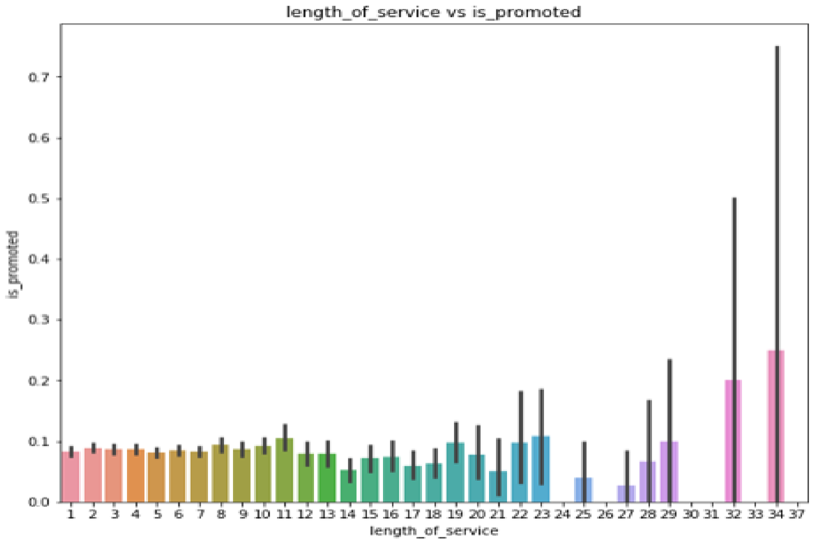
Education VS Promotion

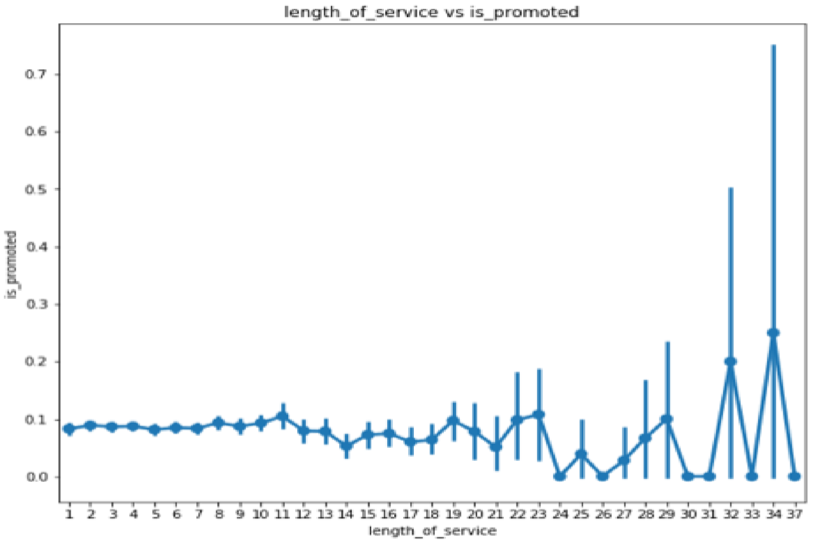




Insights : As we can see that employees who are Post Graduate were given more priority than any other education.
For Masters, % promoted is around 10% while for Bachelors, is around 8%

Length Of Service VS Promotion

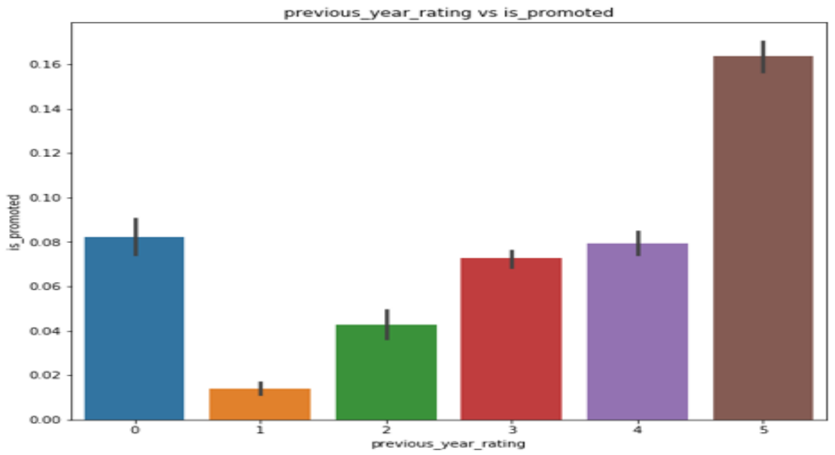


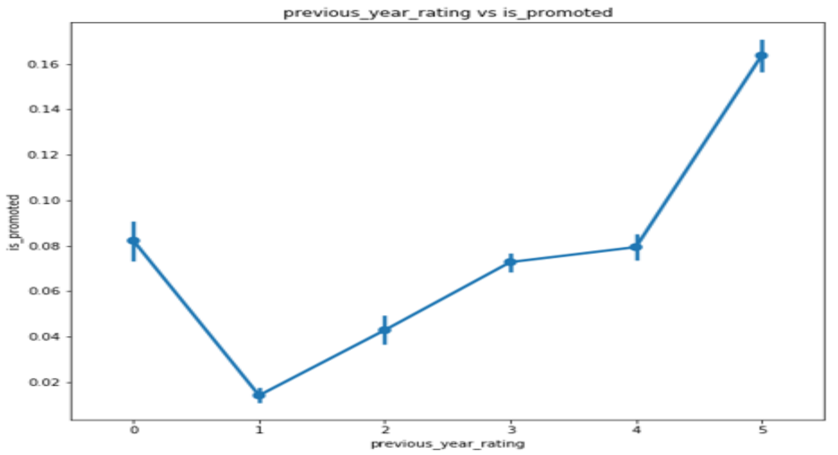


Insights :

1. Here too the results are quite similar. Employee with their more experience have greater chance of promotion.
2. Experience from 1 to 11 years there is constant rate of promotion.
3. It however increases after 26 years of experience as the number goes up.

Previous Year Rating VS Promotion



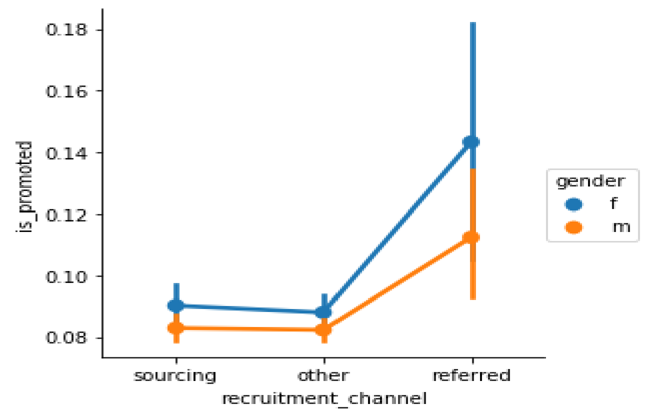


Insights :
We can clearly observed from the Bar plot and Factor Plot that if Employee Previous Rating is High more chances employee get promoted

Gender VS Promotion

Crosstab between : Recruitment_channel , Gender, Promotion.

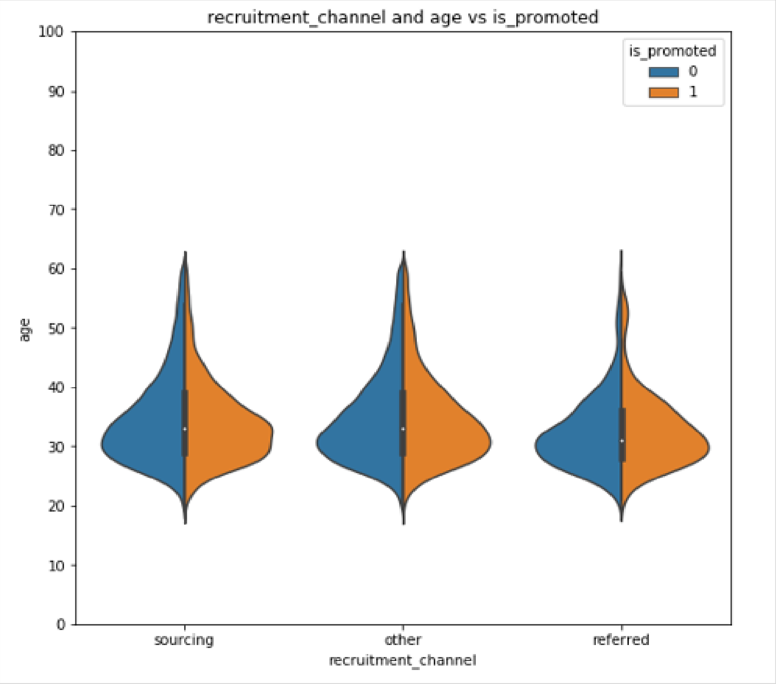
	recruitment_channel	other	referred	sourcing	All
gender	is_promoted				
f	0	8350	269	6226	14845
	1	805	45	617	1467
m	0	19540	735	15020	35295
	1	1751	93	1357	3201
All		30446	1142	23220	54808



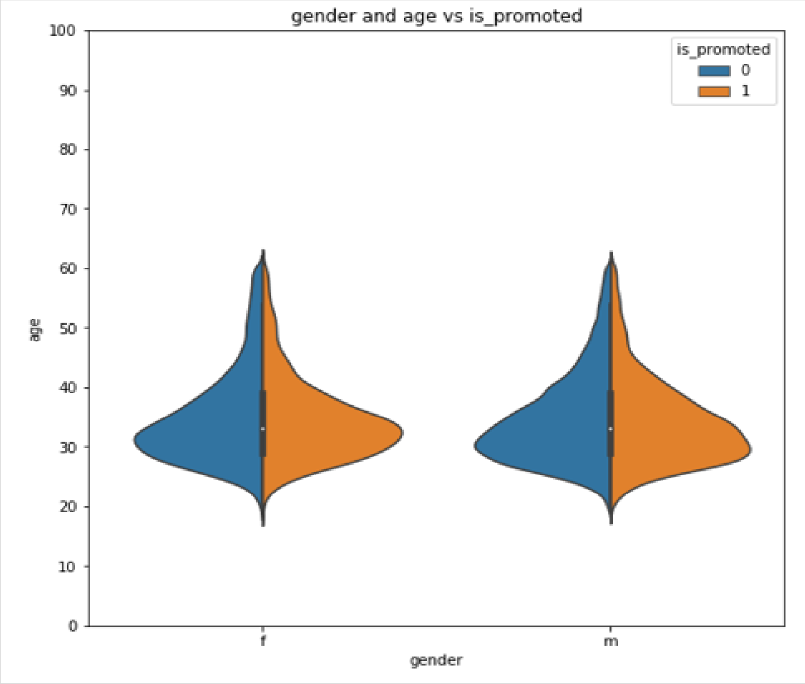
Insights : Looking at the Crosstab and the Factor Plot, we can easily infer that promotion for Women from Referred recruitment channel is about 14%.
It is evident that irrespective of recruitment channel, Women were given first priority while promotion. Even Men from any recruitment channel have a less promotion chance.

AGE VS Promotion

Recruitment_channel and Age VS Promotion



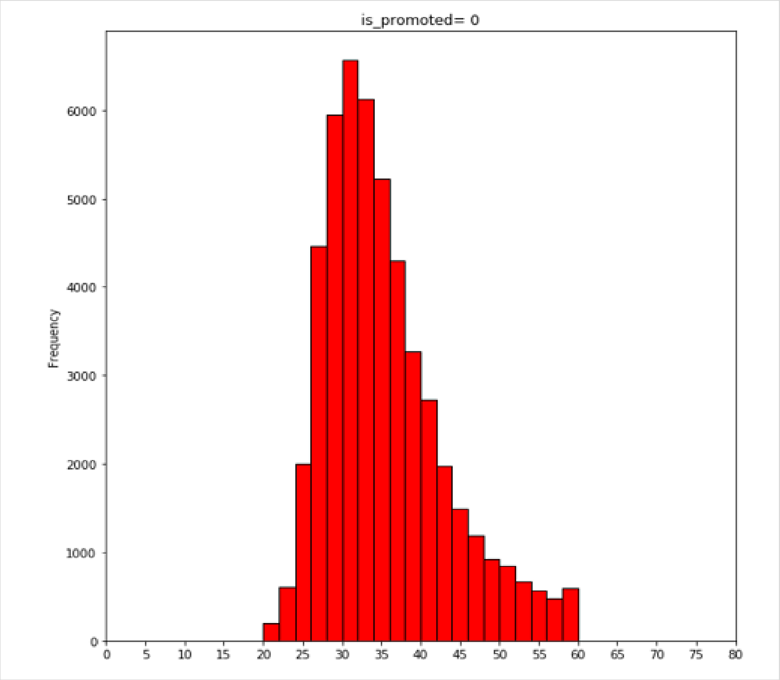
Gender and Age VS Promotion

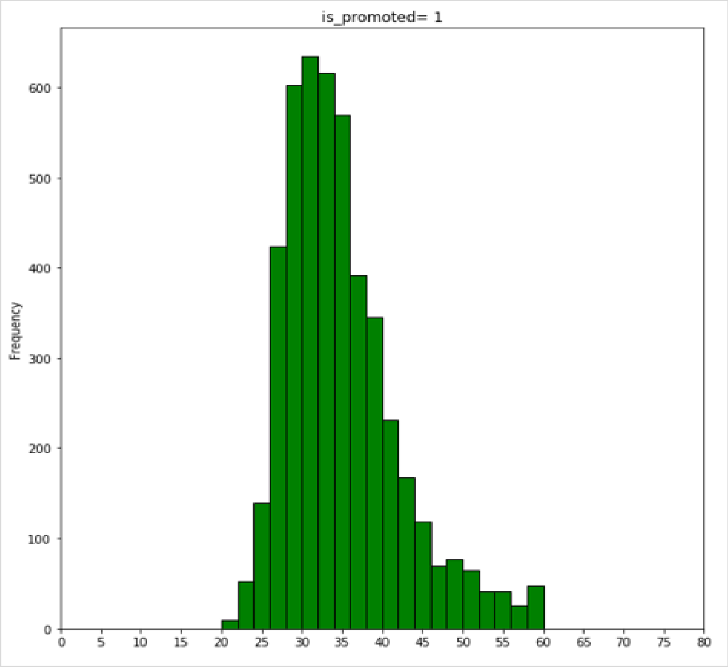


Age VS Promotion.

Insights:

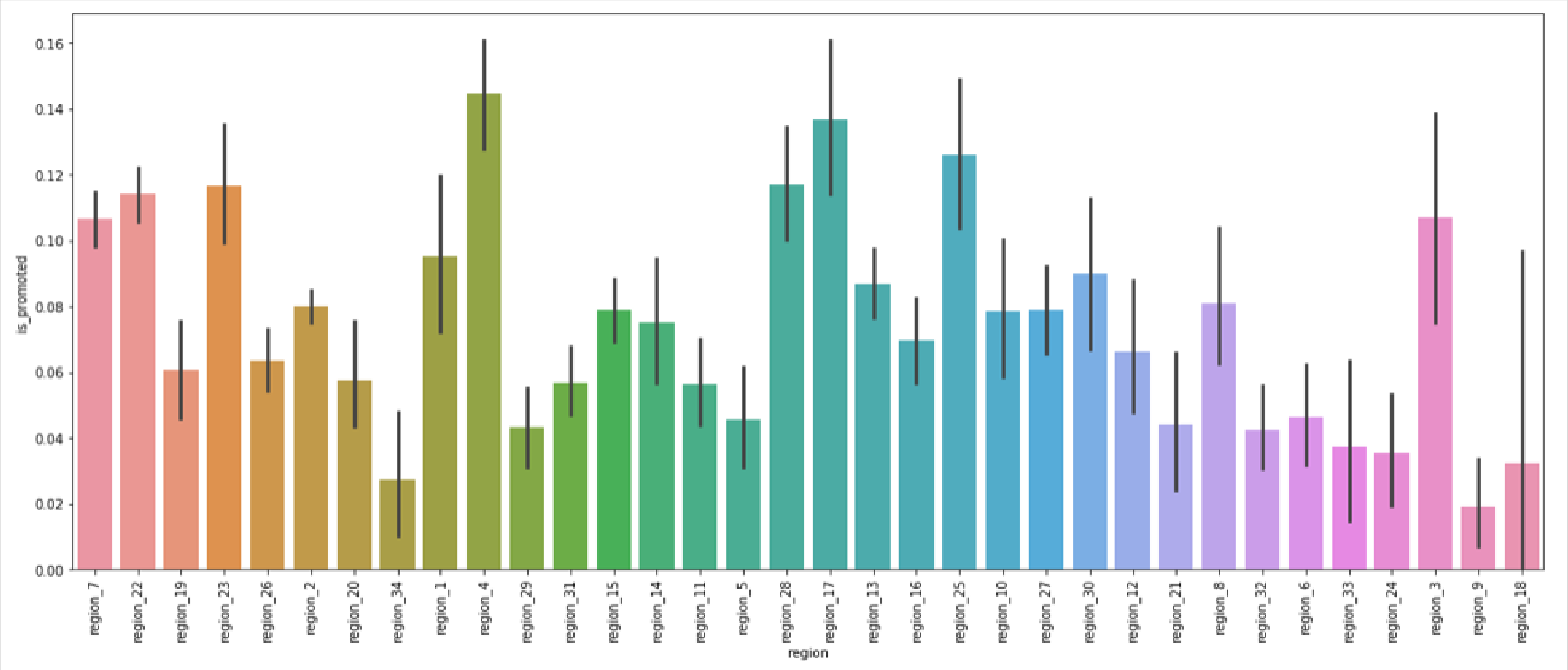
1. The oldest employee was promoted(60 years).
2. Maximum number of promotion were in the age group of 30-40



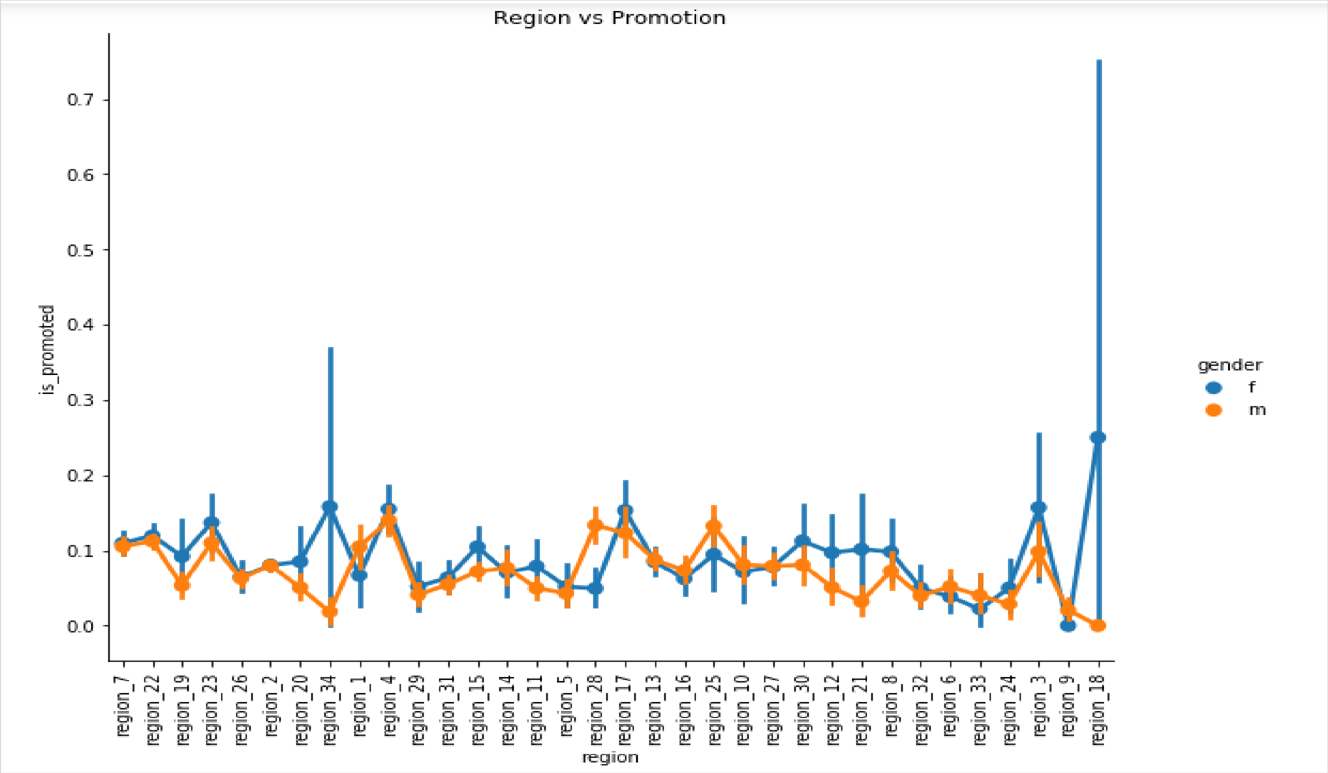


Oldest employee was of: 60 Years
Youngest employee was of: 20 Years
Average Age of employee: 34.8 Years

Region VS Promotion.



Region VS Promotion.



Insights:

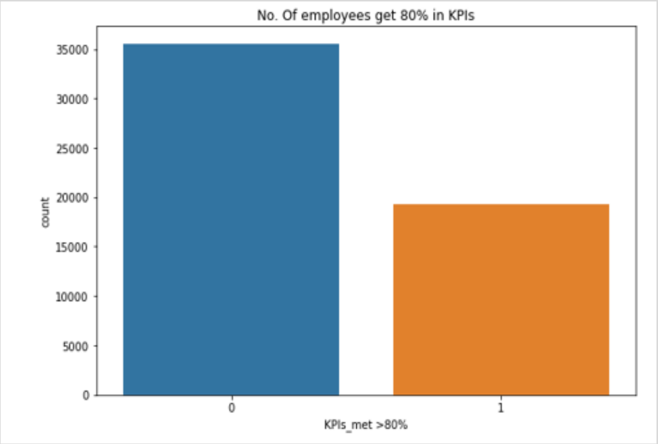
1. From Bar plot and Factor Plot shows that region 4 has greater impact on Promotion. As average of Male and Female highest on region4.
2. We can clearly observed that from many of the regions female have higher promotion rate than male.

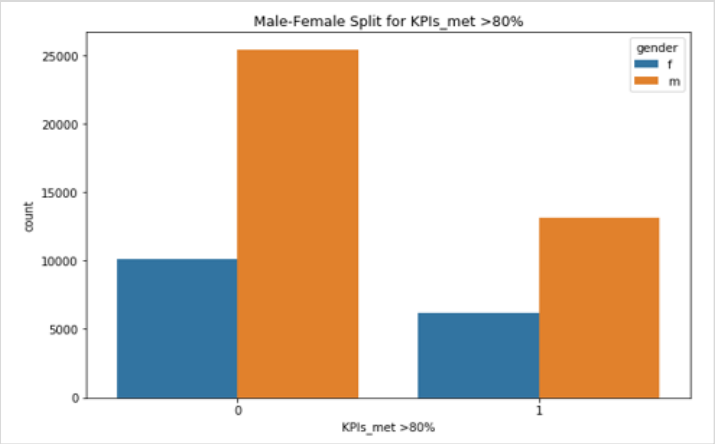
KPIs_met >80% VS Promotion

Crosstab Between KPIs_met >80%, Gender, Education VS promotion

	gender	f		m		All
	is_promoted	0	1	0	1	
KPIs_met >80%	education					
0	Bachelors	6469	272	16300	629	23670
	Below Secondary	166	9	295	13	483
	Masters	2763	147	6340	300	9550
	Not Specified	294	3	1484	33	1814
1	Bachelors	3455	658	7437	1449	12999
	Below Secondary	102	12	175	33	322
	Masters	1513	355	2838	669	5375
	Not Specified	83	11	426	75	595
All		14845	1467	35295	3201	54808

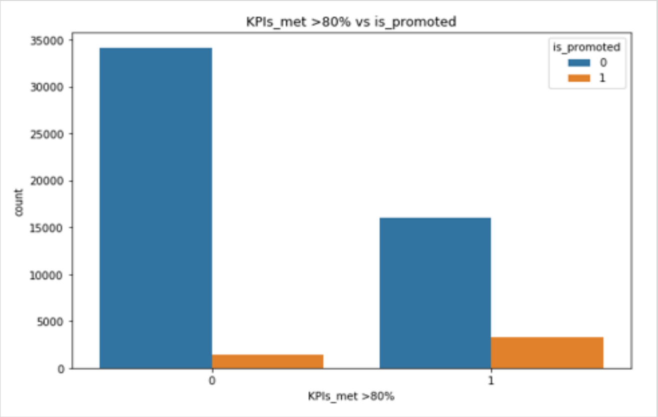
KPIs_met >80% VS Promotion

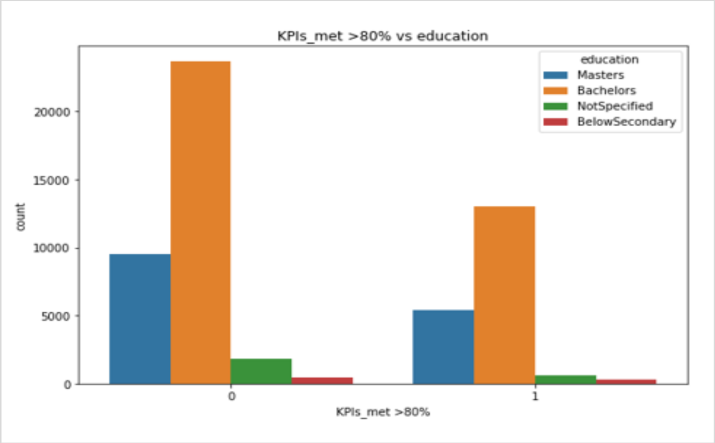




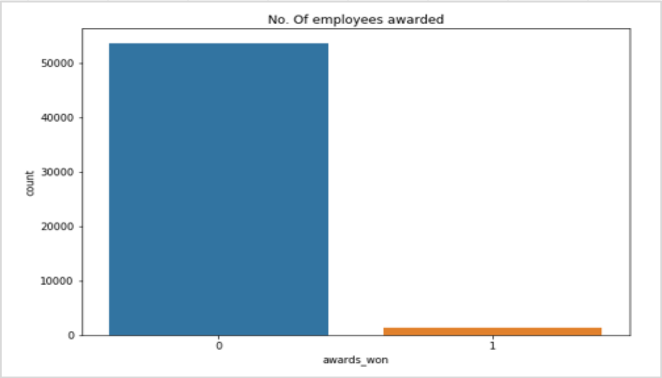
Insights :

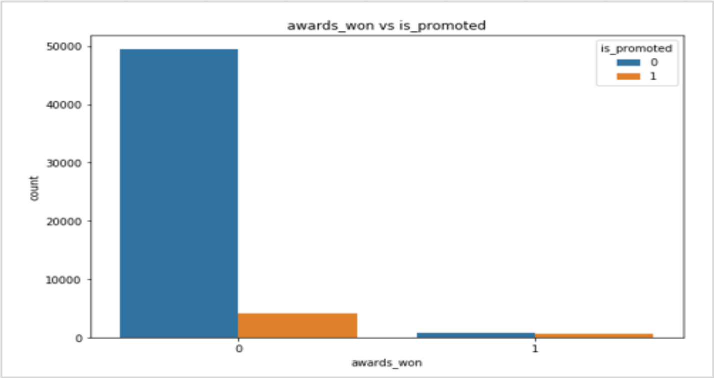
1. More male met the KPIs score i.e. greater than 80%
2. Count of Bachelors employees who scored more than 80% and get promoted is higher than other education degree. But % of Masters degree has the highest among all four degrees.





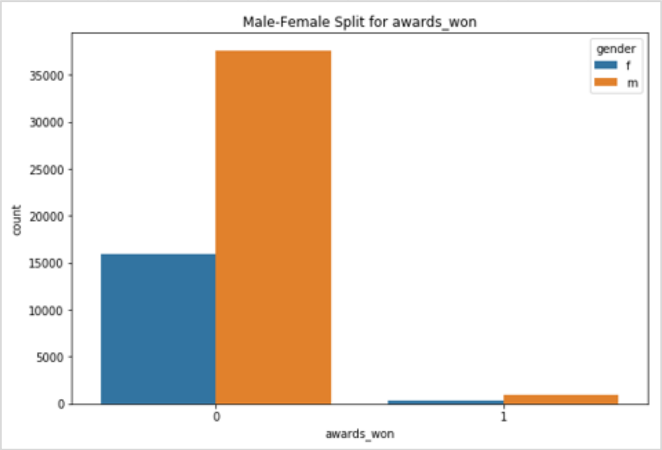
Awards Won VS Promotion

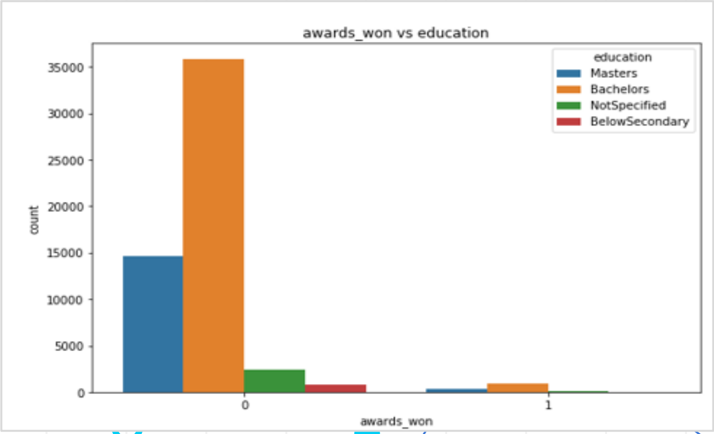




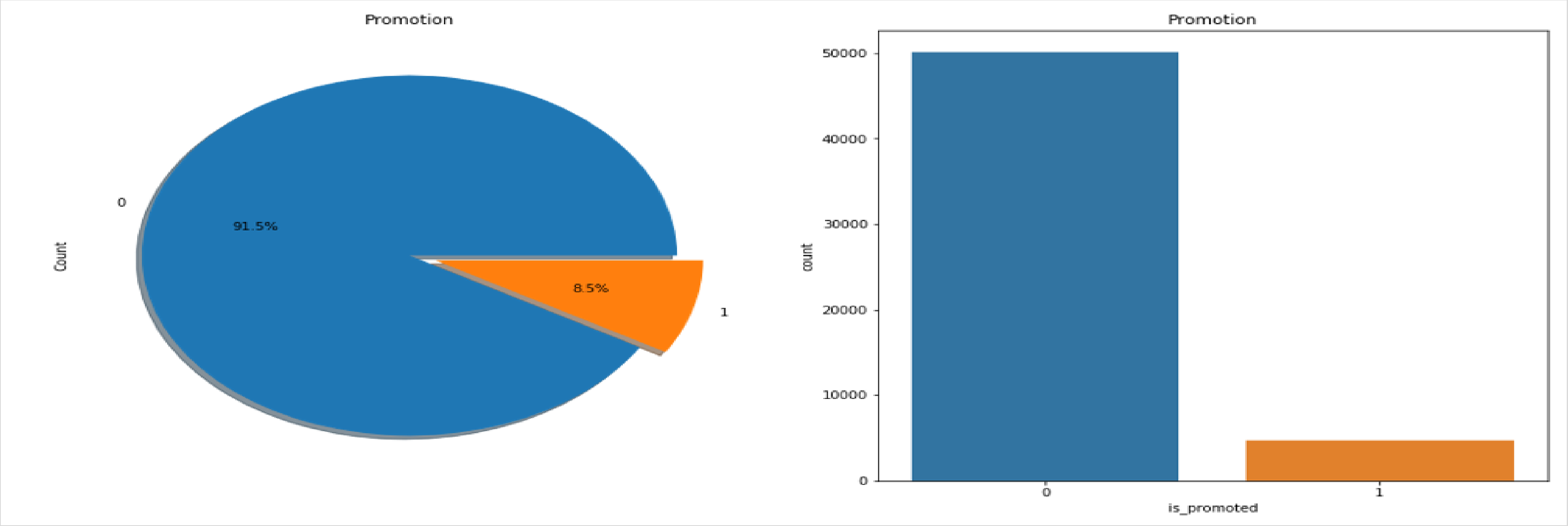
Insights :

1. Male employee have awarded more than female employee.
2. Bachelors employee have awarded more than other education.





Target Variable : Promotion



Education Backgrounds of Departments:

```
pd.crosstab(employee['education'],employee['department'])
```

department education	Analytics	Finance	HR	Legal	Operations	Procurement	RandD	SalesMarketing	Technology
Bachelors	3978	1895	1525	814	7781	4393	542	11099	4642
BelowSecondary	0	106	128	65	176	129	0	0	201
Masters	1037	499	733	156	3165	2544	429	4166	2196
NotSpecified	337	36	32	4	226	72	28	1575	99

Insights:

1. Shocking that dept like finance, HR, legal, operations, procurement and technology have education background below secondary.
2. R&D dept has bag of bachelors and masters.
3. Every dept has more number of bachelors.

Recruitment Channels of Departments:

```
pd.crosstab(employee['recruitment_channel'],employee['department'])
```

department recruitment_channel	Analytics	Finance	HR	Legal	Operations	Procurement	RandD	SalesMarketing	Technology
other	2973	1463	1380	590	6279	4002	555	9290	3914
referred	83	5	103	14	238	79	19	259	342
sourcing	2296	1068	935	435	4831	3057	425	7291	2882

Insights:
1. Technology has most referrals than any other dept.
2. Analytics and R&D has mixed blend of others and sourcing as recruitment channels.
3. The recruitment channel others is the most preferred for recruitment.

Gender Ratio of Departments:

```
pd.crosstab(employee['gender'],employee['department'])
```

department	Analytics	Finance	HR	Legal	Operations	Procurement	RandD	SalesMarketing	Technology
gender									
f	513	681	1006	149	4677	3287	57	3154	2788
m	4839	1855	1412	890	6671	3851	942	13686	4350

Insights:
1. Data is more tilted towards male by huge numbers.
2. Procurement department has the best gender equality(even when female crowd is less in huge numbers) and then HR.
3. The worst is sales & marketing dept., followed by analytics and R&D.

Correlation Between The Features

Insights :
Interpreting The Heatmap:

1. Only the **numeric features** are compared as it is obvious that we cannot correlate between alphabets or strings.

- **POSITIVE CORRELATION**: If an increase in feature A leads to increase in feature B, then they are positively correlated. A value 1 means perfect positive correlation.
- **NEGATIVE CORRELATION**: If an increase in feature A leads to decrease in feature B, then they are negatively correlated. A value -1 means perfect negative correlation.

2. Now lets say that two features are highly or perfectly correlated, so the increase in one leads to increase in the other. This means that both the features are containing highly similar information and there is very little or no variance in information. This is known as Multicollinearity as both of them contains almost the same information.

3. Now from the above heatmap, we can see that the features are not much correlated. The highest correlation is between Length of Service and Age i.e. 0.66. So we can carry on with all features

Modelling On Dataset 4

BaseLine Estimator for this model is

```
employee.is_promoted[employee.is_promoted==0].shape
```

```
(50140,)
```

```
employee.is_promoted[employee.is_promoted==1].shape
```

```
(4668,)
```

```
baseline = 50140/(50140+4668)
print(baseline)
```

BaseLine Estimator is 91.4%

Applying Dummies on the Dataset.

```
new_employee=pd.get_dummies(employee)
```

```
new_employee.columns
```

```
Index(['employee_id', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'KPIs_met >80%', 'awards_won',
       'avg_training_score', 'is_promoted', 'department_Analytics',
       'department_Finance', 'department_HR', 'department_Legal',
       'department_Operations', 'department_Procurement', 'department_RandD',
       'department_SalesMarketing', 'department_Technology', 'region_region_1',
       'region_region_10', 'region_region_11', 'region_region_12',
       'region_region_13', 'region_region_14', 'region_region_15',
       'region_region_16', 'region_region_17', 'region_region_18',
       'region_region_19', 'region_region_2', 'region_region_20',
       'region_region_21', 'region_region_22', 'region_region_23',
       'region_region_24', 'region_region_25', 'region_region_26',
       'region_region_27', 'region_region_28', 'region_region_29',
       'region_region_3', 'region_region_30', 'region_region_31',
       'region_region_32', 'region_region_33', 'region_region_34',
       'region_region_4', 'region_region_5', 'region_region_6',
       'region_region_7', 'region_region_8', 'region_region_9',
       'education_Bachelors', 'education_BelowSecondary', 'education_Masters',
       'education_NotSpecified', 'gender_f', 'gender_m',
       'recruitment_channel_other', 'recruitment_channel_referred',
       'recruitment_channel_sourcing'],
      dtype='object')
```

Applying Train Test Split on the independent variables and dependent variables

```
from sklearn.model_selection import train_test_split
```

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

```
x = new_employee.drop(['employee_id','is_promoted'],axis=1)
```

```
y = new_employee['is_promoted']
```

```
x.columns
```

```
Index(['no_of_trainings', 'age', 'previous_year_rating', 'length_of_service',
       'KPIs_met >80%', 'awards_won', 'avg_training_score',
       'department_Analytics', 'department_Finance', 'department_HR',
       'department_Legal', 'department_Operations', 'department_Procurement',
       'department_RandD', 'department_SalesMarketing',
       'department_Technology', 'region_region_1', 'region_region_10',
       'region_region_11', 'region_region_12', 'region_region_13',
       'region_region_14', 'region_region_15', 'region_region_16',
       'region_region_17', 'region_region_18', 'region_region_19',
       'region_region_2', 'region_region_20', 'region_region_21',
       'region_region_22', 'region_region_23', 'region_region_24',
       'region_region_25', 'region_region_26', 'region_region_27',
       'region_region_28', 'region_region_29', 'region_region_3',
       'region_region_30', 'region_region_31', 'region_region_32',
       'region_region_33', 'region_region_34', 'region_region_4',
       'region_region_5', 'region_region_6', 'region_region_7',
       'region_region_8', 'region_region_9', 'education_Bachelors',
       'education_BelowSecondary', 'education_Masters',
       'education_NotSpecified', 'gender_f', 'gender_m',
       'recruitment_channel_other', 'recruitment_channel_referred',
       'recruitment_channel_sourcing'],
      dtype='object')
```

```
xtrain, xtest , ytrain , ytest = train_test_split(x,y,test_size = 0.3,random_state = 2)
```

Implement different Classifiers : Logistic Regression, Gaussian Naive Bayes, KNN, Decision tree, Random forest

```
model_1 = LogisticRegression()
```

```
model_1.fit(xtrain,ytrain)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
from sklearn.metrics import accuracy_score,r2_score
from sklearn import metrics
```

Inference : Here we can see that our model accuracy is 93% which is better than baseline estimator

```
ypred = model_1.predict(xtest)
metrics.accuracy_score(ytest,ypred)
```

```
0.9315818281335523
```

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

```
array([[14975,    60],
       [ 1065,   343]], dtype=int64)
```

```
cr = metrics.classification_report(ytest,ypred)
print(cr)
```

```
             precision    recall  f1-score   support

          0       0.93      1.00      0.96     15035
          1       0.85      0.24      0.38      1408

avg / total       0.93      0.93      0.91     16443
```

Scaling the Dataset

```python
from sklearn import preprocessing
```

```python
employee_std = new_employee.drop(['is_promoted'],axis=1)
```

```python
a_scaled = preprocessing.scale(employee_std)
b = pd.DataFrame(a_scaled,columns= employee_std.columns)
```

```python
Xtrain,Xtest,Ytrain,Ytest = train_test_split(X,Y,test_size = 0.3,random_state =2)
```

```python
modelimp = LogisticRegression()
modelimp.fit(Xtrain,Ytrain)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```python
Y_Pred = modelimp.predict(Xtest)
```

```python
Y_Pred
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```python
CR = metrics.classification_report(Ytest,Y_Pred)
print(CR)
```

	precision	recall	f1-score	support
0	0.94	0.99	0.96	15035
1	0.81	0.26	0.40	1408
avg / total	0.92	0.93	0.92	16443

Now we will consider best model to our data not only with the basis of Accuracy. We will check the Bias and Variance Error in the model

```
models = []
models.append(('DecisionTree', Dt_model))
models.append(('RandomForest', Rf_model))
models.append(('Base_KNN',knn))
models.append(('Base_LR',modelimp))
models.append(('Base_NB',NB))
```

```
from sklearn import model_selection
results = []
names = []
sho3 = {}
scoring = 'recall'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10,random_state=2)
    cv_results = model_selection.cross_val_score(model, x, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, np.mean(1-cv_results), cv_results.var())
    print(msg)
    sho3.update({name+' Bias Error':round(np.mean(1-cv_results),6),name+' Variance Error':round(cv_results.var(),6)})
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
sorted([(value,key) for (key,value) in sho3.items()])
```

Results

```
[(0.000142, 'RandomForest Variance Error'),
 (0.000163, 'Base_KNN Variance Error'),
 (0.000233, 'DecisionTree Variance Error'),
 (0.000277, 'Base_LR Variance Error'),
 (0.001013, 'Base_NB Variance Error'),
 (0.241024, 'Base_NB Bias Error'),
 (0.673646, 'DecisionTree Bias Error'),
 (0.751702, 'Base_LR Bias Error'),
 (0.858807, 'Base_KNN Bias Error'),
 (0.89591, 'RandomForest Bias Error')]
```

Decision Tree is performing well in the Employee Performance Prediction so predicting test data using Decision tree as trade off Bias and Variance Error is less than others

Applying Label Encoding On Nominal Variables

Input :

```
nominal = ['department','region','gender','recruitment_channel']
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
# department
encode1 = pd.DataFrame(le.fit_transform(employee['department']))
encode1.head()
encode1.columns = ['department']
```

```
# region
encode2 = pd.DataFrame(le.fit_transform(employee['region']))
encode2.columns = ['region']
# gender
encode3 = pd.DataFrame(le.fit_transform(employee['gender']))
encode3.columns = ['gender']
# recruitment channel
encode4 = pd.DataFrame(le.fit_transform(employee['recruitment_channel']))
encode4.columns = ['recruitment_channel']
# dummy variables for education variable
dum1 = pd.get_dummies(employee['education'])
#dum1 = dum1.drop(['recruitment_channel'],1)
# concatenate
end = pd.concat([encode1,encode2,encode3,encode4,dum1],1)
end.head()
```

Output :

	department	region	gender	recruitment_channel	Bachelors	BelowSecondary	Masters	NotSpecified
0	7	31	0	2	0	0	1	0
1	4	14	1	0	1	0	0	0
2	7	10	1	2	1	0	0	0
3	7	15	1	0	1	0	0	0
4	8	18	1	0	1	0	0	0

Applying concatenate on the Variables

Input :

```
train2 = employee.drop(['department','region','gender','recruitment_channel','education'],1)
train2 = pd.concat([end,train2],1)
train2.info()
```

Output :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 16 columns):
department               54808 non-null int32
region                   54808 non-null int32
gender                   54808 non-null int32
recruitment_channel      54808 non-null int32
Bachelors                54808 non-null uint8
BelowSecondary           54808 non-null uint8
Masters                  54808 non-null uint8
NotSpecified             54808 non-null uint8
no_of_trainings          54808 non-null int64
age                      54808 non-null int64
previous_year_rating     54808 non-null int64
length_of_service        54808 non-null int64
KPIs_met >80%            54808 non-null int64
awards_won               54808 non-null int64
avg_training_score       54808 non-null int64
is_promoted              54808 non-null int64
dtypes: int32(4), int64(8), uint8(4)
memory usage: 4.4 MB
```

Model Preparation

```
X = train2.drop(['is_promoted'],1)
y = train2['is_promoted']
```

Applying SMOTE Technique

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

Input :

```
print('Shape of X: {}'.format(X.shape))
print('Shape of y: {}'.format(y.shape))
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)

print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

Output:

```
Shape of X: (54808, 15)
Shape of y: (54808,)
```

```
Number transactions X_train dataset:  (38365, 15)
Number transactions y_train dataset:  (38365,)
Number transactions X_test dataset:  (16443, 15)
Number transactions y_test dataset:  (16443,)
```

Before and After Applying Oversampling

```
Before OverSampling, counts of label in train '1': 3260
Before OverSampling, counts of label in train '0': 35105

Before OverSampling, counts of label in test '1': 1408
Before OverSampling, counts of label i test '0': 15035

After OverSampling, the shape of train_X: (70210, 15)
After OverSampling, the shape of train_y: (70210,)

After OverSampling, counts of label in train '1': 35105
After OverSampling, counts of label in train '0': 35105

After OverSampling, the shape of test_X: (30070, 15)
After OverSampling, the shape of test_y: (30070,)

After OverSampling, counts of label in test '1': 15035
After OverSampling, counts of label in test '0': 15035
```

Random Forest Classifier on Training Accuracy

Library to be exported

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

INPUT :

```
est = range(100,200,20)
train_f1 = []
for i in est:
    rf = RandomForestClassifier(n_estimators = i,random_state = 3)
    rf.fit(X_train_res,y_train_res)
    y_pred = rf.predict(X_train_res)
    print('Training Accuracy Score:',metrics.accuracy_score(y_train_res,y_pred))
    print('Training F1 Score is:',metrics.f1_score(y_train_res,y_pred))
    train_f1.append(metrics.f1_score(y_train_res,y_pred))
```

OUTPUT :

```
Training Accuracy Score: 0.9996296823814271
Training F1 Score is: 0.9996297983825038
Training Accuracy Score: 0.9996296823814271
Training F1 Score is: 0.9996297983825038
Training Accuracy Score: 0.9996296823814271
Training F1 Score is: 0.9996297983825038
Training Accuracy Score: 0.9996296823814271
Training F1 Score is: 0.999629772968048
Training Accuracy Score: 0.9996296823814271
Training F1 Score is: 0.999629772968048
```

Random Forest Classifier on Testing Accuracy

INPUT :

```
est = range(100,200,20)
test_f1 = []
for i in est:
    rf = RandomForestClassifier(n_estimators = i,random_state = 5)
    rf.fit(X_train_res,y_train_res)
    y_pred = rf.predict(X_test_res)
    print('Testing Accuracy Score:',metrics.accuracy_score(y_test_res,y_pred))
    print('Testing F1 Score is:',metrics.f1_score(y_test_res,y_pred))
    test_f1.append(metrics.f1_score(y_test_res,y_pred))
```

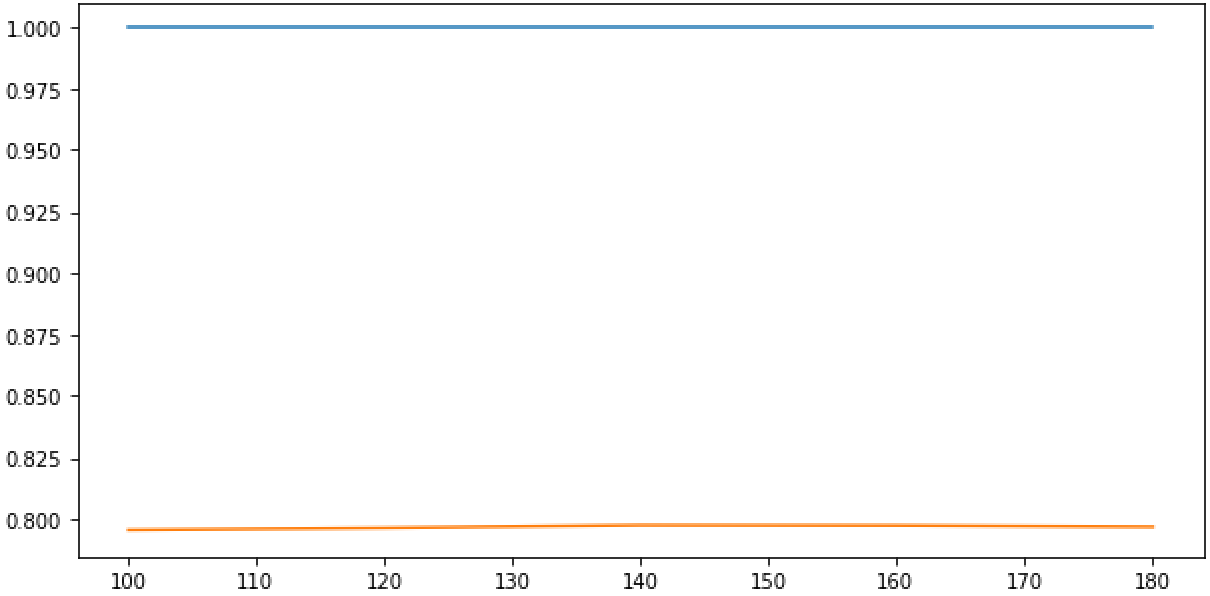
OUTPUT :

```
Testing Accuracy Score: 0.8202527435982707
Testing F1 Score is: 0.7954278793384051
Testing Accuracy Score: 0.8209511140671766
Testing F1 Score is: 0.7962458371177717
Testing Accuracy Score: 0.8218490189557699
Testing F1 Score is: 0.797397980409213
Testing Accuracy Score: 0.8217492517459262
Testing F1 Score is: 0.7972768532526475
Testing Accuracy Score: 0.8213501829065514
Testing F1 Score is: 0.7966845810309 59
```

Plot for F1_Score

Comment : Seeing the accuracy and F1 scores, i am choosing n_estimators = 160.

```
neig = np.arange(100,200,20)
#Plot
plt.figure(figsize=[18,8])
plt.plot(neig,train_f1, label = 'Training F1 Score')
plt.plot(neig,test_f1, label = 'Testing F1 Score')
plt.show()
```



For other parameters like max_features, max_depth and criterion, we will found out by using grid search.

Applying GridSearchCV

```
from sklearn.model_selection import GridSearchCV

rf_params = {'max_depth': [4,6,8,10],'criterion': ['gini','entropy'],'max_features': ['auto','sqrt','log2']}
rf1 = RandomForestClassifier(n_estimators = 160,random_state = 5)
rf_grid = GridSearchCV(rf1,rf_params,cv=5, n_jobs=-1, verbose= 1)
rf_grid.fit(X_train_res, y_train_res)
```

```
print(rf_grid.best_estimator_)
print(rf_grid.best_params_)
print(rf_grid.best_score_)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=160,
                       n_jobs=None, oob_score=False, random_state=5, verbose=0,
                       warm_start=False)
{'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto'}
0.8469306366614442
```

Comment :
Finally our Random forest model parameters are decided and now applying on the test data:
max_depth = 10, max_features = auto, and criteria Gini .

Without scaled, label encoded, hypertuned model :

```
rf1 = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=160, n_jobs=None,
            oob_score=False, random_state=5, verbose=0, warm_start=False)
rf1.fit(X_train_res,y_train_res)
y_pred1 = rf1.predict(X_test_res)
print('F1 Score is:',metrics.f1_score(y_test_res,y_pred1))
```

```
F1 Score is: 0.8257740806760415
```

Now trying to scale the variables:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_res)
X_test_scaled = scaler.fit_transform(X_test_res)

X_train_scaled = pd.DataFrame(X_train_scaled)
X_train_scaled.columns = X_train.columns

X_test_scaled = pd.DataFrame(X_test_scaled)
X_test_scaled.columns = X_test.columns
```

Input :

```
rf2 = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=160, n_jobs=None,
            oob_score=False, random_state=5, verbose=0, warm_start=False)
rf2.fit(X_train_scaled,y_train_res)
y_pred2 = rf2.predict(X_test_scaled)
print('F1 Score is:',metrics.f1_score(y_test_res,y_pred2))
```

Output :

```
F1 Score is: 0.8258243390434696
```

Insights:
Scaling has not significant effect on accuracy and f1 score of the random forest.

THANKS!

Any questions?