Importing libraries

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# For predictive data analysis
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.model_selection import train_test_split

# Classifiers
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_recall_curve
```

2. Data Loading and Cleansing

In [2]:

```
# Load the data

train = pd.read_csv(r"C:\Users\yekul\Downloads\train (1).csv", header=0)
test=pd.read_csv(r"C:\Users\yekul\Downloads\test (1).csv",header=0)
```

In [3]:

```
train
```

Out[3]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_ >
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5.0	8	
1	65141	Operations	region_22	Bachelor's	m	other	1	30	5.0	4	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3.0	7	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	1.0	10	
4	48945	Technology	region_26	Bachelor's	m	other	1	45	3.0	2	
...	...	...	...	...	...	...	...	...	...	...	
54803	3030	Technology	region_14	Bachelor's	m	sourcing	1	48	3.0	17	
54804	74592	Operations	region_27	Master's & above	f	other	1	37	2.0	6	
54805	13918	Analytics	region_1	Bachelor's	m	other	1	27	5.0	3	
54806	13614	Sales & Marketing	region_9	NaN	m	sourcing	1	29	1.0	2	
54807	51526	HR	region_22	Bachelor's	m	other	1	27	1.0	5	

54808 rows × 14 columns

In [4]:

```
test
```

Out[4]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_ >
0	8724	Technology	region_26	Bachelor's	m	sourcing	1	24	NaN	1	
1	74430	HR	region_4	Bachelor's	f	other	1	31	3.0	5	
2	72255	Sales & Marketing	region_13	Bachelor's	m	other	1	31	1.0	4	
3	38562	Procurement	region_2	Bachelor's	f	other	3	31	2.0	9	
4	64486	Finance	region_29	Bachelor's	m	sourcing	1	30	4.0	7	
...	...	...	...	...	...	...	...	...	...	...	
23485	53478	Legal	region_2	Below Secondary	m	sourcing	1	24	3.0	1	
23486	25600	Technology	region_25	Bachelor's	m	sourcing	1	31	3.0	7	
23487	45409	HR	region_16	Bachelor's	f	sourcing	1	26	4.0	4	
23488	1186	Procurement	region_31	Bachelor's	m	sourcing	3	27	NaN	1	
23489	5973	Technology	region_17	Master's & above	m	other	3	40	5.0	5	

23490 rows × 13 columns

In [5]:

```
train.shape
```

Out[5]:

```
(54808, 14)
```

In [6]:

```
test.shape
```

Out[6]:

```
(23490, 13)
```

3.Checking the Duplicate and low variation data

In [7]:

```
train.duplicated().any()
```

Out[7]:

```
False
```

In [8]:

```
test.duplicated().any()
```

Out[8]:

```
False
```

In [9]:

```
train.describe()
```

Out[9]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?	avg_training_score	is_promoted
count	54808.000000	54808.000000	54808.000000	50684.000000	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000
mean	39195.830627	1.253011	34.803915	3.329256	5.865512	0.351974	0.023172	63.386750	0.085170
std	22586.581449	0.609264	7.660169	1.259993	4.265094	0.477590	0.150450	13.371559	0.279137
min	1.000000	1.000000	20.000000	1.000000	1.000000	0.000000	0.000000	39.000000	0.000000
25%	19669.750000	1.000000	29.000000	3.000000	3.000000	0.000000	0.000000	51.000000	0.000000
50%	39225.500000	1.000000	33.000000	3.000000	5.000000	0.000000	0.000000	60.000000	0.000000
75%	58730.500000	1.000000	39.000000	4.000000	7.000000	1.000000	0.000000	76.000000	0.000000
max	78298.000000	10.000000	60.000000	5.000000	37.000000	1.000000	1.000000	99.000000	1.000000

In [10]:

```
test.describe()
```

Out[10]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?	avg_training_score
count	23490.000000	23490.000000	23490.000000	21678.000000	23490.000000	23490.000000	23490.000000	23490.000000
mean	39041.399149	1.254236	34.782929	3.339146	5.810387	0.358834	0.022776	63.263133
std	22640.809201	0.600910	7.679492	1.263294	4.207917	0.479668	0.149191	13.411750
min	3.000000	1.000000	20.000000	1.000000	1.000000	0.000000	0.000000	39.000000
25%	19370.250000	1.000000	29.000000	3.000000	3.000000	0.000000	0.000000	51.000000
50%	38963.500000	1.000000	33.000000	3.000000	5.000000	0.000000	0.000000	60.000000
75%	58690.000000	1.000000	39.000000	4.000000	7.000000	1.000000	0.000000	76.000000
max	78295.000000	9.000000	60.000000	5.000000	34.000000	1.000000	1.000000	99.000000

In [11]:

```
train.columns
```

Out[11]:

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

In [12]:

```
test.columns
```

Out[12]:

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

4.Categorical data,Encoding Techniques,Identify and address the missing variables

In [13]:

```
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
```

In [14]:

```
train.nunique()
```

Out[14]:

```
employee_id             54808
department                  9
region                     34
education                   3
gender                      2
recruitment_channel         3
no_of_trainings            10
age                        41
previous_year_rating        5
length_of_service          35
KPIs_met >80%               2
awards_won?                 2
avg_training_score         61
is_promoted                 2
dtype: int64
```

In [15]:

```
#use LabelEncoder
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
```

```
train['department']=LE.fit_transform(train['department'])
train['region']=LE.fit_transform(train['region'])
train['education']=LE.fit_transform(train['education'])
train['recruitment_channel']=LE.fit_transform(train['recruitment_channel'])
```

In [16]:

```
#Use LabelBinarizer
from sklearn.preprocessing import LabelBinarizer
LB=LabelBinarizer()
train['gender']=LB.fit_transform(train[["gender"]])
```

In [17]:

```
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  int32  
 2   region                54808 non-null  int32  
 3   education             54808 non-null  int32  
 4   gender                54808 non-null  int32  
 5   recruitment_channel   54808 non-null  int32  
 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), int32(5), int64(8)
memory usage: 4.8 MB
```

In [18]:

```
train.isnull().sum()
```

Out[18]:

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

In [19]:

```
# Using KNN Imputer to address  missing values

# KNNImputer(missing_values=np.nan, n_neighbors=5, weights='uniform', metric='nan_euclidean',
# copy=True, add_indicator=False)

from sklearn.impute import KNNImputer

imputer_str = KNNImputer(missing_values=np.nan, n_neighbors=5, weights='uniform', metric='nan_euclidean',
copy=True, add_indicator=False)

# Fill the missing values for 'Driver_Age'

train['previous_year_rating']=imputer_str.fit_transform(train[['previous_year_rating']])
```

In [20]:

```
train.isnull().sum()
```

Out[20]:

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

In [21]:

```
test.info()
```

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

In [22]:

```
test.nunique()
```

Out[22]:

```
employee_id             23490
department                  9
region                     34
education                   3
gender                      2
recruitment_channel         3
no_of_trainings             9
age                        41
previous_year_rating        5
length_of_service          34
KPIs_met >80%               2
awards_won?                 2
avg_training_score         61
dtype: int64
```

In [23]:

```
#use LabelEncoder
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
test['department']=LE.fit_transform(test['department'])
test['region']=LE.fit_transform(test['region'])
test['education']=LE.fit_transform(test['education'])
test['recruitment_channel']=LE.fit_transform(test['recruitment_channel'])
```

In [24]:

```
#Use LabelBinarizer
from sklearn.preprocessing import LabelBinarizer
LB=LabelBinarizer()
test['gender']=LB.fit_transform(test[["gender"]])
```

In [25]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23490 entries, 0 to 23489
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   employee_id           23490 non-null  int64  
 1   department            23490 non-null  int32  
 2   region                23490 non-null  int32  
 3   education             23490 non-null  int32  
 4   gender                23490 non-null  int32  
 5   recruitment_channel   23490 non-null  int32  
 6   no_of_trainings       23490 non-null  int64  
 7   age                   23490 non-null  int64  
 8   previous_year_rating  21678 non-null  float64
 9   length_of_service     23490 non-null  int64  
 10  KPIs_met >80%         23490 non-null  int64  
 11  awards_won?           23490 non-null  int64  
 12  avg_training_score    23490 non-null  int64  
dtypes: float64(1), int32(5), int64(7)
memory usage: 1.9 MB
```

In [26]:

```
test.isnull().sum()
```

Out[26]:

```
employee_id                0
department                 0
region                     0
education                  0
gender                     0
recruitment_channel        0
no_of_trainings            0
age                        0
previous_year_rating    1812
length_of_service          0
KPIs_met >80%              0
awards_won?                0
avg_training_score         0
dtype: int64
```

In [27]:

```
# Using KNN Imputer to address  missing values

# KNNImputer(missing_values=np.nan, n_neighbors=5, weights='uniform', metric='nan_euclidean',
# copy=True, add_indicator=False)

from sklearn.impute import KNNImputer
```

```
imputer_str = KNNImputer(missing_values=np.nan, n_neighbors=5, weights='uniform', metric='nan_euclidean',
copy=True, add_indicator=False)

# Fill the missing values for 'Driver_Age'

test['previous_year_rating']=imputer_str.fit_transform(test[['previous_year_rating']])
```

In [28]:

```
test.isnull().sum()
```

Out[28]:

```
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
dtype: int64
```

5.Handling of Outliers

In [29]:

```
df1=train.copy()
```

In [30]:

```
df1.shape
```

Out[30]:

```
(54808, 14)
```

In [31]:

```
length_of_service_UL=round(df1.length_of_service.mean()+3*df1.length_of_service.std(),3)
length_of_service_LL=round(df1.length_of_service.mean()-3*df1.length_of_service.std(),3)
df2=df1[(df1.length_of_service>length_of_service_LL)&(df1.length_of_service<length_of_service_UL)]
df2.shape
```

Out[31]:

```
(53833, 14)
```

In [32]:

```
df3_EL=df2[(df2.length_of_service<length_of_service_LL)|(df1.length_of_service>length_of_service_UL)]
df3_EL
```

```
C:\Users\yekul\AppData\Local\Temp\ipykernel_16596\1915878653.py:1: UserWarning: Boolean Series key will be reindexed to match Da
taFrame index.
  df3_EL=df2[(df2.length_of_service<length_of_service_LL)|(df1.length_of_service>length_of_service_UL)]
```

Out[32]:

employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	aw

Data Sampling Methods

In [33]:

```
#Count the target or dependent variable by 0 and 1 and their proportion
#(>10:1 ,then the dataset is imbalance data
is_promoted_count=train.is_promoted.value_counts()
print('Class 0:',is_promoted_count[0])
print('Class 1:',is_promoted_count[1])
print('proportion :',round(is_promoted_count[0]/is_promoted_count[1],2),':1')
print('total records:',len(train))
```

```
Class 0: 50140
Class 1: 4668
proportion : 10.74 :1
total records: 54808
```

Selection of Dependent and Independent variables

In [34]:

```
# Identify the Independent and Target variables

IndepVar = []
for col in train.columns:
    if col != 'is_promoted':
        IndepVar.append(col)

TargetVar = 'is_promoted'
```

```
x = train[IndepVar]
y = train[TargetVar]
```

In [35]:

```
pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\yekul\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\yekul\anaconda3\lib\site-packages (from imblearn) (0.9.1)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\yekul\anaconda3\lib\site-packages (from imbalanced-learn->imblear
n) (1.1.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\yekul\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.
7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\yekul\anaconda3\lib\site-packages (from imbalanced-learn->imblea
rn) (2.2.0)
Requirement already satisfied: joblib>=1.0.0 in c:\users\yekul\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.
1.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\yekul\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.
21.5)
Note: you may need to restart the kernel to use updated packages.
```

Feature Scaling

In [36]:

```
# Random oversampling can be implemented using the RandomOverSampler class

from imblearn.over_sampling import RandomOverSampler

oversample = RandomOverSampler(sampling_strategy=0.15)

x_over, y_over = oversample.fit_resample(x, y)

print(x_over.shape)
print(y_over.shape)
```

```
(57661, 13)
(57661,)
```

In [37]:

```
# Split the data into train and test (random sampling)

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=42)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[37]:
```
((38365, 13), (16443, 13), (38365,), (16443,))
```

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

Out[38]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_me >80%
3113	27996	8	3	0	1	0	2	26	2.0	3	
118	38771	5	15	0	0	0	1	27	4.0	4	
17005	55699	1	20	0	1	0	3	34	3.0	6	
14505	64111	0	31	0	1	0	2	40	5.0	9	
31487	27201	5	8	0	1	0	1	45	3.0	17	

In [39]:
```python
train.columns
```

Out[39]:
```
Index(['employee_id', 'department', 'region', 'education', 'gender',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'KPIs_met >80%', 'awards_won?',
       'avg_training_score', 'is_promoted'],
      dtype='object')
```

In [40]:
```python
cols1=['employee_id', 'department', 'region', 'education', 'gender','recruitment_channel', 'no_of_trainings', 'age',
       'previous_year_rating','length_of_service', 'KPIs_met >80%', 'awards_won?','avg_training_score']
```

In [41]:
```python
# Scaling the features by using MinMaxScaler

from sklearn.preprocessing import MinMaxScaler

mmscaler = MinMaxScaler(feature_range=(0, 1))

x_train[cols1] = mmscaler.fit_transform(x_train[cols1])
x_train = pd.DataFrame(x_train)

x_test[cols1] = mmscaler.fit_transform(x_test[cols1])
x_test = pd.DataFrame(x_test)
```

In [44]:

```
#load the results

CSResults=pd.read_csv(r"C:\Users\yekul\Documents\intern\HTResults.csv",header=0)
CSResults.head()
```

Out[44]:

Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity	MCC	ROC_AUC_Score	Balanced Accuracy

Models Used for Development

In [45]:

```
# Build the Calssification models and compare the results

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
import lightgbm as lgb

# Create objects of classification algorithm with default hyper-parameters

ModelLR = LogisticRegression()
ModelDC = DecisionTreeClassifier()
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=5)
ModelSVM = SVC(probability=True)
modelBAG = BaggingClassifier(base_estimator=None, n_estimators=100, max_samples=1.0, max_features=1.0,bootstrap=True,
                             bootstrap_features=False, oob_score=False, warm_start=False,n_jobs=None, random_state=None,
                             verbose=0)
ModelGB = GradientBoostingClassifier(loss='deviance', learning_rate=0.1,n_estimators=100, subsample=1.0,
                                     criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
                                     min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0,
                                     init=None,random_state=None,max_features=None, verbose=0, max_leaf_nodes=None,
```