```
Today Agenda
- Logistic Regression
- SVM (Support Vector Machine)
```

Logistic Regression

In [1]:

```
#1.read the data
import pandas as pd
# read the .txt file by using read_csv
data = pd.read_table("adminsheet.txt")
print(data)
```

```
C:\Users\RANGA\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureWa
rning: read_table is deprecated, use read_csv instead, passing sep='\t'.
  This is separate from the ipykernel package so we can avoid doing imports
until

     34.62365962451697,78.0246928153624,0
0   30.28671076822607,43.89499752400101,0
1   35.84740876993872,72.90219802708364,0
2   60.18259938620976,86.30855209546826,1
3     79.0327360507101,75.3443764369103,1
4    45.08327747668339,56.3163717815305,0
5   61.10666453684766,96.51142588489624,1
6   75.02474556738889,46.55401354116538,1
7   76.09878670226257,87.42056971926803,1
8   84.43281996120035,43.53339331072109,1
9   95.86155507093572,38.22527805795094,0
10  75.01365838958247,30.60326323428011,0
11  82.30705337399482,76.48196330235604,1
12  69.36458875970939,97.71869196188608,1
13  39.53833914367223,76.03681085115882,0
14   53.9710521485623,89.20735013750205,1
15  69.07014406283025,52.74046973016765,1
16  67.94685547711617,46.67857410673128,0
17  70.66150955499435,92.92713789364831,1
18  76.97878372747498,47.57596364975532,1
19  67.37202754570876,42.83843832029179,0
20  89.67677575072079,65.79936592745237,1
21    50.534788289883,48.85581152764205,0
22  34.21206097786789,44.20952859866288,0
23    77.9240914545704,68.9723599933059,1
24  62.27101367004632,69.95445795447587,1
25   80.1901807509566,44.82162893218353,1
26    93.114388797442,38.80067033713209,0
27  61.83020602312595,50.25610789244621,0
28  38.78580379679423,64.99568095539578,0
29    61.379289447425,72.80788731317097,1
..                                    ...
69  32.72283304060323,43.30717306430063,0
70   64.0393204150601,78.03168802018232,1
71  72.34649422579923,96.22759296761404,1
72  60.45788573918959,73.09499809758037,1
73  58.84095621726802,75.85844831279042,1
74  99.82785779692128,72.36925193383885,1
75  47.26426910848174,88.47586499559782,1
76  50.45815980285988,75.80985952982456,1
77  60.45555629271532,42.50840943572217,0
78  82.22666157785568,42.71987853716458,0
79   88.9138964166533,69.80378889835472,1
80  94.83450672430196,45.69430680250754,1
81  67.31925746917527,66.58935317747915,1
82  57.23870631569862,59.51428198012956,1
83  80.36675600171273,90.96014789746954,1
84  68.46852178591112,85.59430710452014,1
```

```
85   42.0754545384731,78.84478600148043,0
86  75.47770200533905,90.42453899753964,1
87  78.63542434898018,96.64742716885644,1
88  52.34800398794107,60.76950525602592,0
89  94.09433112516793,77.15910509073893,1
90  90.44855097096364,87.50879176484702,1
91  55.48216114069585,35.57070347228866,0
92  74.49269241843041,84.84513684930135,1
93  89.84580670720979,45.35828361091658,1
94  83.48916274498238,48.38028579728175,1
95   42.2617008099817,87.10385094025457,1
96  99.31500880510394,68.77540947206617,1
97   55.34001756003703,64.9319380069486,1
98  74.77589300092767,89.52981289513276,1

[99 rows x 1 columns]
```

In [3]:

```
data1 = pd.read_csv("adminsheet.txt",sep=",",header=None)
data1
```

Out[3]:

	0	1	2
0	34.623660	78.024693	0
1	30.286711	43.894998	0
2	35.847409	72.902198	0
3	60.182599	86.308552	1
4	79.032736	75.344376	1
5	45.083277	56.316372	0
6	61.106665	96.511426	1
7	75.024746	46.554014	1
8	76.098787	87.420570	1
9	84.432820	43.533393	1
10	95.861555	38.225278	0
11	75.013658	30.603263	0
12	82.307053	76.481963	1
13	69.364589	97.718692	1
14	39.538339	76.036811	0
15	53.971052	89.207350	1
16	69.070144	52.740470	1
17	67.946855	46.678574	0
18	70.661510	92.927138	1
19	76.978784	47.575964	1
20	67.372028	42.838438	0
21	89.676776	65.799366	1
22	50.534788	48.855812	0
23	34.212061	44.209529	0
24	77.924091	68.972360	1
25	62.271014	69.954458	1
26	80.190181	44.821629	1
27	93.114389	38.800670	0
28	61.830206	50.256108	0
29	38.785804	64.995681	0
...	...	...	...
70	32.722833	43.307173	0
71	64.039320	78.031688	1
72	72.346494	96.227593	1

	0	1	2
73	60.457886	73.094998	1
74	58.840956	75.858448	1
75	99.827858	72.369252	1
76	47.264269	88.475865	1
77	50.458160	75.809860	1
78	60.455556	42.508409	0
79	82.226662	42.719879	0
80	88.913896	69.803789	1
81	94.834507	45.694307	1
82	67.319257	66.589353	1
83	57.238706	59.514282	1
84	80.366756	90.960148	1
85	68.468522	85.594307	1
86	42.075455	78.844786	0
87	75.477702	90.424539	1
88	78.635424	96.647427	1
89	52.348004	60.769505	0
90	94.094331	77.159105	1
91	90.448551	87.508792	1
92	55.482161	35.570703	0
93	74.492692	84.845137	1
94	89.845807	45.358284	1
95	83.489163	48.380286	1
96	42.261701	87.103851	1
97	99.315009	68.775409	1
98	55.340018	64.931938	1
99	74.775893	89.529813	1

100 rows × 3 columns

In [4]:

```
#2.Check the data or preprocess the data
data1.isna().sum()
```

Out[4]:

```
0    0
1    0
2    0
dtype: int64
```

In [6]:

```
#3.Seperate input labels and output labels
x = data1[[0,1]]
# for displaying top 5
x.head()
```

Out[6]:

	0	1
0	34.623660	78.024693
1	30.286711	43.894998
2	35.847409	72.902198
3	60.182599	86.308552
4	79.032736	75.344376

In [7]:

```
# seperate target or output labels
y = data1[2]
y.head()
```

Out[7]:

```
0    0
1    0
2    0
3    1
4    1
Name: 2, dtype: int64
```

In [8]:

```
# find the number of rows and columns
data1.shape
```

Out[8]:

```
(100, 3)
```

In [9]:

```
# import the algorithm and train the model
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
```

In [10]:

```
# train the model using fit method
log.fit(x,y)
```

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

Out[10]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, 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 [11]:

```
# test the model using predict
y_pred = log.predict(x)
y_pred
```

Out[11]:

```
array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [12]:

```
# importing the accuracy and confussion matrix
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(y,y_pred)
```

Out[12]:

```
0.87
```

In [13]:

```
confusion_matrix(y,y_pred)
```

Out[13]:

```
array([[27, 13],
       [ 0, 60]], dtype=int64)
```

In [14]:

```
# predict the values of required values
log.predict([[12.245,14.325]])
```

Out[14]:

```
array([0], dtype=int64)
```

In [15]:

```
log.predict([[19.32,25.0]])
```

Out[15]:

```
array([0], dtype=int64)
```

In [16]:

```
log.predict([[48.23,68.63]])
```

Out[16]:

```
array([1], dtype=int64)
```

SVM(Support Vector Machine)

In [18]:

```
# read the data using read_csv
import pandas as pd
cancer = pd.read_csv("cancer.csv")
cancer.head()
```

Out[18]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	842302	17.99	10.38	122.80	1001.0	0.11840	
1	842517	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	11.42	20.38	77.58	386.1	0.14250	
4	84358402	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 32 columns

In [19]:

```
# for displaying the column or feature names
cancer.columns
```

Out[19]:

```
Index(['id', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_s
e',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'diagnosis'],
      dtype='object')
```

In [20]:

```
cancer.shape
```

Out[20]:

```
(569, 32)
```

In [21]:

```
cancer.isnull().sum()
```

Out[21]:

```
id                         0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
diagnosis                  0
dtype: int64
```

In [22]:

```
#seperate the input and output labels
x = cancer.iloc[:,1:31].values
x
```

Out[22]:

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

In [23]:

```
# seperate the y values from total columns
y = cancer.iloc[:,31].values
y
```

...

In [24]:

```
#target always must be in integer or float
# So that here we are converting our target values "M" and "B" to integers or binaries
# Labelencoder used for converting String to integers using fit_transform method
from sklearn.preprocessing import LabelEncoder
lab = LabelEncoder()
y_tran = lab.fit_transform(y)
y_tran
```

...

In [25]:

```
from sklearn.preprocessing import StandardScaler
k = StandardScaler()
x_tran = k.fit_transform(x)
x_tran
```

Out[25]:

```
array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])
```

In [32]:

```
# seperate train and test values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_tran,
                                               y_tran,
                                              random_state=1)
25% to test
75% to train
```

In [27]:

```
# import support vector machine algorithm
from sklearn.svm import SVC
svm = SVC(kernel='linear',random_state=0)
```

In [28]:

```
# train the model by using fit mrthod
svm.fit(x_train,y_train)
```

Out[28]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=0,
  shrinking=True, tol=0.001, verbose=False)
```

In [29]:

```
#test the model using predict method
pred = svm.predict(x_test)
```

In [30]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(y_test,pred)
```

Out[30]:

```
0.9532163742690059
```

In [31]:

```
confusion_matrix(y_test,pred)
```

Out[31]:

```
array([[103,   5],
       [  3,  60]], dtype=int64)
```

In [ ]:

```

```