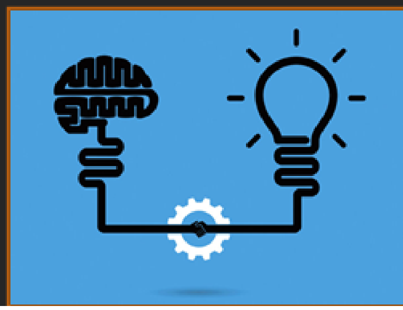
Placement Prediction Using Machine Learning.

Guide :- Prof. Ashwini Barbadekar Ma'am

Name : Chinmay Jayendra Deotale
Div :-ET - A
GR. No. :-11910759
Batch :-2
Roll No. :-60

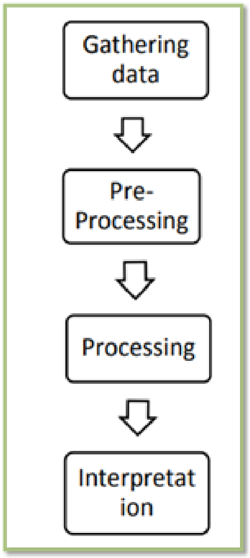


TASK 1 : UNDERSTANDING PROBLEM STATEMENT :

Problem Statement & Objective

- Placements hold great importance for students to build a strong foundation for the professional career ahead as well as a good placement record gives a competitive edge to a college/university in the education market.
- To develop a placement predictor as a part of making a placement management system at college level which predicts the chances of students getting placed and helps in uplifting their skills before the recruitment process starts.
- A placement predictor is to be designed to calculate the possibility of a student being placed in a company, subject to the criterion of the company. The placement predictor takes many parameters which can be used to assess the skill level of the student. While some parameters are taken from the university level, others are obtained from tests conducted in the placement management system itself. Combining these data points, the predictor is to accurately predict if the student will or will not be placed in a company. Data from past students are used for training the predictor.

METHODOLOGY :



1) Data gathering : The sample data has been collected from KAGGLE.com which consists of all the records of student placement through various sources.
2) Pre-Processing Data : Pre processing is a technique that is used to convert raw data into a clean dataset. The data is gathered from different sources is in raw format which is not feasible for the analysis.
   a) Attribute selection
   b) Cleaning missing values
   c) Training and Test data
   d) Feature Scaling
3) Processing : Processing is applying different algorithms to the data to find the best results
   a) Logistic Regression
   b) Naïve Bayes Classifier
   c) SVM

TASK 2 : IMPORT LIBRARIES AND DATASET :

LIBRARIES USED

PANDAS

MATPLOTLIB

SEABORN

SKLEARN

FLASK

PICKLE

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix, classification_repo
```

In [2]:
```
dataset = pd.read_csv('./Placement_Data_Full_Class.csv')
```

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

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etes
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	5
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	8
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	7
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	6
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	9

In [4]:
```
# as salary and sl_no columns are not required for placement status prediction so
dataset.drop(['salary','sl_no'], axis=1, inplace=True)
```

TASK 3: PERFORM EXPLORATORY DATA ANALYSIS

In [5]:
```
# missing values checking
dataset.isnull().sum()
```

Out[5]:
```
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
dtype: int64
```

In [6]:
```
# checking column values data type
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   gender          215 non-null    object 
 1   ssc_p           215 non-null    float64
 2   ssc_b           215 non-null    object 
 3   hsc_p           215 non-null    float64
 4   hsc_b           215 non-null    object 
 5   hsc_s           215 non-null    object 
 6   degree_p        215 non-null    float64
 7   degree_t        215 non-null    object 
 8   workex          215 non-null    object 
 9   etest_p         215 non-null    float64
 10  specialisation  215 non-null    object 
 11  mba_p           215 non-null    float64
 12  status          215 non-null    object 
dtypes: float64(5), object(8)
memory usage: 22.0+ KB
```

A) Label Encoding Data

In [7]:
```
# label encoding needs to be done to ensure all values in the dataset is numeric
# hsc_s, degree_t columns needs to be splitted into columns (get_dummies needs to
features_to_split = ['hsc_s','degree_t']
for feature in features_to_split:
    dummy = pd.get_dummies(dataset[feature])
    dataset = pd.concat([dataset, dummy], axis=1)
    dataset.drop(feature, axis=1, inplace=True)
```

In [8]:
```
dataset
```

Out[8]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	workex	etest_p	specialisation	mba_p	st
0	M	67.00	Others	91.00	Others	58.00	No	55.0	Mkt&HR	58.80	Pl
1	M	79.33	Central	78.33	Others	77.48	Yes	86.5	Mkt&Fin	66.28	Pl
2	M	65.00	Central	68.00	Central	64.00	No	75.0	Mkt&Fin	57.80	Pl
3	M	56.00	Central	52.00	Central	52.00	No	66.0	Mkt&HR	59.43	Pl
4	M	85.80	Central	73.60	Central	73.30	No	96.8	Mkt&Fin	55.50	Pl
...	...	...	...	...	...	...	...	...	...	...	
210	M	80.60	Others	82.00	Others	77.60	No	91.0	Mkt&Fin	74.49	Pl
211	M	58.00	Others	60.00	Others	72.00	No	74.0	Mkt&Fin	53.62	Pl
212	M	67.00	Others	67.00	Others	73.00	Yes	59.0	Mkt&Fin	69.72	Pl
213	F	74.00	Others	66.00	Others	58.00	No	70.0	Mkt&HR	60.23	Pl
214	M	62.00	Central	58.00	Others	53.00	No	89.0	Mkt&HR	60.22	Pl

215 rows × 17 columns

In [9]:
```
dataset.rename(columns={"Others": "Other_Degree"},inplace=True)
```

In [10]:
```
dataset
```

Out[10]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	workex	etest_p	specialisation	mba_p	st
0	M	67.00	Others	91.00	Others	58.00	No	55.0	Mkt&HR	58.80	Pl
1	M	79.33	Central	78.33	Others	77.48	Yes	86.5	Mkt&Fin	66.28	Pl
2	M	65.00	Central	68.00	Central	64.00	No	75.0	Mkt&Fin	57.80	Pl
3	M	56.00	Central	52.00	Central	52.00	No	66.0	Mkt&HR	59.43	Pl
4	M	85.80	Central	73.60	Central	73.30	No	96.8	Mkt&Fin	55.50	Pl
...	...	...	...	...	...	...	...	...	...	...	
210	M	80.60	Others	82.00	Others	77.60	No	91.0	Mkt&Fin	74.49	Pl
211	M	58.00	Others	60.00	Others	72.00	No	74.0	Mkt&Fin	53.62	Pl
212	M	67.00	Others	67.00	Others	73.00	Yes	59.0	Mkt&Fin	69.72	Pl
213	F	74.00	Others	66.00	Others	58.00	No	70.0	Mkt&HR	60.23	Pl
214	M	62.00	Central	58.00	Others	53.00	No	89.0	Mkt&HR	60.22	Pl

215 rows × 17 columns

In [11]:
```
encoder = LabelEncoder() # to encode string to the values like 0,1,2 etc.
```

In [12]:
```
columns_to_encode = ['gender','ssc_b', 'hsc_b','workex','specialisation','status'
for column in columns_to_encode:
    dataset[column] = encoder.fit_transform(dataset[column])
```

In [13]:
```
dataset
```

Out[13]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	workex	etest_p	specialisation	mba_p	stat
0	1	67.00	1	91.00	1	58.00	0	55.0	1	58.80	
1	1	79.33	0	78.33	1	77.48	1	86.5	0	66.28	
2	1	65.00	0	68.00	0	64.00	0	75.0	0	57.80	
3	1	56.00	0	52.00	0	52.00	0	66.0	1	59.43	
4	1	85.80	0	73.60	0	73.30	0	96.8	0	55.50	
...	...	...	...	...	...	...	...	...	...	...	
210	1	80.60	1	82.00	1	77.60	0	91.0	0	74.49	
211	1	58.00	1	60.00	1	72.00	0	74.0	0	53.62	
212	1	67.00	1	67.00	1	73.00	1	59.0	0	69.72	
213	0	74.00	1	66.00	1	58.00	0	70.0	1	60.23	
214	1	62.00	0	58.00	1	53.00	0	89.0	1	60.22	

215 rows × 17 columns

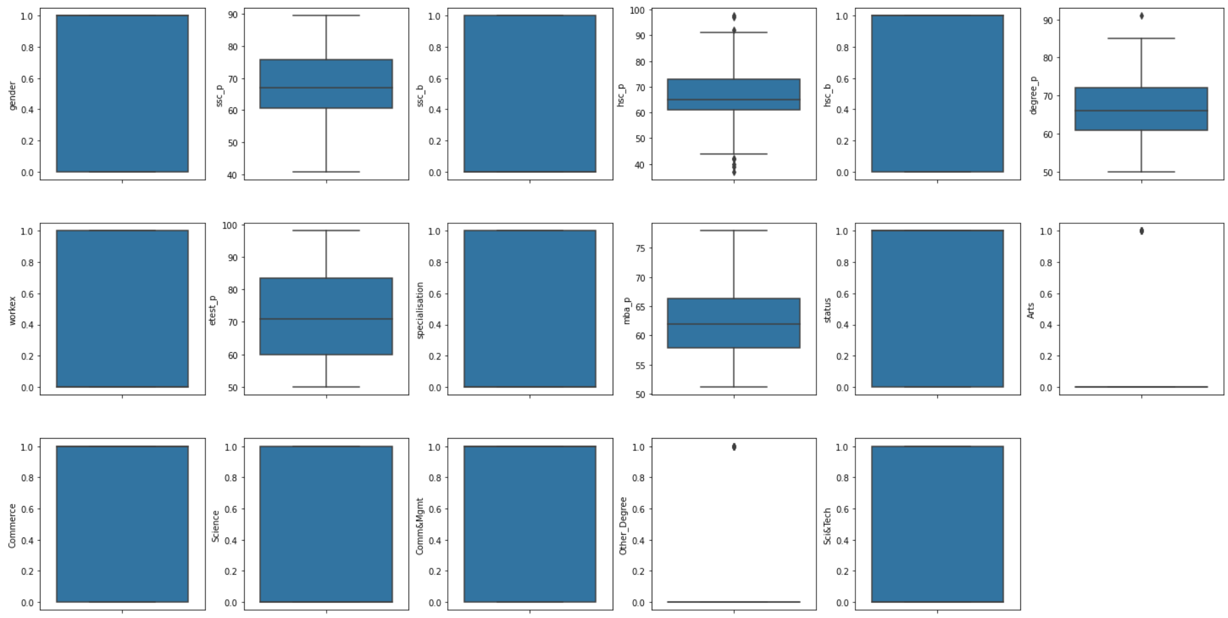
In [14]:
```
dataset.describe()
```

Out[14]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	workex	ete
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.00
mean	0.646512	67.303395	0.460465	66.333163	0.609302	66.370186	0.344186	72.10
std	0.479168	10.827205	0.499598	10.897509	0.489045	7.358743	0.476211	13.27
min	0.000000	40.890000	0.000000	37.000000	0.000000	50.000000	0.000000	50.00
25%	0.000000	60.600000	0.000000	60.900000	0.000000	61.000000	0.000000	60.00
50%	1.000000	67.000000	0.000000	65.000000	1.000000	66.000000	0.000000	71.00
75%	1.000000	75.700000	1.000000	73.000000	1.000000	72.000000	1.000000	83.50
max	1.000000	89.400000	1.000000	97.700000	1.000000	91.000000	1.000000	98.00

B) Checking for Outliers

In [15]:
```
fig, axs = plt.subplots(ncols=6,nrows=3,figsize=(20,10))
index = 0
axs = axs.flatten()
for k,v in dataset.items():
    sns.boxplot(y=v, ax=axs[index])
    index+=1

fig.delaxes(axs[index])
plt.tight_layout(pad=0.3, w_pad=0.5,h_pad = 4.5) # for styling by giving padding
```



In [16]:
```
# deleting some outliers in 2 columns degree_p and hsc_p
dataset = dataset[~(dataset['degree_p']>=90)]
dataset = dataset[~(dataset['hsc_p']>=95)]
```

C) Checking for Correlation

In [17]:
```
dataset.corr()
```

Out[17]:

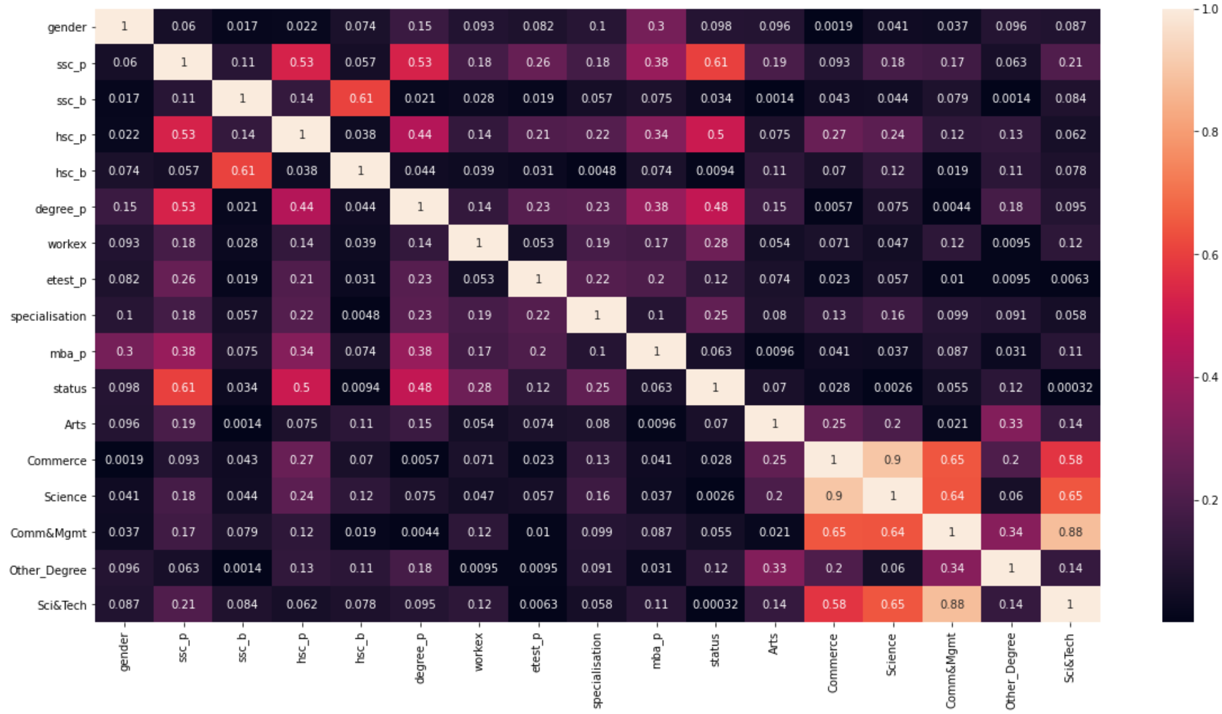
	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	workex	etest_p
gender	1.000000	-0.059818	0.017052	-0.022187	0.074438	-0.154679	0.093325	0.081765
ssc_p	-0.059818	1.000000	0.107995	0.528111	0.056672	0.528753	0.183073	0.264009
ssc_b	0.017052	0.107995	1.000000	-0.140332	0.608493	0.020828	-0.027916	-0.018739
hsc_p	-0.022187	0.528111	-0.140332	1.000000	-0.038259	0.443595	0.135144	0.208809
hsc_b	0.074438	0.056672	0.608493	-0.038259	1.000000	0.043618	0.039061	0.031316
degree_p	-0.154679	0.528753	0.020828	0.443595	0.043618	1.000000	0.135100	0.226353
workex	0.093325	0.183073	-0.027916	0.135144	0.039061	0.135100	1.000000	0.052862
etest_p	0.081765	0.264009	-0.018739	0.208809	0.031316	0.226353	0.052862	1.000000
specialisation	-0.103355	-0.177436	-0.057356	-0.222405	0.004762	-0.232618	-0.187200	-0.222765
mba_p	-0.298466	0.377438	0.074653	0.335610	0.073936	0.376261	0.174951	0.203663
status	0.098189	0.605381	0.033717	0.499777	0.009393	0.479557	0.279091	0.122770
Arts	-0.096386	-0.194514	-0.001410	-0.074931	-0.114855	-0.153777	0.054259	-0.073539
Commerce	0.001870	-0.093283	-0.042586	0.267073	-0.069985	-0.005676	-0.070916	-0.023192
Science	0.041426	0.181772	0.043708	-0.236466	0.122407	0.074850	0.047346	0.056508
Comm&Mgmt	-0.036801	-0.168282	-0.078842	0.121441	-0.019492	-0.004369	-0.118781	-0.010486
Other_Degree	-0.096386	-0.063459	-0.001410	-0.132137	-0.114855	-0.180476	0.009501	0.009482
Sci&Tech	0.086960	0.208907	0.083707	-0.061747	0.077977	0.094883	0.120296	0.006296

TASK 4: PERFORM DATA VISUALIZATION

Type *Markdown* and LaTeX: α^2

In [18]:
```
# heatmap for checking correlation or linearity

plt.figure(figsize=(20,10))
sns.heatmap(dataset.corr().abs(), annot=True)
```

Out[18]: <AxesSubplot:>



Correlation between the features are atmost 0.9 so they are not multi-correlated

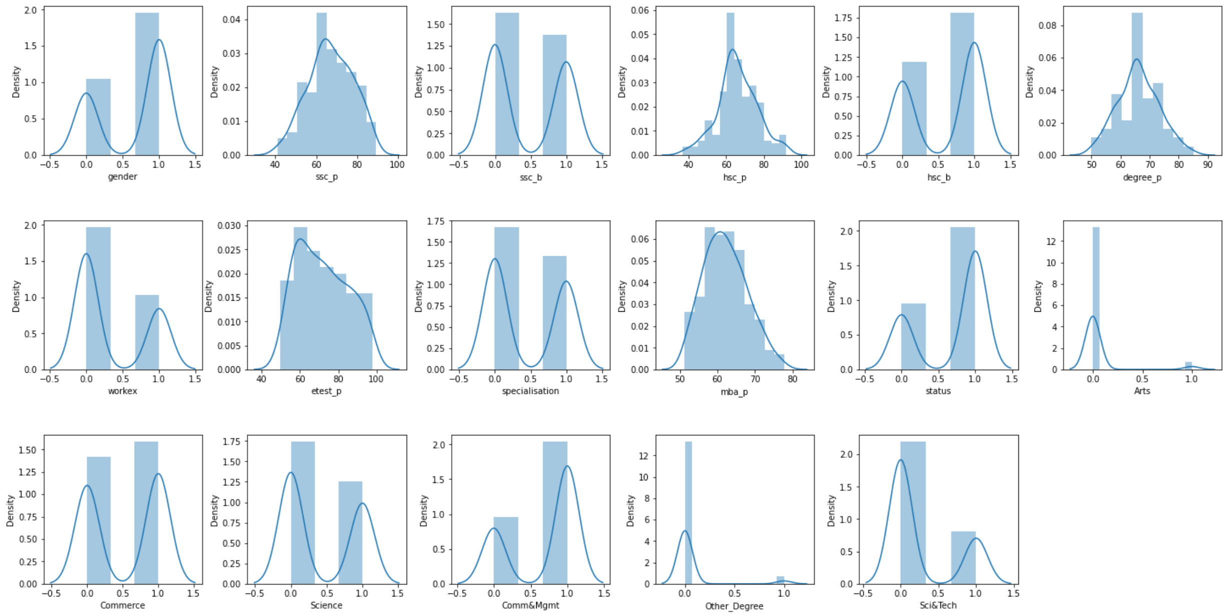
In [19]:
```
dataset.shape
```

Out[19]: (212, 17)

In [20]:
```
# checking distributions of all features
fig, axs = plt.subplots(ncols=6,nrows=3,figsize=(20,10))
index = 0
axs = axs.flatten()
for k,v in dataset.items():
    sns.distplot(v, ax=axs[index])
    index+=1

fig.delaxes(axs[index]) # deleting the 18th figure
plt.tight_layout(pad=0.3, w_pad=0.2,h_pad = 4.5)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
```

```
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```



TASK 5: CREATE TRAINING AND TESTING DATASET

In [21]:
```
x = dataset.loc[:,dataset.columns!='status'] # all features are used
y = dataset.loc[:, 'status'] # label is status of placement
```

In [22]:
```
x
```

Out[22]:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	workex	etest_p	specialisation	mba_p	Arts
0	1	67.00	1	91.00	1	58.00	0	55.0	1	58.80	(
1	1	79.33	0	78.33	1	77.48	1	86.5	0	66.28	(
2	1	65.00	0	68.00	0	64.00	0	75.0	0	57.80	'
3	1	56.00	0	52.00	0	52.00	0	66.0	1	59.43	(
4	1	85.80	0	73.60	0	73.30	0	96.8	0	55.50	(
...	...	...	...	...	...	...	...	...	...	...	..
210	1	80.60	1	82.00	1	77.60	0	91.0	0	74.49	(
211	1	58.00	1	60.00	1	72.00	0	74.0	0	53.62	(
212	1	67.00	1	67.00	1	73.00	1	59.0	0	69.72	(
213	0	74.00	1	66.00	1	58.00	0	70.0	1	60.23	(
214	1	62.00	0	58.00	1	53.00	0	89.0	1	60.22	(

212 rows × 16 columns

In [23]:
```
y
```

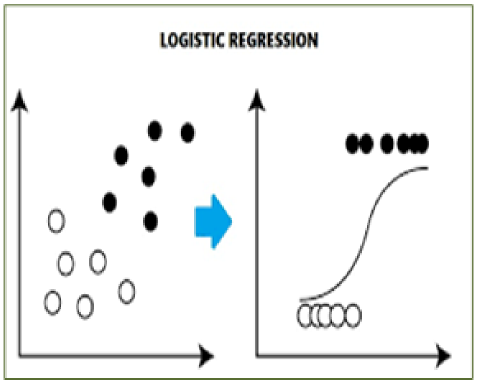
Out[23]:
```
0      1
1      1
2      1
3      0
4      1
      ..
210    1
211    1
212    1
213    1
214    0
Name: status, Length: 212, dtype: int32
```

In [24]:
```
sc= StandardScaler()
x_scaled = sc.fit_transform(x) # for standardising the features
x_scaled = pd.DataFrame(x_scaled)
```

In [25]:
```
x_train,x_test, y_train, y_test = train_test_split(x_scaled,y,test_size=0.18, rar
```

TASK 6: TRAIN AND EVALUATE A LINEAR REGRESSION MODEL

LOGISTIC REGRESSION MODEL



- Logistic regression is a classification technique and it is very good for binary classification.
- The goal of this technique is given a new data point, and predict the class from which the data point is likely to have originated. Input features can be quantitative or qualitative.
- Instead of a hyperplane or straight line, the logistic regression uses the logistic function to obtain the output of a linear equation between 0 and 1.
- The function is defined as logistic(x)=1/(1+exp(-x))

In [26]:
```
lr = LogisticRegression()
```

In [27]:
```
lr.fit(x_train, y_train)
```

Out[27]:
```
LogisticRegression()
```

In [28]:
```
y_pred = lr.predict(x_test)
```

In [29]:
```
y_pred
```

Out[29]:
```
array([1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0])
```

In [30]:
```
y_test
```

Out[30]:
```
209    1
38     1
90     1
192    1
150    1
76     1
97     0
138    1
5      0
84     1
56     1
144    0
159    0
113    1
75     0
203    1
127    1
12     0
169    0
157    1
167    0
201    0
211    1
189    0
184    0
18     0
214    0
15     1
87     0
72     1
7      1
64     1
142    1
98     1
137    1
161    0
34     0
153    1
91     0
Name: status, dtype: int32
```

In [31]:
```
accuracy_score(y_test, y_pred)
```

Out[31]:
```
0.8717948717948718
```

In [32]:
```
lr.score(x_train,y_train)
```

Out[32]:
```
0.9132947976878613
```

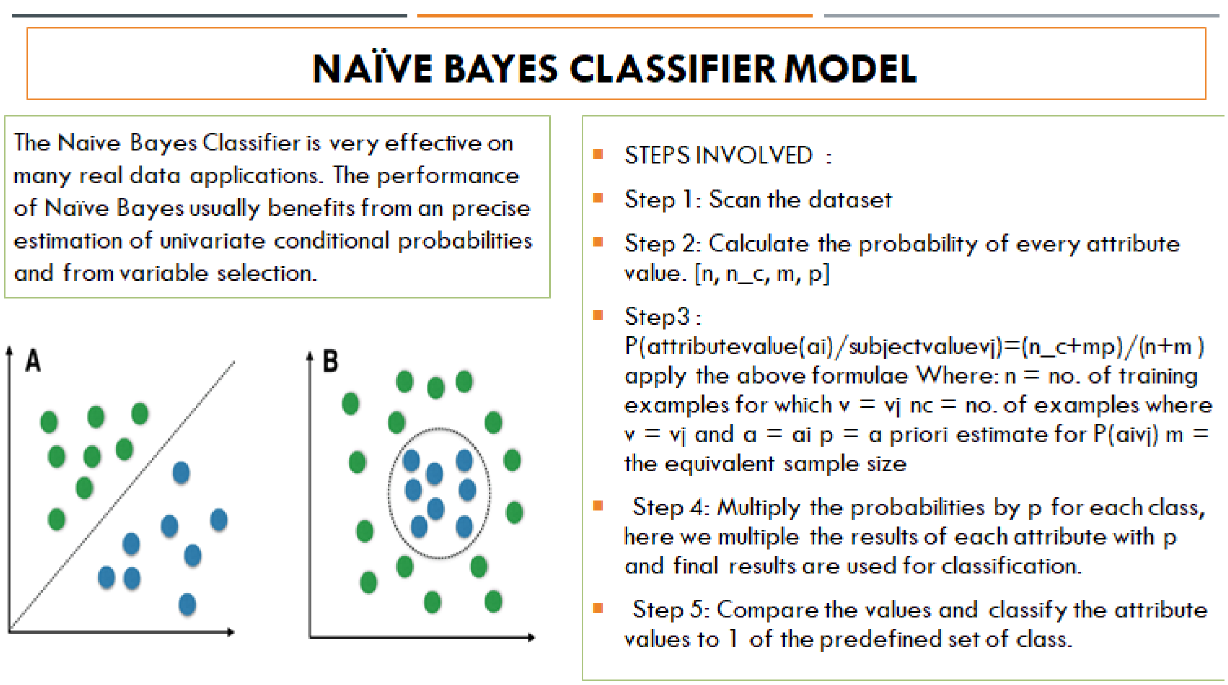
In [33]:
```
confusion_matrix(y_test, y_pred)
```

Out[33]:
```
array([[14,  3],
       [ 2, 20]], dtype=int64)
```

In [34]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85        17
           1       0.87      0.91      0.89        22

    accuracy                           0.87        39
   macro avg       0.87      0.87      0.87        39
weighted avg       0.87      0.87      0.87        39
```

TASK 7: TRAIN AND EVALUATE Naive Bayes Classifier :



In [35]:
```
nbclassifier = GaussianNB()
```

In [36]:
```
nbclassifier.fit(x_train, y_train)
```

Out[36]:
```
GaussianNB()
```

In [37]:
```
y_pred_nb = nbclassifier.predict(x_test)
```

In [38]:
```
accuracy_score(y_test, y_pred_nb)
```

Out[38]:
```
0.8461538461538461
```

In [39]:
```
nbclassifier.score(x_train, y_train)
```

Out[39]:
```
0.8554913294797688
```

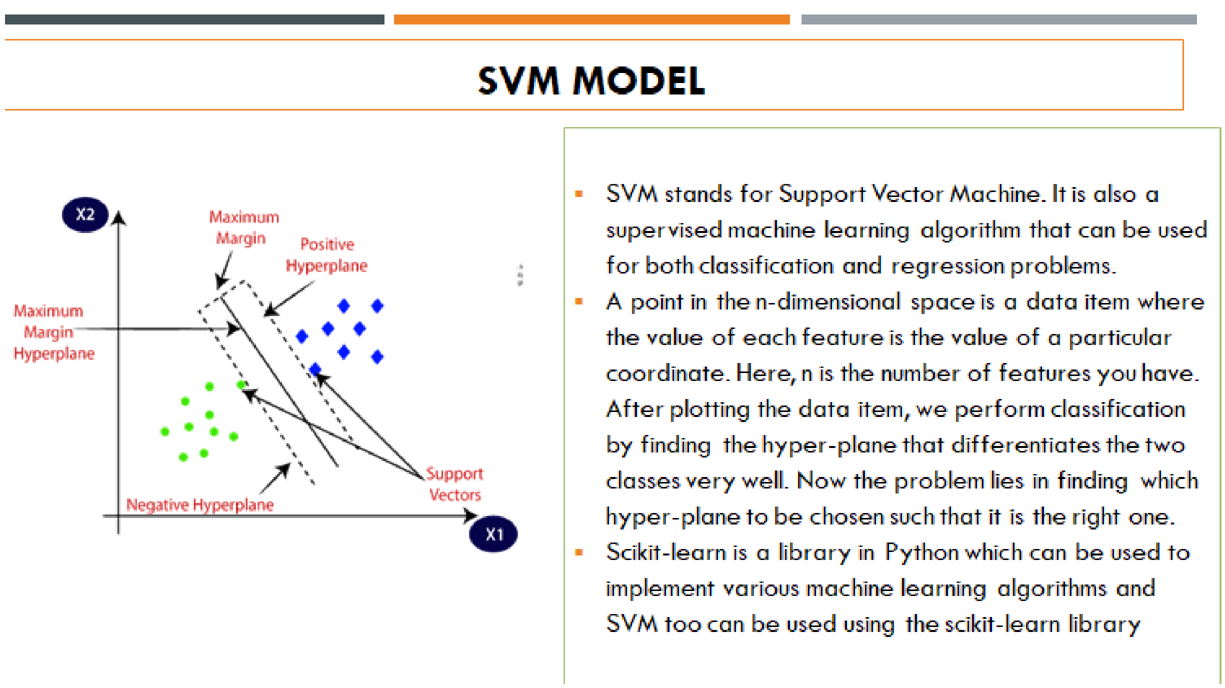
In [40]:
```
confusion_matrix(y_test, y_pred_nb)
```

Out[40]:
```
array([[13,  4],
       [ 2, 20]], dtype=int64)
```

In [41]:
```
print(classification_report(y_test,y_pred_nb))
```

```
              precision    recall  f1-score   support

           0       0.87      0.76      0.81        17
           1       0.83      0.91      0.87        22

    accuracy                           0.85        39
   macro avg       0.85      0.84      0.84        39
weighted avg       0.85      0.85      0.84        39
```

TASK 8: TRAIN AND EVALUATE SVM :



In [42]:
```
clf = svm.SVC(kernel="linear")
```

In [43]:
```
clf.fit(x_train, y_train)
```
Out[43]:
```
SVC(kernel='linear')
```

In [44]:
```
y_pred_svm = clf.predict(x_test)
```

In [45]:
```
accuracy_score(y_test, y_pred_svm)
```
Out[45]:
```
0.8974358974358975
```

In [46]:
```
clf.score(x_train, y_train)
```
Out[46]:
```
0.9017341040462428
```

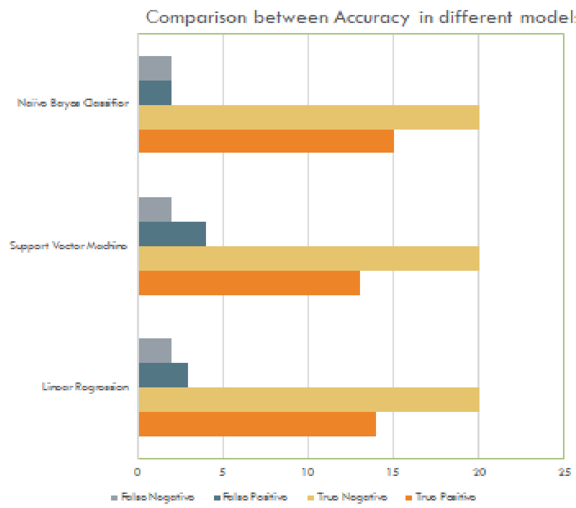
In [47]:
```
confusion_matrix(y_test, y_pred_svm)
```
Out[47]:
```
array([[15,  2],
       [ 2, 20]], dtype=int64)
```

In [48]:
```
print(classification_report(y_test, y_pred_svm))
```
```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88        17
           1       0.91      0.91      0.91        22

    accuracy                           0.90        39
   macro avg       0.90      0.90      0.90        39
weighted avg       0.90      0.90      0.90        39
```

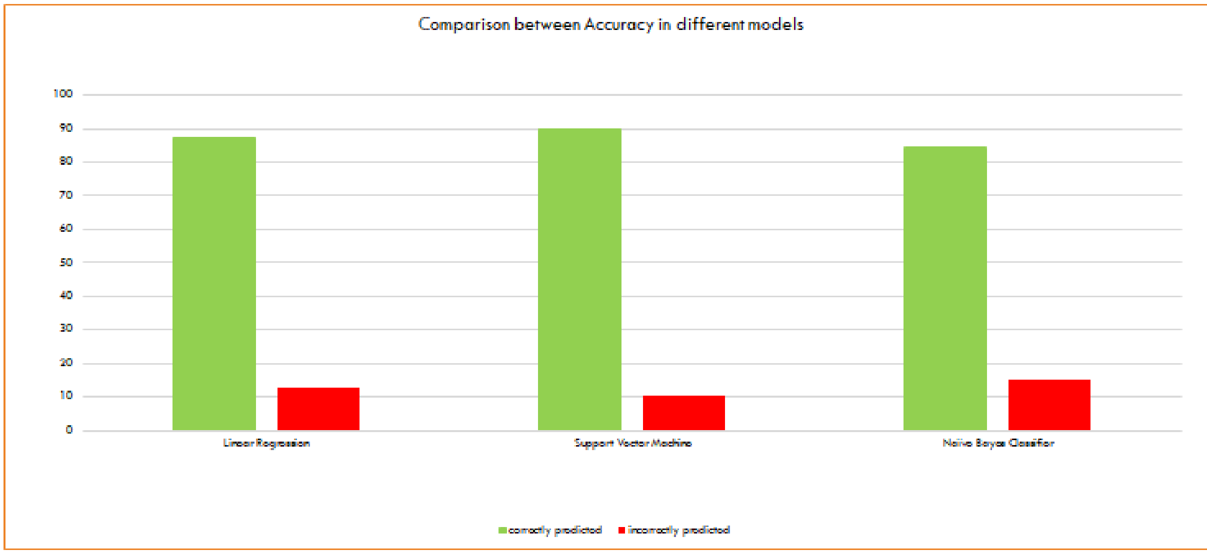
RESULTS AND CONCLUSION

RESULTS

- The data is first trained and then tested with all Three algorithms and out of all SVM gave more accuracy with **89.7435**, Logistic regression with **87.18** percent accuracy and Naïve bayes with accuracy of **85.6.**
- We conclude that Logistic Regression works better with better accuracy but difference in scores is highest among three
- Gaussian Naive Bayes was less accurate but the difference in known and unknown data was lesser.
- But, SVM gave better accuracy with least difference in score. So, Our final model would use SVM for Student Placement Prediction.



RESULTS



In [ ]: