Loading the Dataset

Finding the best agent for a company using the past recruitment data

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('C:\\Users\\SIGMA\\Desktop\\Machine Learning\\10. Logistic Regression\\Module 10\\helper text\\3. Implen
df.shape
```

Out[2]:
```
(8844, 19)
```

In [3]:
```
df.head()
```

Out[3]:

	ID	Office_PIN	Applicant_City_PIN	Applicant_Gender	Applicant_Marital_Status	Applicant_Occupation	Applicant_Qualification	Manager_Jo
0	FIN1000001	842001	844120	M	M	Others	Graduate	
1	FIN1000002	842001	844111	M	S	Others	Class XII	
2	FIN1000003	800001	844101	M	M	Business	Class XII	
3	FIN1000004	814112	814112	M	S	Salaried	Class XII	
4	FIN1000005	814112	815351	M	M	Others	Class XII	

In [4]:
```
df.columns
```

Out[4]:
```
Index(['ID', 'Office_PIN', 'Applicant_City_PIN', 'Applicant_Gender',
       'Applicant_Marital_Status', 'Applicant_Occupation',
       'Applicant_Qualification', 'Manager_Joining_Designation',
       'Manager_Current_Designation', 'Manager_Grade', 'Manager_Status',
       'Manager_Gender', 'Manager_Num_Application', 'Manager_Num_Coded',
       'Manager_Business', 'Manager_Num_Products', 'Manager_Business2',
       'Manager_Num_Products2', 'Business_Sourced'],
      dtype='object')
```

Imputing Missing Values

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
ID                                0
Office_PIN                        0
Applicant_City_PIN                0
Applicant_Gender                 53
Applicant_Marital_Status         59
Applicant_Occupation           1090
Applicant_Qualification          71
Manager_Joining_Designation       0
Manager_Current_Designation       0
Manager_Grade                     0
Manager_Status                    0
Manager_Gender                    0
Manager_Num_Application           0
Manager_Num_Coded                 0
Manager_Business                  0
Manager_Num_Products              0
Manager_Business2                 0
Manager_Num_Products2             0
Business_Sourced                  0
dtype: int64
```

```
In [7]: df[['Applicant_Gender','Applicant_Marital_Status', 'Applicant_Occupation', 'Applicant_Qualification']].head()
```

Out[7]:

	Applicant_Gender	Applicant_Marital_Status	Applicant_Occupation	Applicant_Qualification
0	M	M	Others	Graduate
1	M	S	Others	Class XII
2	M	M	Business	Class XII
3	M	S	Salaried	Class XII
4	M	M	Others	Class XII

1. Missing Values in Applicant Gender

```
In [8]: #checking value_Counts
        df['Applicant_Gender'].value_counts()
```

```
Out[8]: M    6656
        F    2135
        Name: Applicant_Gender, dtype: int64
```

```
In [9]: #imputing missing with mode
        df['Applicant_Gender'].fillna('M', inplace=True)
```

2. Missing Values in Applicant Marital Status

```
In [10]: #checking value_Counts
         df['Applicant_Marital_Status'].value_counts()
```

```
Out[10]: M    5733
         S    3042
         W       6
         D       4
         Name: Applicant_Marital_Status, dtype: int64
```

```
In [11]: #imputing missing with mode

         df['Applicant_Marital_Status'].fillna('M', inplace=True)
```

3. Missing Values in Applicant Occupation

```
In [12]: #checking value_Counts
         df['Applicant_Occupation'].value_counts()

Out[12]: Salaried         3546
         Business         2157
         Others           1809
         Self Employed     146
         Student            96
         Name: Applicant_Occupation, dtype: int64
```

```
In [13]: #imputing missing with mode

         df['Applicant_Occupation'].fillna('Salaried', inplace=True)
```

4. Missing Values in Applicant Qualification

```
In [14]: #checking value_Counts
         df['Applicant_Qualification'].value_counts()

Out[14]: Class XII                                                    5426
         Graduate                                                     2958
         Class X                                                       195
         Others                                                        116
         Masters of Business Administration                             71
         Associate / Fellow of Institute of Chartered Accountans of India      3
         Professional Qualification in Marketing                         1
         Associate/Fellow of Insurance Institute of India                1
         Associate/Fellow of Acturial Society of India                   1
         Associate/Fellow of Institute of Company Secretories of India        1
         Name: Applicant_Qualification, dtype: int64
```

In [15]:
```
#imputing missing with mode

df['Applicant_Qualification'].fillna('Class XII', inplace=True)
```

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
ID                              0
Office_PIN                      0
Applicant_City_PIN              0
Applicant_Gender                0
Applicant_Marital_Status        0
Applicant_Occupation            0
Applicant_Qualification         0
Manager_Joining_Designation     0
Manager_Current_Designation     0
Manager_Grade                   0
Manager_Status                  0
Manager_Gender                  0
Manager_Num_Application         0
Manager_Num_Coded               0
Manager_Business                0
Manager_Num_Products            0
Manager_Business2               0
Manager_Num_Products2           0
Business_Sourced                0
dtype: int64
```

Dealing with Categorical Variables

```
In [17]: df.dtypes
```

```
Out[17]: ID                               object
         Office_PIN                        int64
         Applicant_City_PIN                int64
         Applicant_Gender                 object
         Applicant_Marital_Status         object
         Applicant_Occupation             object
         Applicant_Qualification          object
         Manager_Joining_Designation      object
         Manager_Current_Designation      object
         Manager_Grade                   float64
         Manager_Status                   object
         Manager_Gender                   object
         Manager_Num_Application         float64
         Manager_Num_Coded               float64
         Manager_Business                float64
         Manager_Num_Products            float64
         Manager_Business2               float64
         Manager_Num_Products2           float64
         Business_Sourced                  int64
         dtype: object
```

```
In [38]: df.columns
```

```
Out[38]: Index(['ID', 'Office_PIN', 'Applicant_City_PIN', 'Applicant_Gender',
                'Applicant_Marital_Status', 'Applicant_Occupation',
                'Applicant_Qualification', 'Manager_Joining_Designation',
                'Manager_Current_Designation', 'Manager_Grade', 'Manager_Status',
                'Manager_Gender', 'Manager_Num_Application', 'Manager_Num_Coded',
                'Manager_Business', 'Manager_Num_Products', 'Manager_Business2',
                'Manager_Num_Products2', 'Business_Sourced'],
               dtype='object')
```

```
In [18]: categorical_cols = ['Applicant_Gender','Applicant_Marital_Status','Applicant_Occupation','Applicant_Qualification',
                            'Manager_Joining_Designation', 'Manager_Current_Designation', 'Manager_Status', 'Manager_Gender']

         for i in categorical_cols:
             print('*****', i, '*****')
             print(df[i].value_counts())
             print('')
```

```
***** Applicant_Gender *****
M    6709
F    2135
Name: Applicant_Gender, dtype: int64

***** Applicant_Marital_Status *****
M    5792
S    3042
W       6
D       4
Name: Applicant_Marital_Status, dtype: int64

***** Applicant_Occupation *****
Salaried         4636
Business         2157
Others           1809
Self Employed     146
Student            96
Name: Applicant_Occupation, dtype: int64

***** Applicant_Qualification *****
Class XII                                                    5497
Graduate                                                     2958
Class X                                                       195
Others                                                        116
Masters of Business Administration                             71
Associate / Fellow of Institute of Chartered Accountans of India      3
Professional Qualification in Marketing                         1
Associate/Fellow of Insurance Institute of India                1
Associate/Fellow of Acturial Society of India                   1
Associate/Fellow of Institute of Company Secretories of India      1
Name: Applicant_Qualification, dtype: int64
```

```
***** Manager_Joining_Designation *****
Level 1    4632
Level 2    2787
Level 3    1146
Level 4     200
Other        58
Level 6      18
Level 7       2
Level 5       1
Name: Manager_Joining_Designation, dtype: int64

***** Manager_Current_Designation *****
Level 2    3208
Level 1    2479
Level 3    2033
Level 4    1031
Level 5      93
Name: Manager_Current_Designation, dtype: int64

***** Manager_Status *****
Confirmation    5277
Probation       3567
Name: Manager_Status, dtype: int64

***** Manager_Gender *****
M    7627
F    1217
Name: Manager_Gender, dtype: int64
```

```
In [19]: df = pd.get_dummies(df)
```

Logistic regression

Train Test split

In [23]:
```
x = df.drop(['Business_Sourced'],axis=1)
y = df['Business_Sourced']
```

In [51]:
```
from sklearn.model_selection import train_test_split
train_x, valid_x, train_y, valid_y= train_test_split(x, y, test_size = 0.3, random_state=1)
```

Linear regression Model

In [113]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, auc
from sklearn.metrics import roc_auc_score
```

In [53]:
```
logreg = LogisticRegression()
logreg.fit(train_x, train_y)
```

```
/home/aishwarya/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solv
er will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[53]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [115]:
```
pred_train = logreg.predict_proba(train_x)
pred_valid = logreg.predict_proba(valid_x)
```

In [117]:
```
roc_auc_score(train_y, pred_train[:,1])
```

Out[117]:
```
0.4723336326910138
```

In [114]:
```
roc_auc_score(valid_y, pred_valid[:,1])
```

Out[114]:
```
0.4697613206972208
```