In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("11-4-Dataset-Predicting Placement in Campus Recruitment.csv")
```

In [3]:

```
df.head(10)
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No
5	6	M	55.00	Others	49.80	Others	Science	67.25	Sci&Tech	Yes
6	7	F	46.00	Others	49.20	Others	Commerce	79.00	Comm&Mgmt	No
7	8	M	82.00	Central	64.00	Central	Science	66.00	Sci&Tech	Yes
8	9	M	73.00	Central	79.00	Central	Commerce	72.00	Comm&Mgmt	No
9	10	M	58.00	Central	70.00	Central	Commerce	61.00	Comm&Mgmt	No

In [4]:

```
df.drop("sl_no",axis = 1)
```

Out[4]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_
0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55
1	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86
2	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75
3	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66
4	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96
...	...	...	...	...	...	...	...	...	...	
210	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91
211	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74
212	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59
213	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70
214	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89

215 rows × 14 columns

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
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
```

In [6]:

```
df["salary"] = df["salary"].fillna(df["salary"].median())
```

In [12]:

```
df.isnull().sum()
```

Out[12]:

```
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 [8]:

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['gender','ssc_b','hsc_b','hsc_s','degree_t','workex','specialisation','status']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
```

In [37]:

```
df.head(10)
```

Out[37]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p
0	1	1	67.00	1	91.00	1	1	58.00	2	0	55.00
1	2	1	79.33	0	78.33	1	2	77.48	2	1	86.50
2	3	1	65.00	0	68.00	0	0	64.00	0	0	75.00
3	4	1	56.00	0	52.00	0	2	52.00	2	0	66.00
4	5	1	85.80	0	73.60	0	1	73.30	0	0	96.80
5	6	1	55.00	1	49.80	1	2	67.25	2	1	55.00
6	7	0	46.00	1	49.20	1	1	79.00	0	0	74.28
7	8	1	82.00	0	64.00	0	2	66.00	2	1	67.00
8	9	1	73.00	0	79.00	0	1	72.00	0	0	91.34
9	10	1	58.00	0	70.00	0	1	61.00	0	0	54.00

In [11]:

```
df.describe()
```

Out[11]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	d
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	21
mean	108.000000	0.646512	67.303395	0.460465	66.333163	0.609302	1.372093	6
std	62.209324	0.479168	10.827205	0.499598	10.897509	0.489045	0.580978	7
min	1.000000	0.000000	40.890000	0.000000	37.000000	0.000000	0.000000	5
25%	54.500000	0.000000	60.600000	0.000000	60.900000	0.000000	1.000000	6
50%	108.000000	1.000000	67.000000	0.000000	65.000000	1.000000	1.000000	6
75%	161.500000	1.000000	75.700000	1.000000	73.000000	1.000000	2.000000	7
max	215.000000	1.000000	89.400000	1.000000	97.700000	1.000000	2.000000	9

In [14]:

```
#so here mean==median varies for most of the columns except
```

In [15]:

```
df.drop("sl_no",axis=1)
```

Out[15]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	speci
0	1	67.00	1	91.00	1	1	58.00	2	0	55.0	
1	1	79.33	0	78.33	1	2	77.48	2	1	86.5	
2	1	65.00	0	68.00	0	0	64.00	0	0	75.0	
3	1	56.00	0	52.00	0	2	52.00	2	0	66.0	
4	1	85.80	0	73.60	0	1	73.30	0	0	96.8	
...	...	...	...	...	...	...	...	...	...	...	
210	1	80.60	1	82.00	1	1	77.60	0	0	91.0	
211	1	58.00	1	60.00	1	2	72.00	2	0	74.0	
212	1	67.00	1	67.00	1	1	73.00	0	1	59.0	
213	0	74.00	1	66.00	1	1	58.00	0	0	70.0	
214	1	62.00	0	58.00	1	2	53.00	0	0	89.0	

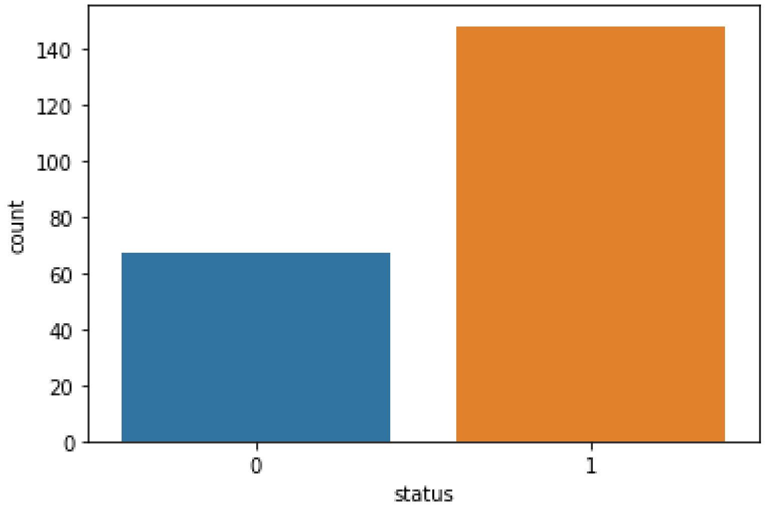
215 rows × 14 columns

In [16]:

```
sns.countplot(x='status',data=df)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x26e08d0eb08>
```



In [17]:

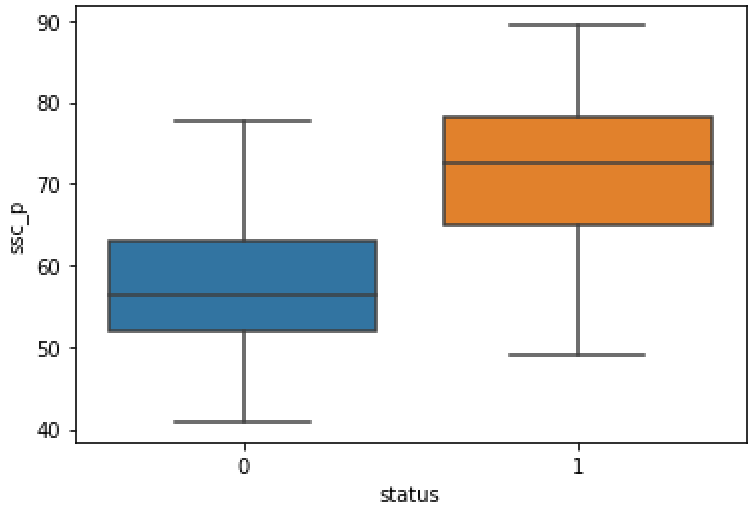
```
#no of placed students is more than not placed students
```

In [23]:

```
sns.boxplot(x='status',y='ssc_p',data=df)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x26e0934f488>
```



In [24]:

```
#50% students are not placed having SSC percentage between 41% to 56%
#50% students are not placed having SSC percentage between 57% to 78%
#not placed max = 78% and min= 41%

#50% students are placed having SSC percentage between 50% to 72%
#50% students are placed having SSC percentage between 73% to 90%
# placed max =90% and min 50%
```

In [28]:

```
X = df.drop(['status'],axis = 1)
y = df['status']
```

In [29]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.8,random_state = 0)
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
```

In [31]:

```
sc= StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

SVM

In [32]:

```
from sklearn.svm import SVC
my_model = SVC(kernel = 'rbf', random_state = 0)
result = my_model.fit(X_train,y_train)
```

In [34]:

```
predictions = result.predict(X_test)
predictions
```

Out[34]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1])
```

In [35]:

```
from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,predictions))
```

```
Accuracy :  0.7616279069767442
```

In [36]:

```
import seaborn as sn
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(predictions,y_test)
confusion_df = pd.DataFrame(conf_matrix,index=['Actual 0','Actual 1'],columns = ['Predictec
confusion_df
```

Out[36]:

	Predicted 0	Predicted 1
Actual 0	18	5
Actual 1	36	113

In [52]:

```
from sklearn import metrics
print("\n**classification report :\n",metrics.classification_report(y_test,predictions))
```

```

**classification report :
               precision    recall  f1-score   support

           0       0.78      0.33      0.47        54
           1       0.76      0.96      0.85       118

    accuracy                           0.76       172
   macro avg       0.77      0.65      0.66       172
weighted avg       0.77      0.76      0.73       172

```

In [39]:

```
new_pred = list(result.predict([[0,89.40,0,64.31,0,2,66.40,0,0,80.00,0,74.00,1,200000]]))
new_pred
```

Out[39]:

```
[1]
```

Logistic regression

In [42]:

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

In [43]:

```
my_model_1 = LogisticRegression()
```

In [44]:

```
result_1 = my_model_1.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

In [46]:

```
predictions_1 = result_1.predict(X_test)
predictions_1
```

Out[46]:

```
array([0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1])
```

In [47]:

```
from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,predictions_1))
```

```
Accuracy :  0.8313953488372093
```

In [48]:

```
import seaborn as sn
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(predictions_1,y_test)
confusion_df = pd.DataFrame(conf_matrix,index=['Actual 0','Actual 1'],columns = ['Predicted
confusion_df
```

Out[48]:

	Predicted 0	Predicted 1
Actual 0	42	17
Actual 1	12	101

In [67]:

```
new_pred = list(result_1.predict([[0,89.40,0,64.31,0,2,66.40,0,0,80.00,0,74.00,1,200000]]))
new_pred
```

Out[67]:

```
[1]
```

In [53]:

```
from sklearn import metrics
print("\n**classification report :\n",metrics.classification_report(y_test,predictions_1))
```

```
**classification report :
               precision    recall  f1-score   support

           0       0.71      0.78      0.74        54
           1       0.89      0.86      0.87       118

    accuracy                           0.83       172
   macro avg       0.80      0.82      0.81       172
weighted avg       0.84      0.83      0.83       172
```

Decision Tree

In [55]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [56]:

```
my_model_2 = DecisionTreeClassifier(random_state = 0)
result_2 = my_model_2.fit(X_train,y_train)
```

In [61]:

```
predictions_2 = result_2.predict(X_test)
predictions_2
```

Out[61]:

```
array([1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1])
```

In [63]:

```
from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,predictions_2))
```

```
Accuracy :  0.8313953488372093
```

In [64]:

```
import seaborn as sn
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(predictions_2,y_test)
confusion_df = pd.DataFrame(conf_matrix,index=['Actual 0','Actual 1'],columns = ['Predicted
confusion_df
```

Out[64]:

	Predicted 0	Predicted 1
Actual 0	44	19
Actual 1	10	99

In [66]:

```
from sklearn import metrics
print("\n**classification report :\n",metrics.classification_report(y_test,predictions_2))
```

```

**classification report :
               precision    recall  f1-score   support

           0       0.70      0.81      0.75        54
           1       0.91      0.84      0.87       118

    accuracy                           0.83       172
   macro avg       0.80      0.83      0.81       172
weighted avg       0.84      0.83      0.83       172

```

In [68]:

```
new_pred = list(result_2.predict([[0,89.40,0,64.31,0,2,66.40,0,0,80.00,0,74.00,1,200000]]))
new_pred
```

Out[68]:

```
[0]
```

Random Forest

In [69]:

```
from sklearn.ensemble import RandomForestClassifier
my_model_3 = RandomForestClassifier(n_estimators = 20,criterion = 'entropy',random_state=42
```

In [70]:

```
result_3 = my_model_3.fit(X_train,y_train)
```

In [72]:

```
predictions_3 = result_3.predict(X_test)
predictions_3
```

Out[72]:

```
array([1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1])
```

In [73]:

```
from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,predictions_3))
```

```
Accuracy :  0.8430232558139535
```

In [74]:

```
import seaborn as sn
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(predictions_3,y_test)
confusion_df = pd.DataFrame(conf_matrix,index=['Actual 0','Actual 1'],columns = ['Predictec
confusion_df
```

Out[74]:

	Predicted 0	Predicted 1
Actual 0	47	20
Actual 1	7	98

In [75]:

```
from sklearn import metrics
print("\n**classification report :\n",metrics.classification_report(y_test,predictions_3))
```

```

**classification report :
               precision    recall  f1-score   support

           0       0.70      0.87      0.78        54
           1       0.93      0.83      0.88       118

    accuracy                           0.84       172
   macro avg       0.82      0.85      0.83       172
weighted avg       0.86      0.84      0.85       172
```

In [76]:

```
new_pred = list(result_3.predict([[0,89.40,0,64.31,0,2,66.40,0,0,80.00,0,74.00,1,200000]]))
new_pred
```

Out[76]:

```
[1]
```

KNN

In [78]:

```
from sklearn.neighbors import KNeighborsClassifier
my_model_4 = KNeighborsClassifier(n_neighbors = 3)
result_4 = my_model_4.fit(X_train,y_train)
```

In [80]:

```
predictions_4 = result_4.predict(X_test)
predictions_4
```

Out[80]:

```
array([0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1])
```

In [89]:

```
print("Accuracy : ",result.score(X_test,y_test))
```

```
Accuracy :  0.7616279069767442
```

In [84]:

```
import seaborn as sn
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(predictions_4,y_test)
confusion_df = pd.DataFrame(conf_matrix,index=['Actual 0','Actual 1'],columns = ['Predictec
confusion_df
```

Out[84]:

	Predicted 0	Predicted 1
Actual 0	30	17
Actual 1	24	101

In [86]:

```
from sklearn import metrics
print("\n**classification report :\n",metrics.classification_report(y_test,predictions_4))
```

```
**classification report :
               precision    recall  f1-score   support

           0       0.64      0.56      0.59        54
           1       0.81      0.86      0.83       118

    accuracy                           0.76       172
   macro avg       0.72      0.71      0.71       172
weighted avg       0.75      0.76      0.76       172
```

In [90]:

```
new_pred = list(result_4.predict([[0,89.40,0,64.31,0,2,66.40,0,0,80.00,0,74.00,1,200000]]))
new_pred
```

Out[90]:

```
[1]
```

In [ ]: