Assignment No.4

Name: Vaishnav Kalidas Temgire   Roll No:23107127

Class : TY-B   Batch : B

In [1]:

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

In [13]:

```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

In [14]:

```
df
```

Out[14]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	c
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	
...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	

569 rows × 31 columns

In [15]:

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

Out[15]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

In [16]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #    Column                    Non-Null Count  Dtype
---   ------                    --------------  -----
 0    mean radius               569 non-null    float64
 1    mean texture              569 non-null    float64
 2    mean perimeter            569 non-null    float64
 3    mean area                 569 non-null    float64
 4    mean smoothness           569 non-null    float64
 5    mean compactness          569 non-null    float64
 6    mean concavity            569 non-null    float64
 7    mean concave points       569 non-null    float64
 8    mean symmetry             569 non-null    float64
 9    mean fractal dimension    569 non-null    float64
 10   radius error              569 non-null    float64
 11   texture error             569 non-null    float64
 12   perimeter error           569 non-null    float64
 13   area error                569 non-null    float64
 14   smoothness error          569 non-null    float64
 15   compactness error         569 non-null    float64
 16   concavity error           569 non-null    float64
 17   concave points error      569 non-null    float64
 18   symmetry error            569 non-null    float64
 19   fractal dimension error   569 non-null    float64
 20   worst radius              569 non-null    float64
 21   worst texture             569 non-null    float64
 22   worst perimeter           569 non-null    float64
 23   worst area                569 non-null    float64
 24   worst smoothness          569 non-null    float64
 25   worst compactness         569 non-null    float64
 26   worst concavity           569 non-null    float64
 27   worst concave points      569 non-null    float64
 28   worst symmetry            569 non-null    float64
 29   worst fractal dimension   569 non-null    float64
 30   target                    569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

In [17]:

```
df.describe()
```

Out[17]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	me compactne
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.0000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.1043
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.0528
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.0193
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.0649
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.0926
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.1304
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.3454

8 rows × 31 columns

In [18]:
```
df.ndim
```

Out[18]:
```
2
```

In [19]:
```
df.size
```

Out[19]:
```
17639
```

In [20]:
```
df.shape
```

Out[20]:
```
(569, 31)
```

In [21]:
```
df.columns
```

Out[21]:
```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension',
       'target'],
      dtype='object')
```

In [23]:
```
df['target'].value_counts()
```

Out[23]:
```
target
1    357
0    212
Name: count, dtype: int64
```

In [24]:

```
df['target'].mean()
```

Out[24]:

```
np.float64(0.6274165202108963)
```

In [25]:

```
df['target'].median()
```

Out[25]:

```
np.float64(1.0)
```

In [26]:

```
df['target'].mode()
```

Out[26]:

```
0    1
Name: target, dtype: int64
```

In [27]:

```
df['target'].std()
```

Out[27]:

```
np.float64(0.48391795640316865)
```

In [28]:

```
df['target'].var()
```

Out[28]:

```
np.float64(0.23417658852941906)
```

In [31]:

```
x = df.drop(['target'], axis=1)
x.head()
```

Out[31]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	cor P
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.

5 rows × 30 columns

In [33]:

```
y = df['target']
y
```

Out[33]:

```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: target, Length: 569, dtype: int64
```

In [34]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size= 0.3, ran
```

In [35]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [57]:

```
x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.transform(x_test)
```

In [58]:

```
from sklearn.svm import SVC
svm_model = SVC(kernel='rbf', C=1, gamma= 'scale', random_state = 30 )
```

In [59]:

```
svm_model.fit(x_train_scaled, y_train)
```

Out[59]:

▾ SVC ⓘ ?

SVC(C=1, random_state=30)

In [60]:

```
y_pred = svm_model.predict(x_test_scaled)
```

In [61]:

```
y_pred
```

```
Out[61]:  array([0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
                 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
                 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
                 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0,
                 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
                 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
                 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
                 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1,
                 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0,
                 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
                 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0,
                 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1,
                 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1,
                 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
                 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                 0, 0, 1])
```

```
In [68]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_r
         cm = confusion_matrix(y_test, y_pred)
         accuracy = accuracy_score(y_test, y_pred)
         cr  = classification_report(y_test,  y_pred)

         print("Accuracy : ",accuracy)
         print("Classification Report : \n",cr)
         print("Confusion Matrix : ",cm)
```

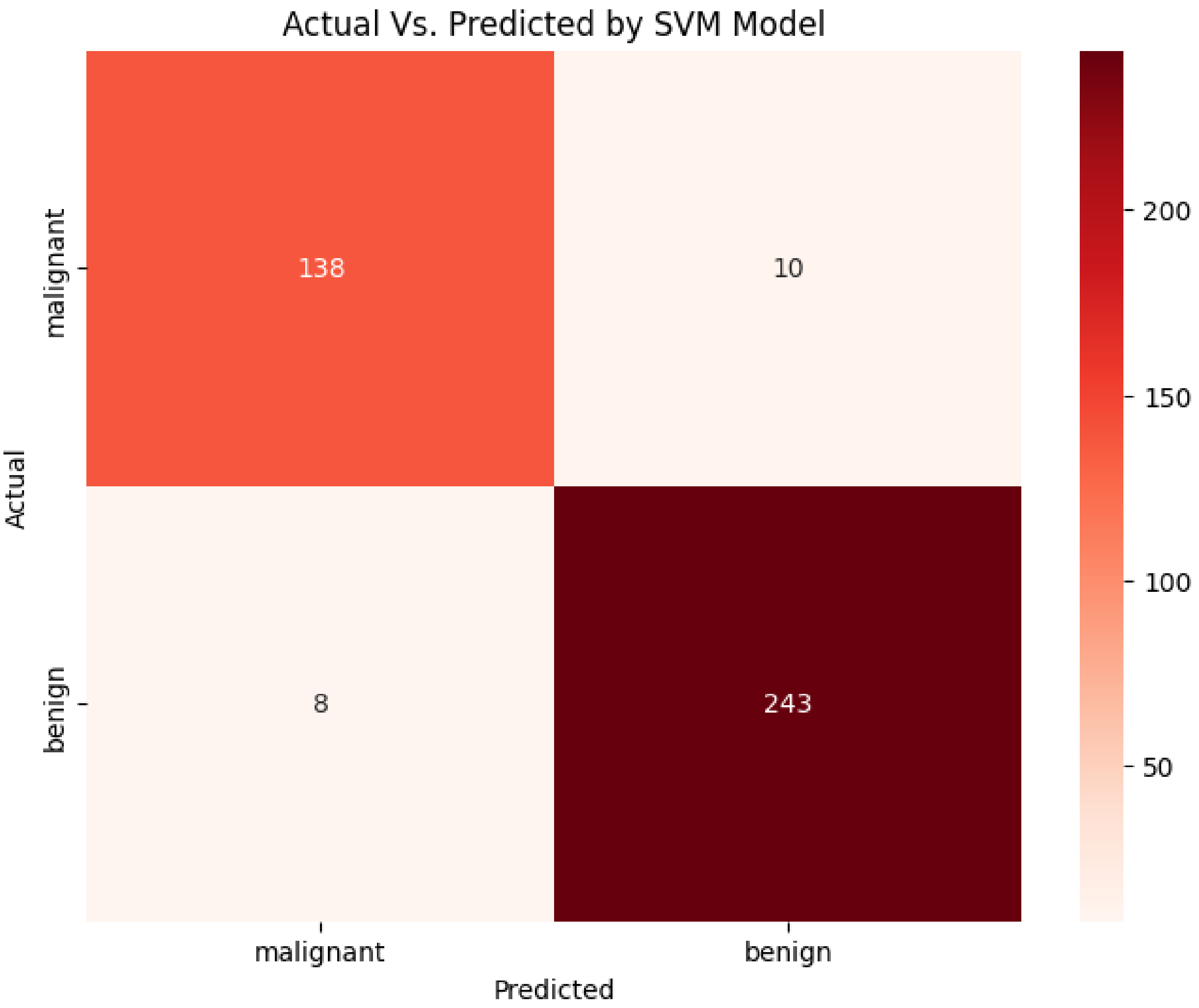
```
Accuracy  :   0.9548872180451128
Classification Report :
               precision    recall  f1-score   support

           0       0.95      0.93      0.94       148
           1       0.96      0.97      0.96       251

    accuracy                           0.95       399
   macro avg       0.95      0.95      0.95       399
weighted avg       0.95      0.95      0.95       399

Confusion Matrix :   [[138   10]
 [   8 243]]
```

```
In [72]: plt.figure(figsize=(8,6))
         sns.heatmap(cm, annot= True, fmt='d', cmap='Reds',xticklabels=data.target_name
         plt.xlabel("Predicted")
         plt.ylabel("Actual")
         plt.title("Actual Vs. Predicted by SVM Model")
         plt.show()
```


Actual Vs. Predicted by SVM Model
Actual
malignant
benign
138
10
8
243
malignant
benign
Predicted
200
150
100
50

In [ ]: