Today Agenda
- Logistic Regression
- SVM

Logistic Regression(Classification Algorithm)

- It forms relation between features and targets
- It is having some similar properties related to Linear Regression

In [1]:

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

In [2]:

```
cancer = pd.read_csv("https://raw.githubusercontent.com/APSSDC-Data-Analysis/DA-TOT/ma
cancer.head()
```

Out[2]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
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 [3]:

```
cancer.shape
```

Out[3]:

```
(569, 32)
```

In [4]:

```
1 cancer.columns
```

Out[4]:

```
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 [5]:

```
1 # Preprocess the data
2 cancer.isnull().sum()
```

Out[5]:

```
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 [6]:

```
1 cancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
id                         569 non-null int64
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
diagnosis                  569 non-null object
dtypes: float64(30), int64(1), object(1)
memory usage: 142.3+ KB
```

In [7]:

```
# corelarion
cancer.corr()
```

Out[7]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.0129
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.1705
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	-0.0233
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.2072
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	0.1770
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	1.0000
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.6591
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.5219
concave points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.5536

In [8]:

```
cancer.columns
```

Out[8]:

```
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 [9]:

```
from sklearn.preprocessing import LabelEncoder
lab = LabelEncoder()
target = lab.fit_transform(cancer['diagnosis'])
target
```

Out[9]:

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

In [10]:

```
cancer.diagnosis.unique()
```

Out[10]:

```
array(['M', 'B'], dtype=object)
```

In [11]:

```
cancer['diagnosis'] = target
cancer.head()
```

Out[11]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
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 [12]:

```
p = cancer.corr()
p
```

Out[12]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	sm
id	1.000000	0.074626	0.099770	0.073159	0.096893	
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	
concave points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	
area_se	0.177742	0.735864	0.259845	0.744983	0.800086	
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	
compactness_se	0.033961	0.206000	0.191975	0.250744	0.212583	
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	
concave points_se	0.078768	0.376169	0.163851	0.407217	0.372320	
symmetry_se	-0.017306	-0.104321	0.009127	-0.081629	-0.072497	
fractal_dimension_se	0.025725	-0.042641	0.054458	-0.005523	-0.019887	
radius_worst	0.082405	0.969539	0.352573	0.969476	0.962746	
texture_worst	0.064720	0.297008	0.912045	0.303038	0.287489	
perimeter_worst	0.079986	0.965137	0.358040	0.970387	0.959120	
area_worst	0.107187	0.941082	0.343546	0.941550	0.959213	
smoothness_worst	0.010338	0.119616	0.077503	0.150549	0.123523	
compactness_worst	-0.002968	0.413463	0.277830	0.455774	0.390410	
concavity_worst	0.023203	0.526911	0.301025	0.563879	0.512606	
concave points_worst	0.035174	0.744214	0.295316	0.771241	0.722017	
symmetry_worst	-0.044224	0.163953	0.105008	0.189115	0.143570	
fractal_dimension_worst	-0.029866	0.007066	0.119205	0.051019	0.003738	
diagnosis	0.039769	0.730029	0.415185	0.742636	0.708984	

32 rows × 32 columns

In [13]:

```
1 cancer.columns
```

Out[13]:

```
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 [14]:

```
p1 = p['diagnosis']
p1
```

Out[14]:

```
id                         0.039769
radius_mean                0.730029
texture_mean               0.415185
perimeter_mean             0.742636
area_mean                  0.708984
smoothness_mean            0.358560
compactness_mean           0.596534
concavity_mean             0.696360
concave points_mean        0.776614
symmetry_mean              0.330499
fractal_dimension_mean    -0.012838
radius_se                  0.567134
texture_se                -0.008303
perimeter_se               0.556141
area_se                    0.548236
smoothness_se             -0.067016
compactness_se             0.292999
concavity_se               0.253730
concave points_se          0.408042
symmetry_se               -0.006522
fractal_dimension_se       0.077972
radius_worst               0.776454
texture_worst              0.456903
perimeter_worst            0.782914
area_worst                 0.733825
smoothness_worst           0.421465
compactness_worst          0.590998
concavity_worst            0.659610
concave points_worst       0.793566
symmetry_worst             0.416294
fractal_dimension_worst    0.323872
diagnosis                  1.000000
Name: diagnosis, dtype: float64
```

In [15]:

```
p2 = p1[p1>0.3]
p2
```

Out[15]:

```
radius_mean               0.730029
texture_mean              0.415185
perimeter_mean            0.742636
area_mean                 0.708984
smoothness_mean           0.358560
compactness_mean          0.596534
concavity_mean            0.696360
concave points_mean       0.776614
symmetry_mean             0.330499
radius_se                 0.567134
perimeter_se              0.556141
area_se                   0.548236
concave points_se         0.408042
radius_worst              0.776454
texture_worst             0.456903
perimeter_worst           0.782914
area_worst                0.733825
smoothness_worst          0.421465
compactness_worst         0.590998
concavity_worst           0.659610
concave points_worst      0.793566
symmetry_worst            0.416294
fractal_dimension_worst   0.323872
diagnosis                 1.000000
Name: diagnosis, dtype: float64
```

In [16]:

```
p3 = p2.index
p3
```

Out[16]:

```
Index(['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'radius_se', 'perimeter_se',
       'area_se', 'concave points_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 [17]:

```
input_featues = cancer[['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'radius_se', 'perimeter_se',
       'area_se', 'concave points_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst']]
```

In [18]:

```
output_features = cancer['diagnosis']
```

In [19]:

```
# Seperate the data into train and test data
from sklearn.model_selection import train_test_split
x_tr,x_te,y_tr,y_te=train_test_split(input_featues,output_features,
                                     test_size=0.3,
                                     random_state=2)
```

In [20]:

```
# importing required Algorithm
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
```

In [21]:

```
# train the model using fit method
log.fit(x_tr,y_tr)
```

```
C:\Users\RANGA\anakonda\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specif
y a solver to silence this warning.
  FutureWarning)
```

Out[21]:

```
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 [22]:

```
# predict the values for testing purpose
pred_values = log.predict(x_te)
```

In [23]:

```
# Evaluate the model
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(y_te,pred_values)
```

Out[23]:

```
0.935672514619883
```

Change the random_state and test_size paramer values check the accuracy

In [24]:

```
confusion_matrix(y_te,pred_values)
```

Out[24]:

```
array([[98,  6],
       [ 5, 62]], dtype=int64)
```

In [25]:

```
# sns.pairplot(cancer)
```

Support Vector Machine(SVM)

- It forms the margin between one group to another group
- By using kernal Technique it select the best margin and convert onto high dimen sionals
- We always better to take more margin and high dimension

There are two seperation techniques in svm
- Linear
- Circluer

In [26]:

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

In [27]:

```
tt = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Development-Corporat
tt.head()
```

Out[27]:

	survived	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarke
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [28]:

```
tt.columns
```

Out[28]:

```
Index(['survived', 'pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticke
t',
       'fare', 'cabin', 'embarked'],
      dtype='object')
```

In [29]:

```
tt.shape
```

Out[29]:

```
(891, 11)
```

In [30]:

```
1 tt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
survived    891 non-null int64
pclass      891 non-null int64
name        891 non-null object
sex         891 non-null object
age         714 non-null float64
sibsp       891 non-null int64
parch       891 non-null int64
ticket      891 non-null object
fare        891 non-null float64
cabin       204 non-null object
embarked    889 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 76.6+ KB
```

In [31]:

```
1 tt.isna().sum()
```

. . .

In [32]:

```
1 tt.drop('cabin',inplace=True,axis=1)
```

In [33]:

```
1 tt.columns
```

Out[33]:

```
Index(['survived', 'pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticke
t',
       'fare', 'embarked'],
      dtype='object')
```

In [34]:

```
1 tt.drop('name',axis=1,inplace=True)
```

In [35]:

```
1 tt.columns
```

Out[35]:

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'ticket', 'far
e',
       'embarked'],
      dtype='object')
```

In [36]:

```
from sklearn.preprocessing import LabelEncoder
lae = LabelEncoder()
```

In [37]:

```
tt['sex']=lae.fit_transform(tt['sex'])
tt['sex']
```

. . .

In [38]:

```
1 tt['age']
```

Out[38]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
5       NaN
6      54.0
7       2.0
8      27.0
9      14.0
10      4.0
11     58.0
12     20.0
13     39.0
14     14.0
15     55.0
16      2.0
17      NaN
18     31.0
19      NaN
20     35.0
21     34.0
22     15.0
23     28.0
24      8.0
25     38.0
26      NaN
27     19.0
28      NaN
29      NaN
       ...
861    21.0
862    48.0
863     NaN
864    24.0
865    42.0
866    27.0
867    31.0
868     NaN
869     4.0
870    26.0
871    47.0
872    33.0
873    47.0
874    28.0
875    15.0
876    20.0
877    19.0
878     NaN
879    56.0
880    25.0
881    33.0
882    22.0
883    28.0
884    25.0
```

```
885    39.0
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [39]:

```
tt['age']=tt['age'].fillna(round(tt['age'].mean()))
```

In [40]:

```
round(tt.age.mean())
```

Out[40]:

```
30
```

In [41]:

```
tt.age.isna().sum()
```

Out[41]:

```
0
```

In [42]:

```
tt['ticket'] = lae.fit_transform(tt['ticket'])
tt['ticket']
```

...

In [43]:

```
tt.embarked
```

...

In [44]:

```
tt.embarked.isna().sum()
```

Out[44]:

```
2
```

In [45]:

```
tt.embarked.unique()
```

Out[45]:

```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [46]:

```
tt.embarked.value_counts()
```

...

In [47]:

```
tt.embarked.unique()
```

Out[47]:

```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [48]:

```
tt.embarked.isna().sum()
```

Out[48]:

```
2
```

In [49]:

```
tt['embarked'] = tt.embarked.fillna('S')
tt['embarked'].isna().sum()
```

Out[49]:

```
0
```

In [50]:

```
tt['embarked'] = lae.fit_transform(tt['embarked'])
tt['embarked']
```

...

In [51]:

```
tt.info()
```

...

In [52]:

```
# seperate the data into input and output labels
input_labels = tt.drop('survived',axis=1)
input_labels.head()
```

...

In [53]:

```
output_labels = tt['survived']
```

In [60]:

```
# seperate data into training and testing data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(
input_labels,output_labels,random_state=10,test_size=0.3)
```

In [61]:

```
# import required Algorithm
from sklearn.svm import SVC
sv = SVC(kernel='linear',random_state=0)
```

In [62]:

```
# train the model
sv.fit(x_train,y_train)
```

...

In [63]:

```
# predict the values for testing
pred = sv.predict(x_test)
```

In [64]:

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

Out[64]:

```
0.8022388059701493
```

In [65]:

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

Out[65]:

```
array([[152,  22],
       [ 31,  63]], dtype=int64)
```

In [ ]:

```

```