Predicting Promotion Rates through Logistic Regression Modeling

In [86]:

```
## import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import sklearn

## library versions
print('pandas version:', pd.__version__)
print('numpy version:', np.__version__)
print('seaborn version:', sns.__version__)
print('sklearn version:', sklearn.__version__)
```

```
pandas version: 1.0.3
numpy version: 1.18.1
seaborn version: 0.11.2
sklearn version: 0.24.2
```

In [87]:

```
## ignore warnings
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')

## print option
pd.set_option('display.max_columns', None)
```

In [151]:

```
## load dataset
df = pd.read_csv("train.csv")
df.head()
```

Out[151]:

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

In [152]:

```
## scrub df
df.replace([np.inf, -np.inf], np.nan, inplace=True)
```

In [153]:

```
df = df.dropna()
```

EDA

1. Employee ID is PII and not necessary for the model

2. Demographic data includes gender, age, and education
3. Work-related data make up the remainder of the fields

In [114]:
```
## df dimensions
df.shape
```

Out[114]:
```
(48660, 13)
```

In [115]:
```
## datatypes
df.dtypes
```

Out[115]:
```
employee_id               int64
department               object
region                   object
education                object
gender                   object
recruitment_channel      object
no_of_trainings           int64
age                       int64
previous_year_rating    float64
length_of_service         int64
awards_won?               int64
avg_training_score        int64
is_promoted               int64
dtype: object
```

In [94]:
```
## summary stats
df.describe()
```

Out[94]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	av(
count	48660.000000	48660.000000	48660.000000	48660.000000	48660.00000	48660.00000	
mean	39169.271681	1.251993	35.589437	3.337526	6.31157	0.02314	
std	22630.461554	0.604994	7.534571	1.257922	4.20476	0.15035	
min	1.000000	1.000000	20.000000	1.000000	1.00000	0.00000	
25%	19563.500000	1.000000	30.000000	3.000000	3.00000	0.00000	
50%	39154.000000	1.000000	34.000000	3.000000	5.00000	0.00000	
75%	58788.250000	1.000000	39.000000	4.000000	8.00000	0.00000	
max	78298.000000	10.000000	60.000000	5.000000	37.00000	1.00000	

In [95]:
```
## summary stats - non-numerical
df.describe(include = ['O'])
```

Out[95]:

	department	region	education	gender	recruitment_channel
count	48660	48660	48660	48660	48660
unique	9	34	3	2	3
top	Sales & Marketing	region_2	Bachelor's	m	other
freq	14239	10811	33404	33852	27017

Numerical Visualizations Observations

Length of Service

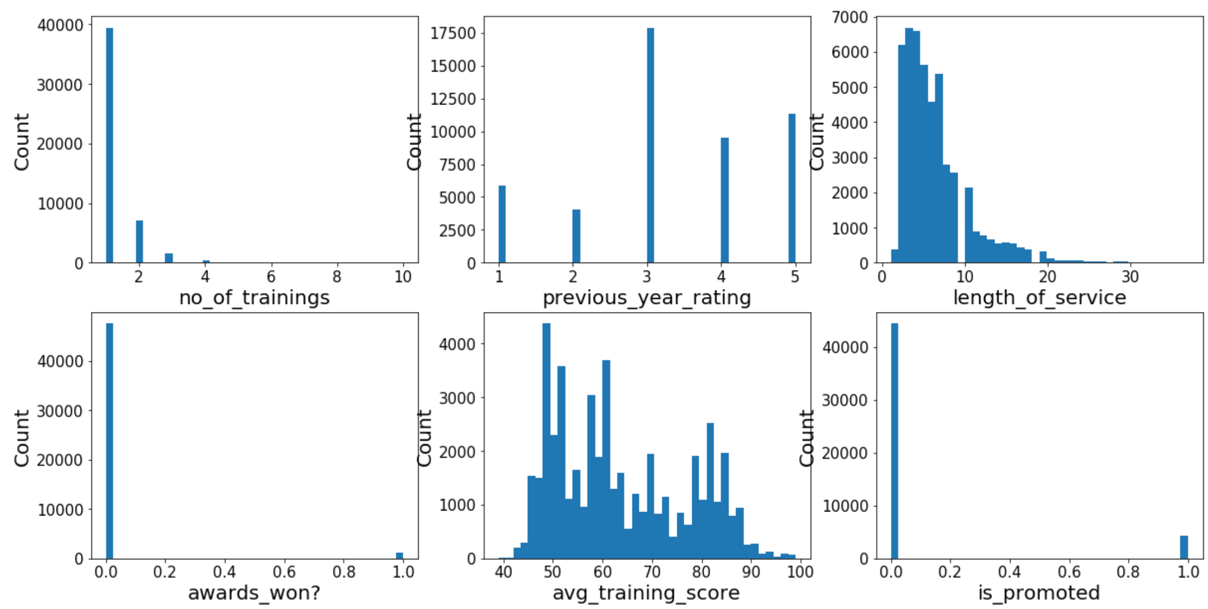
1. Mean(5.8) > median(5.0)
2. Length of service ranges from 1 yr to 37 yrs
3. The length of service of most employees is centered between 1 and 6 yrs

Average training score

1. Mean(63.38) > median(60.00)
2. Average training score ranges from 39 and 99

In [13]:

```
## numerical data histograms

plt.rcParams['figure.figsize'] = (20, 10)
fig, axes = plt.subplots(nrows = 2, ncols = 3)
num_features = ['no_of_trainings', 'previous_year_rating', 'length_of_service', 'awards
xaxes = num_features
yaxes = ['Count', 'Count', 'Count', 'Count','Count', 'Count', 'Count']
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(df[num_features[idx]].dropna(), bins=40)
    ax.set_xlabel(xaxes[idx], fontsize=20)
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)
plt.show()
```

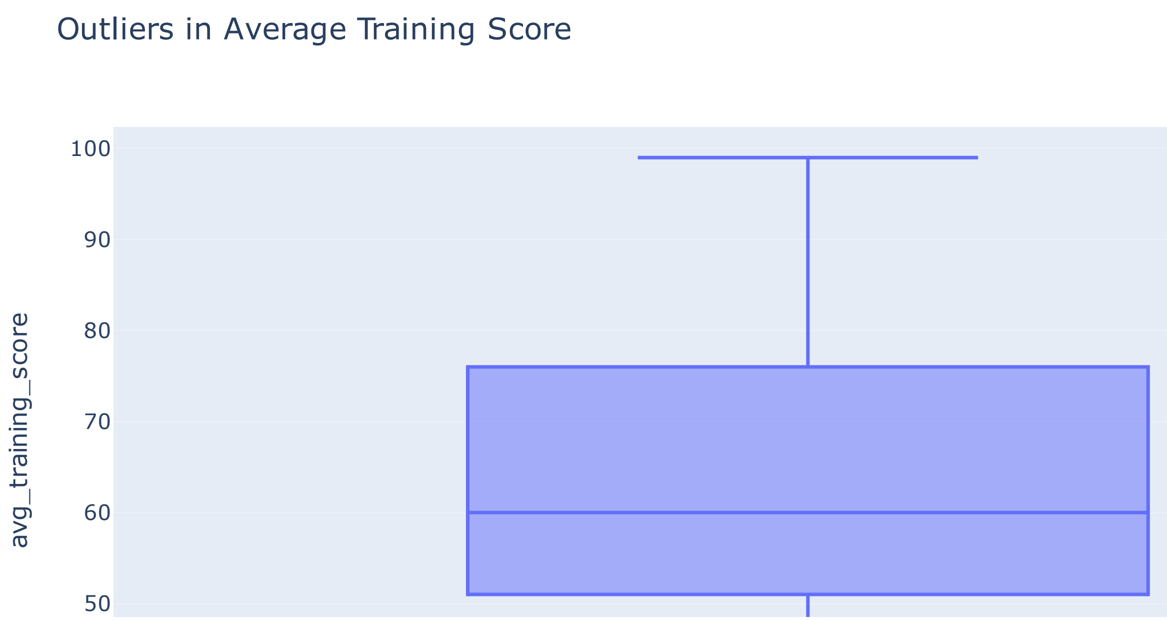


Outliers Observations

1. Average Training Score does not have a noticeable outlier
2. Length of Service returned a slight outlier. Potential max range would be 13

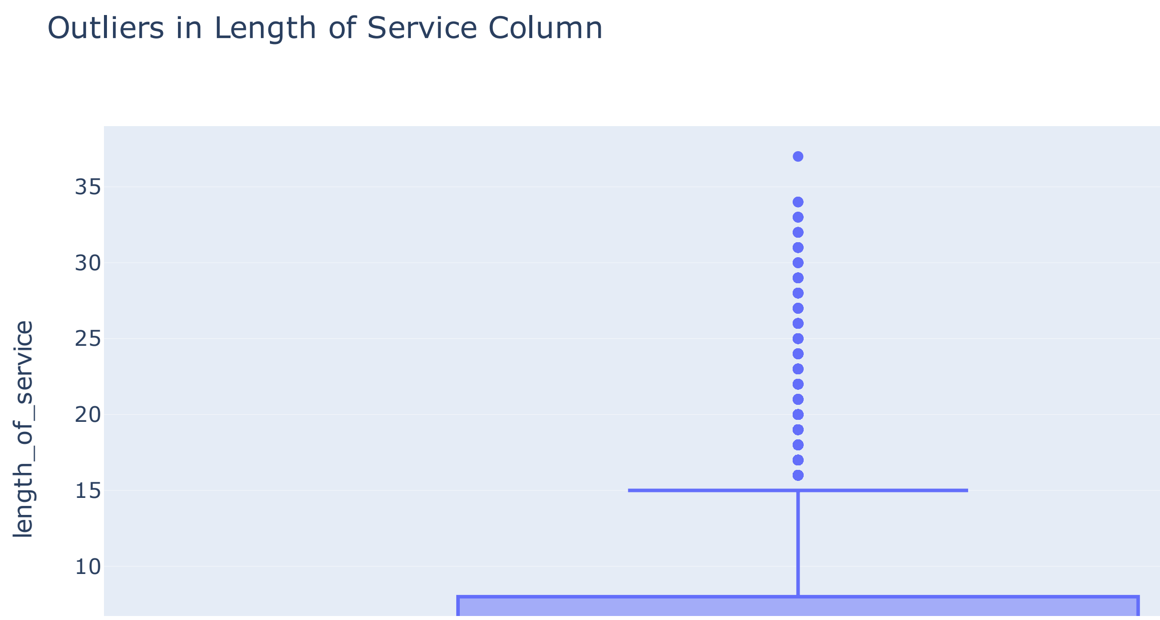
In [24]:

```
## outlier check
fig = px.box(df,y='avg_training_score',points='outliers', title='Outliers in Average Tr

fig.update_layout(hovermode='x')
```



In [25]:

```
fig = px.box(df,y='length_of_service', points='outliers', title='Outliers in Length of
fig.update_layout(hovermode='x')
```



Promoted Visualizations Observations

Department

1. R&D is the smallest department

Education

1. More than 35000 employees hold a bachelor's degree
2. At least 15000 employees have a Master's and Phd

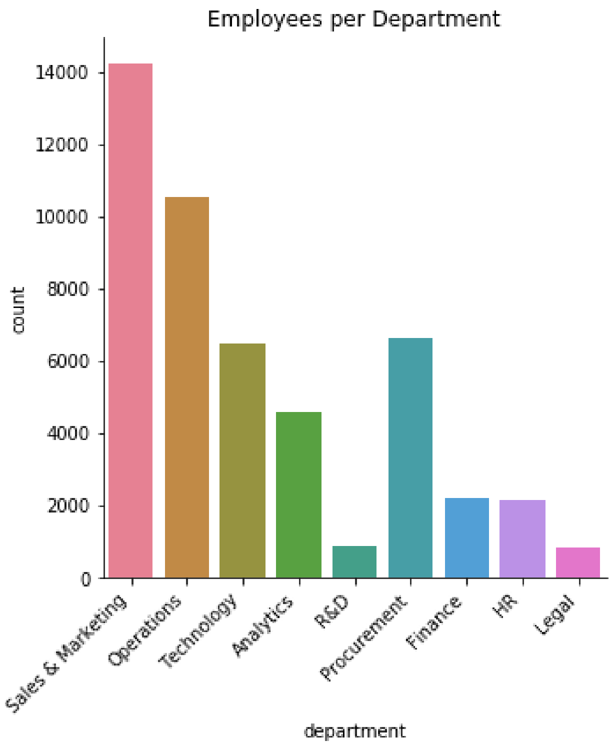
Gender

1. Male employees account for more than 35000 employees in the company
2. The number of female employees is slightly above 15000

In [28]:

```
## department distribution
plt.figure(figsize = (12, 10))
sns.catplot(x ='department', kind='count', data = df, palette = 'husl')
plt.xticks(rotation = 45, horizontalalignment = 'right')
plt.title('Employees per Department')
plt.show()
```

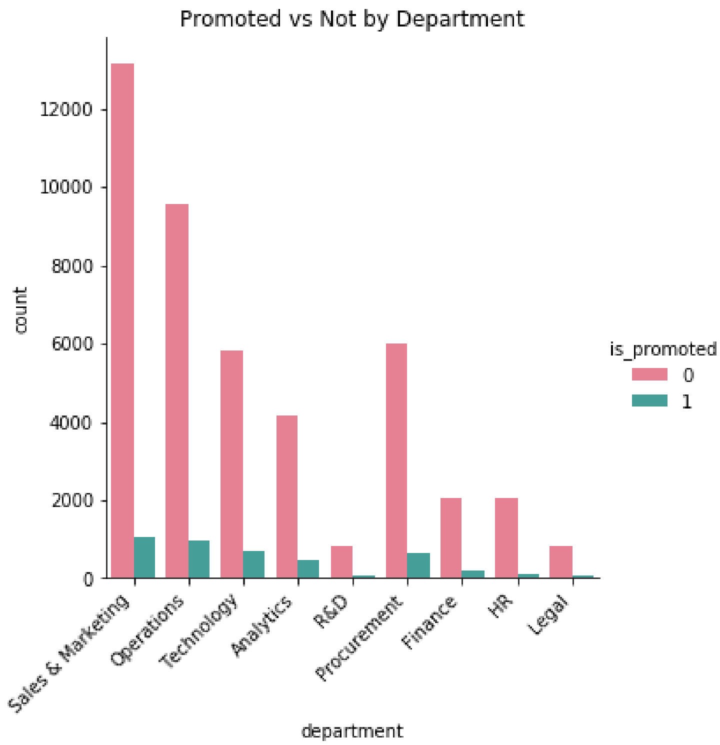
```
<Figure size 864x720 with 0 Axes>
```



In [29]:

```
## promoted vs not by department
plt.figure(figsize = (12, 10))
sns.catplot(x = 'department', hue = 'is_promoted', kind = 'count', data = df, palette=
plt.xticks(rotation=45, horizontalalignment='right')
plt.title('Promoted vs Not by Department')
plt.show()
```

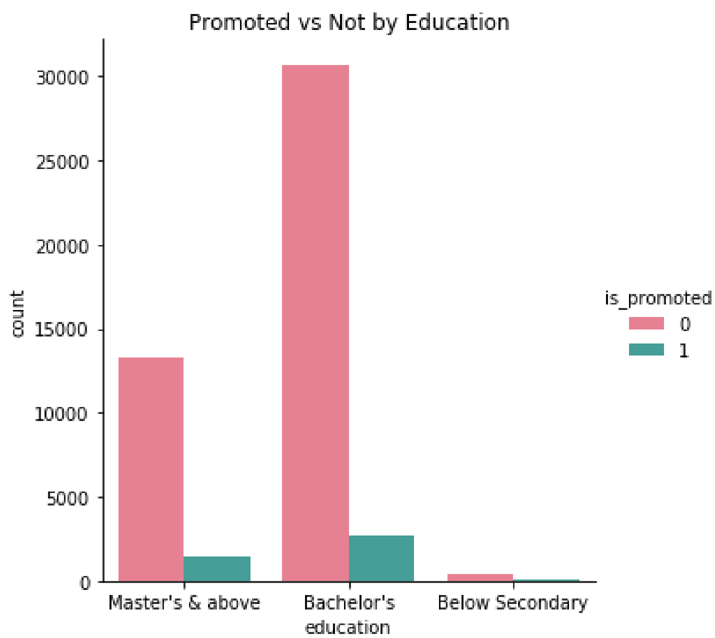
```
<Figure size 864x720 with 0 Axes>
```



In [31]:

```
## promoted vs not by education
plt.figure(figsize=(12, 10))
sns.catplot(x = 'education', hue = 'is_promoted', kind = 'count', data = df, palette =
plt.title('Promoted vs Not by Education')
plt.show()
```

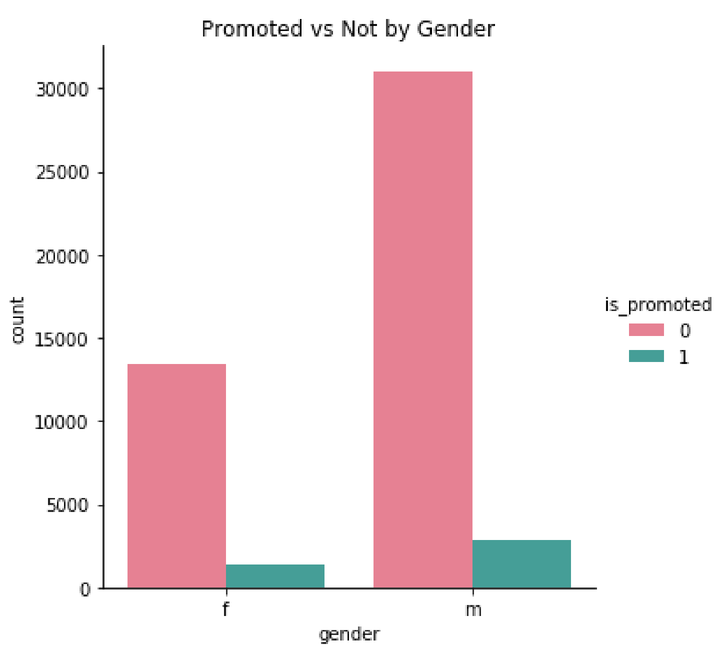
```
<Figure size 864x720 with 0 Axes>
```



In [32]:

```
## promoted vs not by gender
plt.figure(figsize=(12, 10))
sns.catplot(x = 'gender', hue = 'is_promoted', kind = 'count', data = df, palette = 'hu
plt.title('Promoted vs Not by Gender')
plt.show()
```

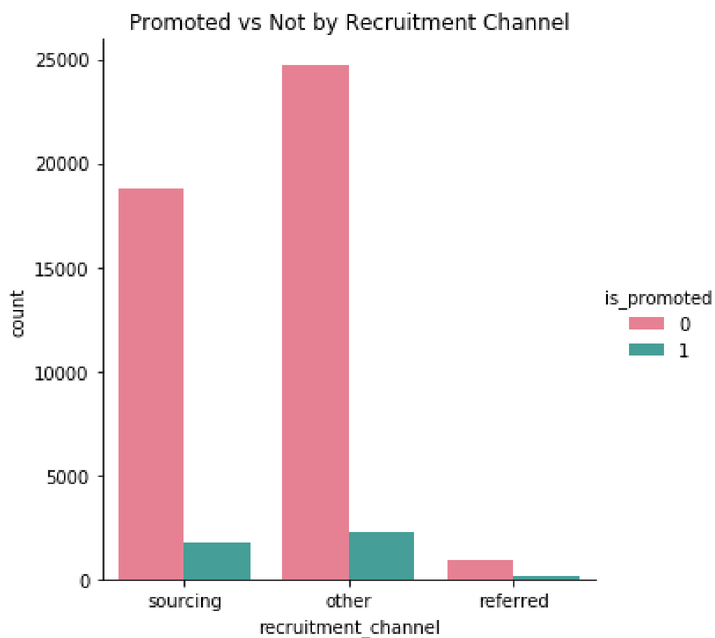
```
<Figure size 864x720 with 0 Axes>
```



In [33]:

```
## promoted vs not by recruitment channel
plt.figure(figsize=(12, 10))
sns.catplot(x = 'recruitment_channel', hue = 'is_promoted', kind = 'count', data = df,
plt.title('Promoted vs Not by Recruitment Channel')
plt.show()
```

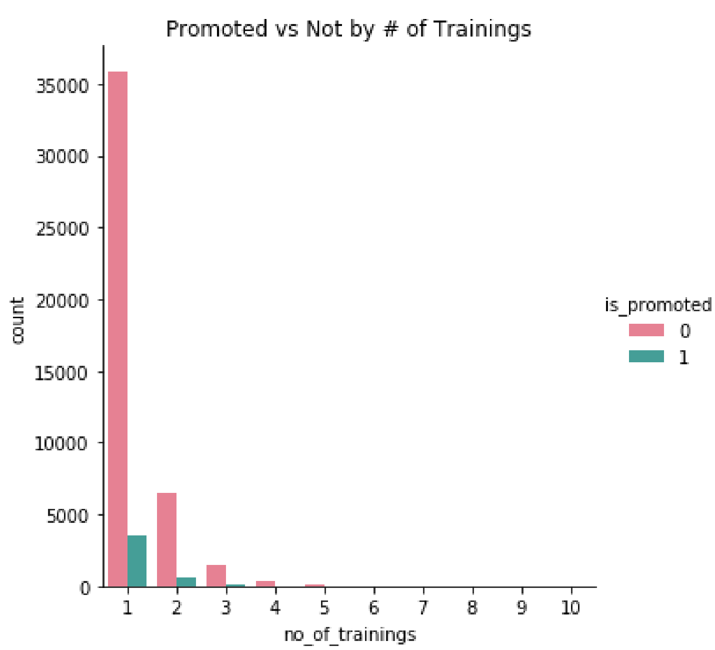
```
<Figure size 864x720 with 0 Axes>
```



In [36]:

```
## promoted vs not by # of trainings
plt.figure(figsize=(12, 10))
sns.catplot(x = 'no_of_trainings', hue = 'is_promoted', kind = 'count', data = df, pal
plt.title('Promoted vs Not by # of Trainings')
plt.show()
```

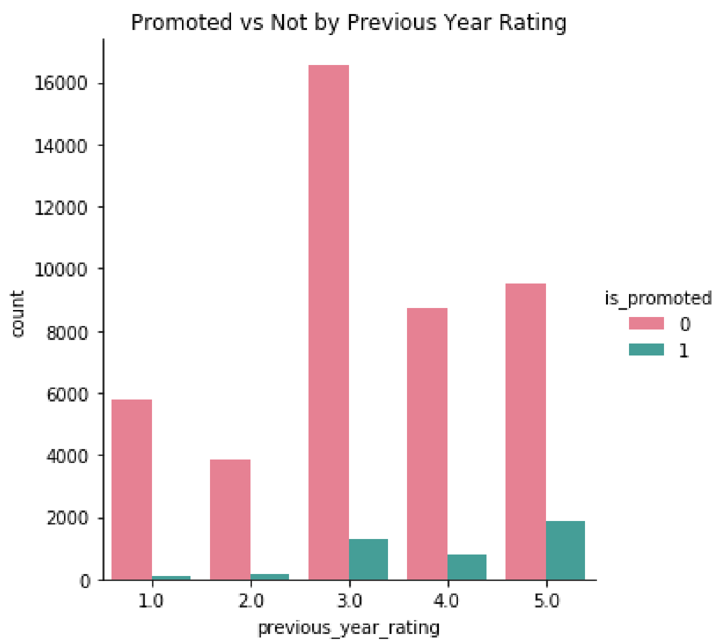
```
<Figure size 864x720 with 0 Axes>
```



In [35]:

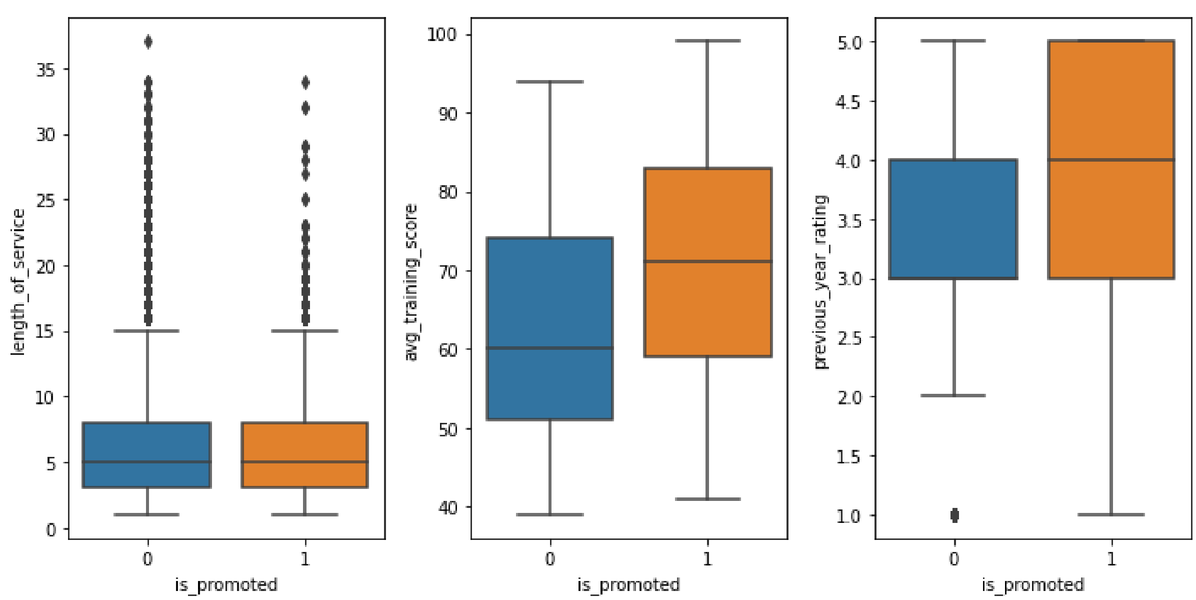
```
## promoted vs not by previuos year rating
plt.figure(figsize=(12, 10))
sns.catplot(x = 'previous_year_rating', hue = 'is_promoted', kind = 'count', data = df,
plt.title('Promoted vs Not by Previous Year Rating')
plt.show()
```

```
<Figure size 864x720 with 0 Axes>
```



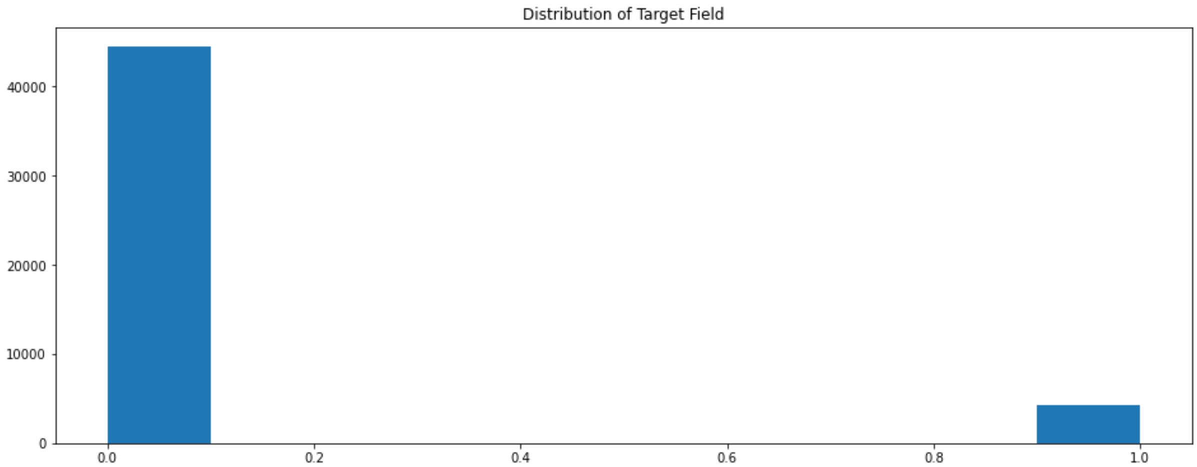
In [33]:

```
## target field vs workforce data

fig = plt.figure(figsize=(10,5) )
fig.add_subplot(1,3,1)
ar_6 = sns.boxplot(x=df["is_promoted"],y=df["length_of_service"])
fig.add_subplot(1,3,2)
ar_6 = sns.boxplot(x=df["is_promoted"],y=df["avg_training_score"])
fig.add_subplot(1,3,3)
ar_6 = sns.boxplot(x=df["is_promoted"],y=df["previous_year_rating"])
plt.tight_layout()
plt.show()
```



In [34]:

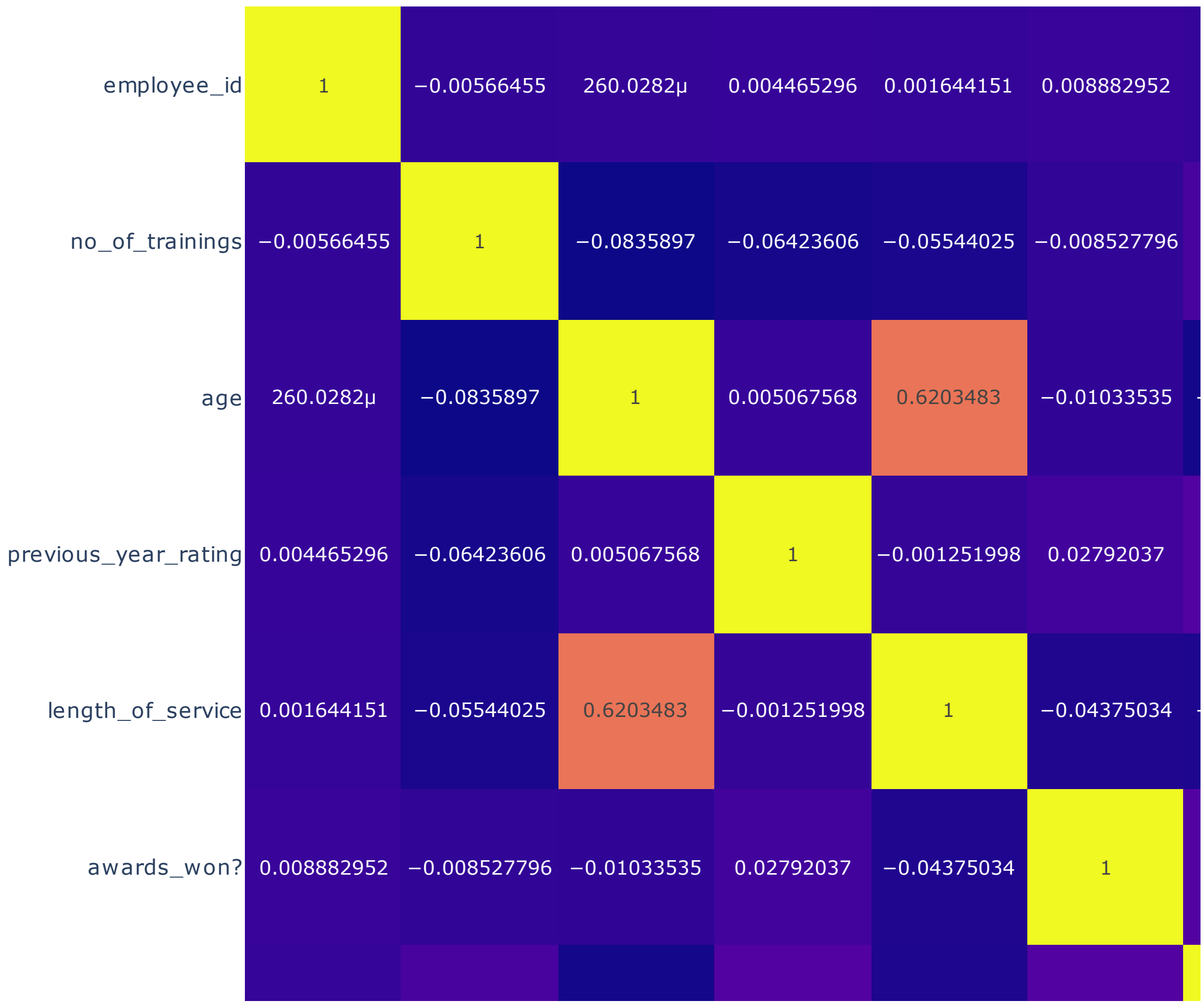
```
## distribution of target field
fig = plt.figure(figsize =(16, 6))
plt.hist(df['is_promoted'])
plt.title('Distribution of Target Field', loc = 'center', fontsize = 12)
plt.show()
```



In [116]:

```
px.imshow(df.corr(), text_auto= True, title='Correlation Between the Variables in the
```

Correlation Between the Variables in the Model



In [156]:
```python
## create dummy variables

df["gender"] = df["gender"].apply(lambda x: 1 if x=="m" else 0)

cols = df.select_dtypes(["object"]).columns
ds = pd.get_dummies(df[cols],drop_first=True)
ds

df = pd.concat([df,ds],axis=1)

## drop original columns
df.drop(cols,axis=1,inplace=True)
```

In [155]:
```python
## drop employee_id

df = df.drop('employee_id', axis=1)
df.head()
```

Out[155]:

	department	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_s
0	Sales & Marketing	Master's & above	f	sourcing	1	35	5.0	
1	Operations	Bachelor's	m	other	1	30	5.0	
2	Sales & Marketing	Bachelor's	m	sourcing	1	34	3.0	
3	Sales & Marketing	Bachelor's	m	other	2	39	1.0	
4	Technology	Bachelor's	m	other	1	45	3.0	

In [37]:
```python
df.shape
```

Out[37]: (48660, 53)

In [38]:
```python
df.head()
```

Out[38]:

	gender	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	avg_training_score	is_
0	0	1	35	5.0	8	0	49	
1	1	1	30	5.0	4	0	60	
2	1	1	34	3.0	7	0	50	
3	1	2	39	1.0	10	0	50	
4	1	1	45	3.0	2	0	73	

Logistic Regression Model

In [140]:
```
## import library
from sklearn.model_selection import train_test_split

## split data
y = df.pop("is_promoted")
X = df


X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=42,train_size=0.8)


print("train size X : ",X_train.shape)
print("train size y : ",y_train.shape)
print("test size X : ",X_test.shape)
print("test size y : ",y_test.shape)
```

```
train size X :  (38928, 52)
train size y :  (38928,)
test size X :  (9732, 52)
test size y :  (9732,)
```

In [142]:
```
from sklearn.preprocessing import StandardScaler

scale = StandardScaler()

X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

In [143]:
```
## distribution values check
y_train.value_counts(normalize=True)
```

Out[143]:
```
0    0.913558
1    0.086442
Name: is_promoted, dtype: float64
```

In [144]:
```
## import library
from sklearn.linear_model import LogisticRegression

## add class weight to address 90/10 promoted split
lr = LogisticRegression(class_weight={0:0.1,1:0.9})
lr.fit(X_train,y_train)
```

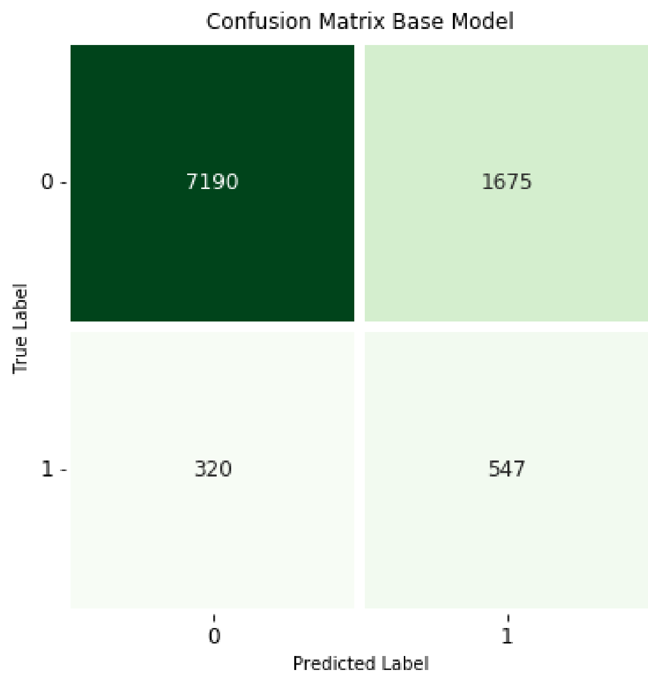
Out[144]:
```
LogisticRegression(class_weight={0: 0.1, 1: 0.9})
```

In [145]:
```
## set base model
base_model = lr
y_pred_base_model = base_model.predict(X_test)
pred_prob = base_model.predict_proba(X_test)
```

In [146]:
```
## iomport library
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_base_model)
```

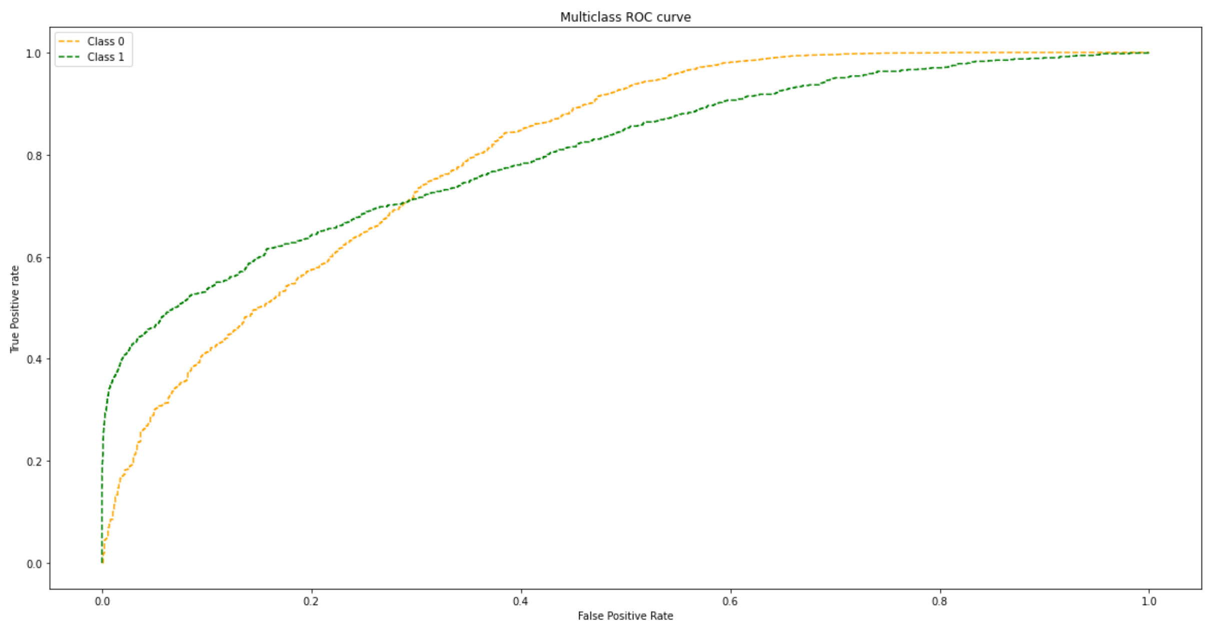
In [147]:

```
## confusion matrix of base model

df1 = pd.DataFrame(columns=["0","1"], index= ["0","1"], data= cm )

f,ax = plt.subplots(figsize=(6,6))

sns.heatmap(df1, annot=True,cmap="Greens", fmt= '.0f',
            ax=ax,linewidths = 5, cbar = False,annot_kws={"size": 12})
plt.xlabel("Predicted Label")
plt.xticks(size = 12)
plt.yticks(size = 12, rotation = 0)
plt.ylabel("True Label")
plt.title("Confusion Matrix Base Model", size = 12)
plt.show()
```



In [62]:

```
## import library
from sklearn.metrics import roc_curve

## roc curve for classes
fpr = {}
tpr = {}
thresh ={}

n_class = 2

for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)

plt.plot(fpr[0], tpr[0], linestyle='--',color='orange', label='Class 0 ')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green', label='Class 1 ')

plt.title('Multiclass ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.savefig('Multiclass ROC');
```



In [148]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred_base_model)
```

Out[148]: 0.7950061652281134

Feature Engineering

In [149]:

```
## combine 'awards_won' and 'previous_year_rating'
## combine 'avg_training_score' and 'no_of_trainings'

#Creating a sum metric column
df['sum_metric'] = df['awards_won?']+ df['previous_year_rating']

# creating a total score column
df['total_score'] = df['avg_training_score'] * df['no_of_trainings']
```

In [154]:
```
## drop region
df = df.drop(['region'], axis = 1)
```

In [157]:
```
df.columns
```

Out[157]:
```
Index(['gender', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'awards_won?', 'avg_training_score', 'is_promoted',
       'department_Finance', 'department_HR', 'department_Legal',
       'department_Operations', 'department_Procurement', 'department_R&D',
       'department_Sales & Marketing', 'department_Technology',
       'education_Below Secondary', 'education_Master's & above',
       'recruitment_channel_referred', 'recruitment_channel_sourcing'],
      dtype='object')
```

Label Encode

In [158]:
```
df.select_dtypes('object').head()
```

Out[158]:

0
1
2
3
4

In [162]:
```
df.head()
```

Out[162]:

	gender	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	avg_training_score	is_
0	0	1	35	5.0	8	0	49	
1	1	1	30	5.0	4	0	60	
2	1	1	34	3.0	7	0	50	
3	1	2	39	1.0	10	0	50	
4	1	1	45	3.0	2	0	73	

2nd Logistic Regression Model

In [163]:
```
#split target from features

y = df['is_promoted']
x = df.drop(['is_promoted'],axis=1)
```

In [165]:
```
## split data
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=42,train_size=0.8)


print("train size X : ",X_train.shape)
print("train size y : ",y_train.shape)
print("test size X : ",X_test.shape)
print("test size y : ",y_test.shape)
```

```
train size X :  (38928, 54)
train size y :  (38928,)
test size X :  (9732, 54)
test size y :  (9732,)
```

In [166]:
```
scale = StandardScaler()

X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

In [167]:
```
## distribution values check
y_train.value_counts(normalize=True)
```

Out[167]:
```
0    0.913558
1    0.086442
Name: is_promoted, dtype: float64
```

In [168]:
```
## add class weight to address 90/10 promoted split
lr = LogisticRegression(class_weight={0:0.1,1:0.9})
lr.fit(X_train,y_train)
```

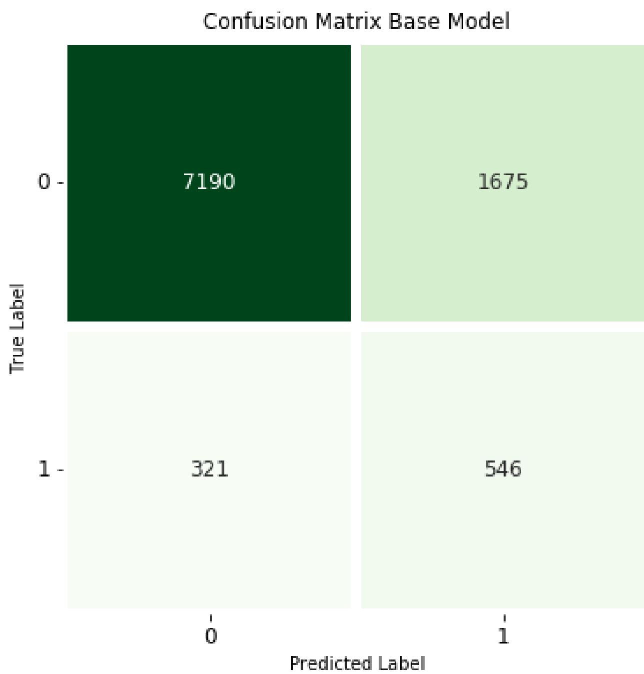
Out[168]:
```
LogisticRegression(class_weight={0: 0.1, 1: 0.9})
```

In [169]:
```
## set base model
base_model = lr
y_pred_base_model = base_model.predict(X_test)
pred_prob = base_model.predict_proba(X_test)
```

In [170]:
```
cm = confusion_matrix(y_test, y_pred_base_model)
```

In [172]:

```
## confusion matrix of base model

df1 = pd.DataFrame(columns=["0","1"], index= ["0","1"], data= cm )

f,ax = plt.subplots(figsize=(6,6))

sns.heatmap(df1, annot=True,cmap="Greens", fmt= '.0f',
            ax=ax,linewidths = 5, cbar = False,annot_kws={"size": 12})
plt.xlabel("Predicted Label")
plt.xticks(size = 12)
plt.yticks(size = 12, rotation = 0)
plt.ylabel("True Label")
plt.title("Confusion Matrix Base Model", size = 12)
plt.show()
```



In [173]:

```
accuracy_score(y_test, y_pred_base_model)
```

Out[173]: 0.7949034114262228

Feature Evaluation

In [186]:

```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
from matplotlib import pyplot

X, y = make_regression(n_samples=1000, n_features=20, n_informative=5, random_state=1)

lr = LinearRegression()

lr.fit(X, y)

importance = lr.coef_

for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.title('Feature Importance Scores')
pyplot.show()
```

```
Feature: 0, Score: 0.00000
Feature: 1, Score: -0.00000
Feature: 2, Score: -0.00000
Feature: 3, Score: 0.00000
Feature: 4, Score: -0.00000
Feature: 5, Score: 0.00000
Feature: 6, Score: 9.61372
Feature: 7, Score: 0.00000
Feature: 8, Score: 30.51944
Feature: 9, Score: 0.00000
Feature: 10, Score: 0.00000
Feature: 11, Score: 0.00000
Feature: 12, Score: 48.38204
Feature: 13, Score: 0.00000
Feature: 14, Score: -0.00000
Feature: 15, Score: -0.00000
Feature: 16, Score: 39.99774
Feature: 17, Score: -0.00000
Feature: 18, Score: -0.00000
Feature: 19, Score: 70.86224
```