PLACEMENT PREDICTION

By: Sakshi Neeraj

This is a study of publicly accessible data regarding on-campus placement of applicants depending on characteristics such as high school graduation percentage and domain of specialisation.

Dataset for practical- https://www.kaggle.com/benroshan/factors-affecting-campus-placement (https://www.kaggle.com/benroshan/factors-affecting-campus-placement)

In [1]:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

Importing the dataset

In [2]:

```
data=pd.read_csv("Placement_Data_Full_Class.csv")
```

In [3]:

```
data.describe()
```

Out[3]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [4]:

```
data.describe(include="all")
```

Out[4]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	
count	215.000000	215	215.000000	215	215.000000	215	215	215.000000	
unique	NaN	2	NaN	2	NaN	2	3	NaN	
top	NaN	M	NaN	Central	NaN	Others	Commerce	NaN	Co
freq	NaN	139	NaN	116	NaN	131	113	NaN	
mean	108.000000	NaN	67.303395	NaN	66.333163	NaN	NaN	66.370186	
std	62.209324	NaN	10.827205	NaN	10.897509	NaN	NaN	7.358743	
min	1.000000	NaN	40.890000	NaN	37.000000	NaN	NaN	50.000000	
25%	54.500000	NaN	60.600000	NaN	60.900000	NaN	NaN	61.000000	
50%	108.000000	NaN	67.000000	NaN	65.000000	NaN	NaN	66.000000	
75%	161.500000	NaN	75.700000	NaN	73.000000	NaN	NaN	72.000000	
max	215.000000	NaN	89.400000	NaN	97.700000	NaN	NaN	91.000000	

In [5]:

```
print(data.shape)
print(data.head())
```

```
(215, 15)
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0      1      M  67.00   Others  91.00   Others  Commerce     58.00
1      2      M  79.33  Central  78.33   Others   Science     77.48
2      3      M  65.00  Central  68.00  Central      Arts     64.00
3      4      M  56.00  Central  52.00  Central   Science     52.00
4      5      M  85.80  Central  73.60  Central  Commerce     73.30

    degree_t workex  etest_p specialisation  mba_p      status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
```

Handling missing values

In [6]:

```
data.isnull()
```

Out[6]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
210	False	False	False	False	False	False	False	False	False	False	False
211	False	False	False	False	False	False	False	False	False	False	False
212	False	False	False	False	False	False	False	False	False	False	False
213	False	False	False	False	False	False	False	False	False	False	False
214	False	False	False	False	False	False	False	False	False	False	False

215 rows × 15 columns

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
sl_no            0
gender           0
ssc_p            0
ssc_b            0
hsc_p            0
hsc_b            0
hsc_s            0
degree_p         0
degree_t         0
workex           0
etest_p          0
specialisation   0
mba_p            0
status           0
salary          67
dtype: int64
```

Replacing null salaries by 0

In [8]:

```
data['salary'].fillna(value=0 , inplace = True )
```

In [9]:

```
data.isnull()
```

Out[9]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
210	False	False	False	False	False	False	False	False	False	False	False
211	False	False	False	False	False	False	False	False	False	False	False
212	False	False	False	False	False	False	False	False	False	False	False
213	False	False	False	False	False	False	False	False	False	False	False
214	False	False	False	False	False	False	False	False	False	False	False

215 rows × 15 columns

In [10]:

```
data.isnull().sum()
```

Out[10]:

```
sl_no            0
gender           0
ssc_p            0
ssc_b            0
hsc_p            0
hsc_b            0
hsc_s            0
degree_p         0
degree_t         0
workex           0
etest_p          0
specialisation   0
mba_p            0
status           0
salary           0
dtype: int64
```

In [11]:

```
data.drop(['sl_no','ssc_b','hsc_b'], axis = 1 , inplace = True)
```

In [12]:

```
print(data.head())
```

```
  gender  ssc_p  hsc_p     hsc_s  degree_p   degree_t workex  etest_p  \
0      M  67.00  91.00  Commerce     58.00   Sci&Tech     No     55.0   
1      M  79.33  78.33   Science     77.48   Sci&Tech    Yes     86.5   
2      M  65.00  68.00      Arts     64.00  Comm&Mgmt     No     75.0   
3      M  56.00  52.00   Science     52.00   Sci&Tech     No     66.0   
4      M  85.80  73.60  Commerce     73.30  Comm&Mgmt     No     96.8   

  specialisation  mba_p      status    salary  
0         Mkt&HR  58.80      Placed  270000.0  
1        Mkt&Fin  66.28      Placed  200000.0  
2        Mkt&Fin  57.80      Placed  250000.0  
3         Mkt&HR  59.43  Not Placed       0.0  
4        Mkt&Fin  55.50      Placed  425000.0  
```

Encoding

In [13]:

```
columns= ['gender','hsc_s','degree_t','workex','specialisation','status']
label_encoder = LabelEncoder()
for column in columns:
    data[column]= label_encoder.fit_transform(data[column])
data.head()
```

Out[13]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p
0	1	67.00	91.00	1	58.00	2	0	55.0	1	58.80
1	1	79.33	78.33	2	77.48	2	1	86.5	0	66.28
2	1	65.00	68.00	0	64.00	0	0	75.0	0	57.80
3	1	56.00	52.00	2	52.00	2	0	66.0	1	59.43
4	1	85.80	73.60	1	73.30	0	0	96.8	0	55.50

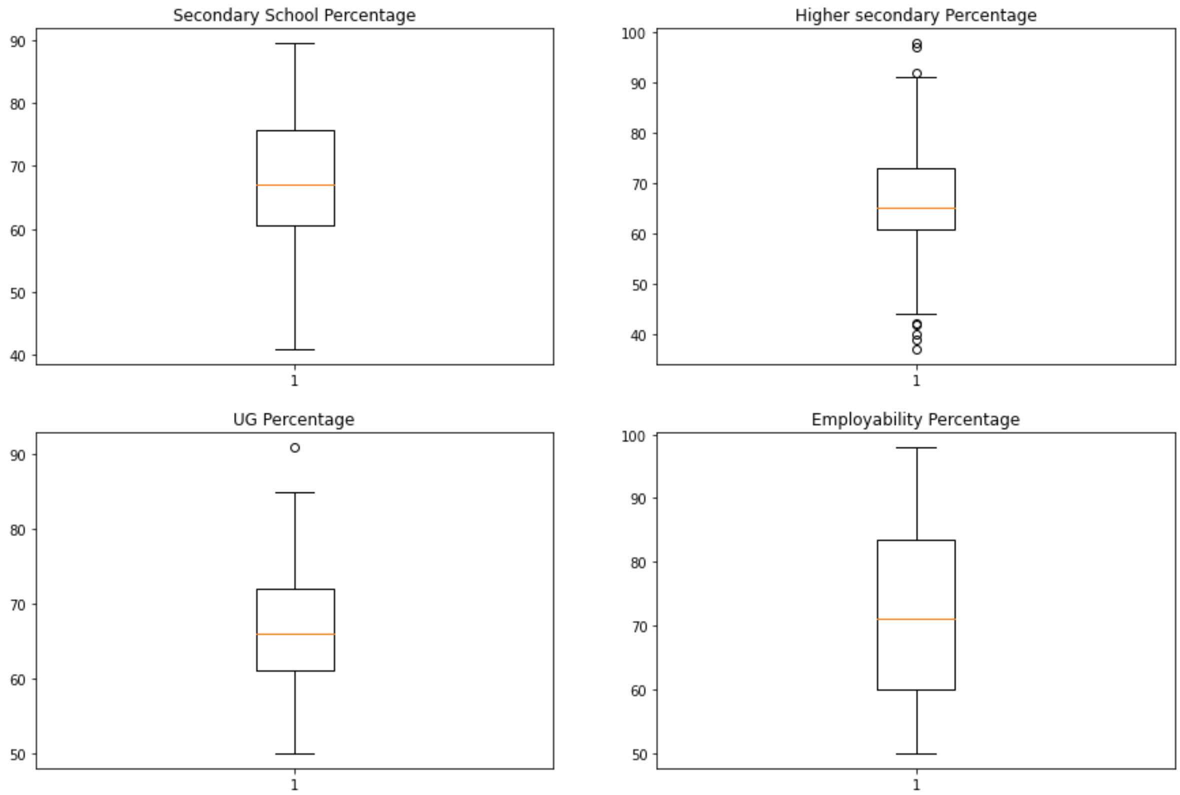
Outlier detection

In [14]:

```
plt.figure(figsize = (15,10))

ax = plt.subplot(221)
plt.boxplot(data['ssc_p'])
ax.set_title('Secondary School Percentage')

ax = plt.subplot(222)
plt.boxplot(data['hsc_p'])
ax.set_title('Higher secondary Percentage')

ax = plt.subplot(223)
plt.boxplot(data['degree_p'])
ax.set_title('UG Percentage')

ax = plt.subplot(224)
plt.boxplot(data['etest_p'])
ax.set_title('Employability Percentage')
```

Out[14]:

```
Text(0.5, 1.0, 'Employability Percentage')
```

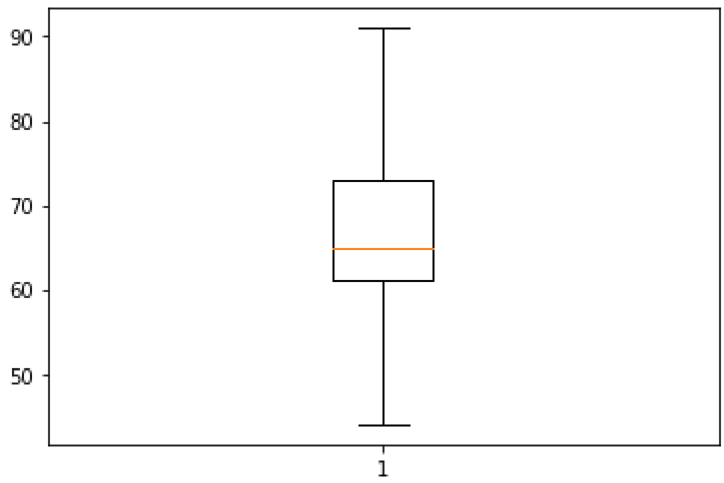


In [15]:

```
Q1 = data['hsc_p'].quantile(0.25)
Q3 = data['hsc_p'].quantile(0.75)
IQR = Q3 - Q1

filter = (data['hsc_p'] >= Q1 - 1.5 * IQR) & (data['hsc_p']<= Q3+ 1.5*IQR)
data_filtered = data.loc[filter]

plt.boxplot(data_filtered['hsc_p'])
```

Out[15]:

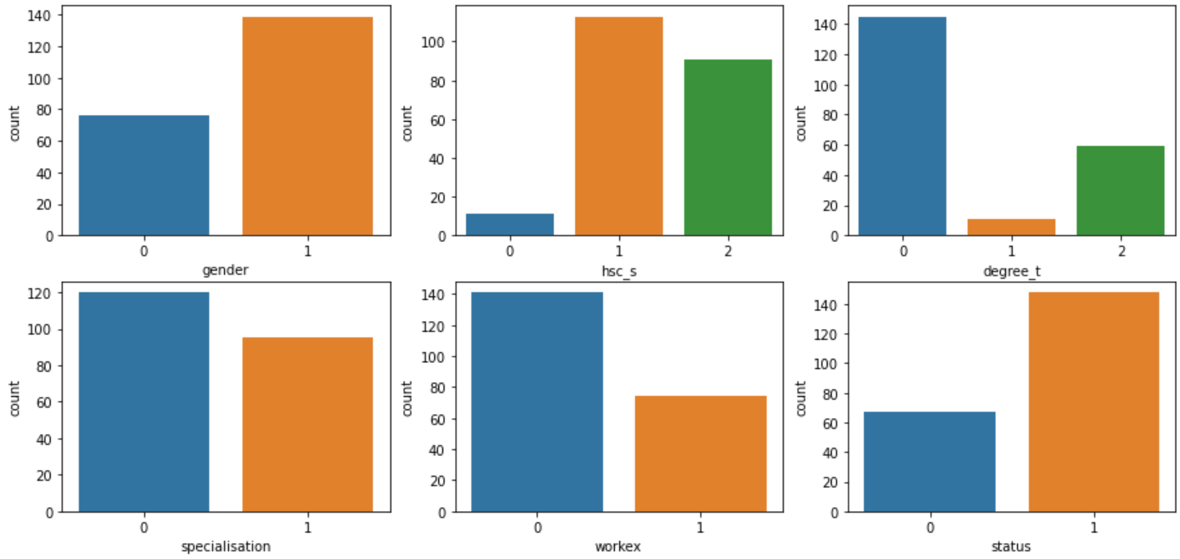
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ba0236ddc0>,
  <matplotlib.lines.Line2D at 0x1ba0237d160>],
 'caps': [<matplotlib.lines.Line2D at 0x1ba0237d4c0>,
  <matplotlib.lines.Line2D at 0x1ba0237d820>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ba0236dc40>],
 'medians': [<matplotlib.lines.Line2D at 0x1ba0237db80>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ba0237dee0>],
 'means': []}
```



Visualisation

In [16]:

```
plt.figure(figsize = (15,7))
plt.subplot(231)
ax = sns.countplot(x= 'gender' , data = data)
plt.subplot(232)
ax = sns.countplot(x= 'hsc_s' , data = data)
plt.subplot(233)
ax = sns.countplot(x= 'degree_t' , data = data)
plt.subplot(234)
ax = sns.countplot(x= 'specialisation' , data = data)
plt.subplot(235)
ax = sns.countplot(x= 'workex' , data = data)
plt.subplot(236)
ax = sns.countplot(x= 'status' , data = data)
```



Normalization

In [17]:

```
from sklearn.model_selection import train_test_split
x=data.iloc[:,0:10]
y=data.iloc[:,10]
x_train, x_test,y_train,y_test = train_test_split(x,y,test_size=.2,random_state=0)
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler
import numpy as np
norm = MinMaxScaler().fit(x_train.iloc[:, np.r_[1,2,4,7,9]])
train_norm=norm.transform(x_train.iloc[:, np.r_[1,2,4,7,9]])
test_norm=norm.transform(x_test.iloc[:, np.r_[1,2,4,7,9]])
```

In [18]:

```
x_training = pd.DataFrame()
x_training['0']=x_train.iloc[:,0]
x_training['1']=pd.DataFrame(train_norm).iloc[:,0]
x_training['2']=pd.DataFrame(train_norm).iloc[:,1]
x_training['3']=x_train.iloc[:,3]
x_training['4']=pd.DataFrame(train_norm).iloc[:,2]
x_training['5']=x_train.iloc[:,5]
x_training['6']=x_train.iloc[:,6]
x_training['7']=pd.DataFrame(train_norm).iloc[:,3]
x_training['8']=x_train.iloc[:,8]
x_training['9']=pd.DataFrame(train_norm).iloc[:,4]
print(x_training)
```

```
     0         1         2  3         4  5  6         7  8         9
16   1  0.231620  0.620428  1  0.241463  0  1  0.128125  0  0.553973
135  0  0.956625  0.378913  2  0.571463  0  0  0.208333  1  0.377811
122  0  0.457818  0.411862  0  0.243902  0  1  0.270833  0  0.043853
22   0  0.219258  0.280066  2  0.268293  2  0  0.208333  1  0.353448
80   0  0.653004  0.356837  1  0.195122  0  1  0.125000  1  0.378186
..  ..       ...       ... ..       ... .. ..       ... ..       ...
67   1  0.609629  0.593081  1  0.365854  0  0  0.416667  0  0.228636
192  1       NaN       NaN  1       NaN  0  1       NaN  0       NaN
117  1  0.306007  0.494234  2  0.341463  2  0  0.208333  0  0.602324
47   1  0.240946  0.296540  1  0.153659  0  1  0.187500  0  0.507121
172  1       NaN       NaN  1       NaN  0  0       NaN  1       NaN

[172 rows x 10 columns]
```

In [19]:

```
print(x_training.isnull().sum()* 100 / len(x_training))
```

```
0     0.000000
1    20.930233
2    20.930233
3     0.000000
4    20.930233
5     0.000000
6     0.000000
7    20.930233
8     0.000000
9    20.930233
dtype: float64
```

In [20]:

```
# Even when the percentage of null values is less than 40%, we are not imputing the values
```

In [21]:

```
print(x_training.index[x_training.isnull().any(1)].tolist())
x_training.dropna(axis=0,inplace=True)
print(x_training)
```

```
[197, 173, 179, 187, 194, 210, 188, 200, 206, 208, 191, 182, 205, 212, 196,
213, 201, 183, 178, 199, 184, 214, 186, 176, 189, 180, 177, 175, 207, 202, 1
74, 204, 193, 195, 192, 172]
     0         1         2  3         4  5  6         7  8         9
16   1  0.231620  0.620428  1  0.241463  0  1  0.128125  0  0.553973
135  0  0.956625  0.378913  2  0.571463  0  0  0.208333  1  0.377811
122  0  0.457818  0.411862  0  0.243902  0  1  0.270833  0  0.043853
22   0  0.219258  0.280066  2  0.268293  2  0  0.208333  1  0.353448
80   0  0.653004  0.356837  1  0.195122  0  1  0.125000  1  0.378186
..  ..       ...       ... ..       ... .. ..       ... ..       ...
9    1  0.934071  0.263591  1  1.000000  0  0  0.194167  0  0.693403
103  1  0.333767  0.425535  2  0.238780  2  1  0.208333  1  0.227886
67   1  0.609629  0.593081  1  0.365854  0  0  0.416667  0  0.228636
117  1  0.306007  0.494234  2  0.341463  2  0  0.208333  0  0.602324
47   1  0.240946  0.296540  1  0.153659  0  1  0.187500  0  0.507121

[136 rows x 10 columns]
```

In [22]:

```
y_training=y_train.drop([197, 173, 179, 187, 194, 210, 188, 200, 206, 208, 191, 182, 205, 2
213, 201, 183, 178, 199, 184, 214, 186, 176, 189, 180, 177, 175, 207, 202, 174, 204, 193, 1
print(y_training)
```

```
16     1
135    1
122    1
22     1
80     1
      ..
9      0
103    1
67     1
117    1
47     1
Name: status, Length: 136, dtype: int32
```

In [23]:

```
x_testing = pd.DataFrame()
x_testing['0']=x_test.iloc[:,0]
x_testing['1']=pd.DataFrame(test_norm).iloc[:,0]
x_testing['2']=pd.DataFrame(test_norm).iloc[:,1]
x_testing['3']=x_test.iloc[:,3]
x_testing['4']=pd.DataFrame(test_norm).iloc[:,2]
x_testing['5']=x_test.iloc[:,5]
x_testing['6']=x_test.iloc[:,6]
x_testing['7']=pd.DataFrame(test_norm).iloc[:,3]
x_testing['8']=x_test.iloc[:,8]
x_testing['9']=pd.DataFrame(test_norm).iloc[:,4]
print(x_testing)
```

```
     0         1         2  3         4  5  6         7  8         9
198  0       NaN       NaN  1       NaN  1  0       NaN  1       NaN
37   0  0.262633  0.428336  2  0.243902  2  0  0.416667  1  0.074588
89   0       NaN       NaN  2       NaN  2  1       NaN  1       NaN
168  0       NaN       NaN  1       NaN  0  1       NaN  1       NaN
171  1       NaN       NaN  1       NaN  0  1       NaN  0       NaN
75   0       NaN       NaN  1       NaN  0  0       NaN  1       NaN
96   0       NaN       NaN  2       NaN  0  1       NaN  0       NaN
137  1       NaN       NaN  1       NaN  0  0       NaN  1       NaN
5    1  0.392756  0.411862  2  0.670732  2  1  0.500000  0  0.591829
83   1       NaN       NaN  2       NaN  2  1       NaN  0       NaN
55   1       NaN       NaN  2       NaN  0  0       NaN  1       NaN
145  1       NaN       NaN  2       NaN  2  0       NaN  1       NaN
160  1       NaN       NaN  2       NaN  2  1       NaN  1       NaN
112  1       NaN       NaN  1       NaN  0  0       NaN  1       NaN
74   1       NaN       NaN  1       NaN  0  0       NaN  0       NaN
203  1       NaN       NaN  1       NaN  0  0       NaN  1       NaN
126  0       NaN       NaN  2       NaN  2  1       NaN  0       NaN
12   0  1.000000  0.609555  2  0.365854  0  0  0.520833  1  0.790105
```

In [24]:

```
print(x_testing.isnull().sum()* 100 / len(x_testing))
```

```
0     0.00000
1    83.72093
2    83.72093
3     0.00000
4    83.72093
5     0.00000
6     0.00000
7    83.72093
8     0.00000
9    83.72093
dtype: float64
```

In [25]:

```
print(x_testing.index[x_testing.isnull().any(1)].tolist())
x_testing.dropna(axis=0,inplace=True)
print(x_testing)
```

```
[198, 89, 168, 171, 75, 96, 137, 83, 55, 145, 160, 112, 74, 203, 126, 153, 1
58, 169, 141, 209, 190, 144, 185, 86, 71, 63, 143, 97, 136, 162, 154, 90, 21
1, 106, 181, 139]
    0         1         2  3         4  5  6         7  8         9
37  0  0.262633  0.428336  2  0.243902  2  0  0.416667  1  0.074588
5   1  0.392756  0.411862  2  0.670732  2  1  0.500000  0  0.591829
12  0  1.000000  0.609555  2  0.365854  0  0  0.520833  1  0.790105
18  0  0.175884  0.362438  1  0.365854  0  0  0.750000  1  0.422414
15  0  0.320755  0.400824  1  0.167561  0  1  0.333333  0  0.265742
7   1  0.566255  0.428336  2  0.536585  2  1  0.125000  0  0.344828
33  0  0.642160  0.420099  2  0.268293  0  1  0.914792  0  0.667916
```

In [26]:

```
y_testing=y_test.drop([198, 89, 168, 171, 75, 96, 137, 83, 55, 145, 160, 112, 74, 203, 126,
print(y_testing)
```

```
37    1
5     0
12    0
18    0
15    1
7     1
33    1
Name: status, dtype: int32
```

Prediction using Logistic Regression

In [27]:

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(x_training, y_training)
LR.score(x_testing, y_testing)
```

Out[27]:

```
0.7142857142857143
```

In [28]:

```
a=[[1,70,91,0,60,2,0,70,1,58.80]]
LR.predict(a)
```

```
C:\Users\SAKSHI NEERAJ\anaconda3\lib\site-packages\sklearn\base.py:450: User
Warning: X does not have valid feature names, but LogisticRegression was fit
ted with feature names
  warnings.warn(
```

Out[28]:

```
array([1])
```

In [29]:

```
from sklearn.metrics import confusion_matrix
print(LR.predict(x_testing))
print(confusion_matrix(y_testing,LR.predict(x_testing)))
```

```
[1 1 1 0 1 1 1]
[[1 2]
 [0 4]]
```