```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Attribute Information:

1-29: Ten Real-Valued Features are Computed for Each Cell Nucleus:
a. Radius (Mean of Distances from Center to Points on the Perimeter)
b. Texture (Standard Deviation of Gray-Scale Values)
c. Perimeter
d. Area
e. Smoothness (Local Variation in Radius Lengths)
f. Compactness (Perimeter^2 / Area - 1.0)
g. Concavity (Severity of Concave Portions of the Contour)
h. Concave Points (Number of Concave Portions of the Contour)
i. Symmetry
j. Fractal Dimension ("Coastline Approximation" - 1)

30: Diagnosis (M = Malignant, B = Benign)

The Mean, Standard Error, and "Worst" or Largest (Mean of the Three Largest Values) of These Features Were Computed for Each Image, Resulting in 30 Features. For Instance, Field 3 is Mean Radius, Field 13 is Radius SE, Field 23 is Worst Radius.

All Feature Values are Recoded with Four Significant Digits.

Missing Attribute Values: None

Class Distribution: 357 Benign, 212 Malignant

```
#loading the data
dataset = pd.read_csv('/content/breastcancer.csv')
```

```
dataset.head()
```

	rm	tm	pm	am	sm	cm	cnm	cpm	sym	fdm	...	tw	pw	aw	sw	cw	cnw	cpw	
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0

5 rows × 31 columns

```
# no. of rows and column in dataset
dataset.shape
```

```
(569, 31)
```

```
dataset['Outcome'] = dataset['Outcome'].map({'M': 1, 'B': 0})
```

```
dataset.head()
```

	rm	tm	pm	am	sm	cm	cnm	cpm	sym	fdm	...	tw	pw	aw	sw	cw	cnw	cpw	
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0

```
dataset['Outcome'].value_counts()
# 0 --> malignant
# 1 --> benign
```

```
0    357
1    212
Name: Outcome, dtype: int64
```

```
dataset.groupby('Outcome').mean()
# mean value of the inputs for a particular coutcomes
```

Outcome	rm	tm	pm	am	sm	cm	cnm	cpm	sym	fdm	...	rw	tw
0	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058	0.025717	0.174186	0.062867	...	13.379801	23.515070
1	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775	0.087990	0.192909	0.062680	...	21.134811	29.318208

2 rows × 30 columns

```
# Checking Co-relation
correlation_matrix = dataset.corr()
label_correlations = correlation_matrix['Outcome']
print(label_correlations)
```

```
rm         0.730029
tm         0.415185
pm         0.742636
am         0.708984
sm         0.358560
cm         0.596534
cnm        0.696360
cpm        0.776614
sym        0.330499
fdm       -0.012838
rse        0.567134
tse       -0.008303
pse        0.556141
ase        0.548236
sse       -0.067016
cse        0.292999
cnse       0.253730
cpse       0.408042
symse     -0.006522
fdse       0.077972
rw         0.776454
tw         0.456903
pw         0.782914
aw         0.733825
sw         0.421465
cw         0.590998
cnw        0.659610
cpw        0.793566
symw       0.416294
fdw        0.323872
Outcome    1.000000
Name: Outcome, dtype: float64
```

```
dropping_features = ['tm', 'sm', 'sym', 'fdm', 'rse', 'tse', 'pse', 'ase', 'sse', 'cse', 'cnse', 'cpse', 'symse', 'fdse', 'rw', 'tw', 'pw', '
X = dataset.drop(columns = 'Outcome', axis = 1)
Y = dataset['Outcome']
X = X.drop(columns=dropping_features)
```

```
print(X)
print(Y)
```

```
        rm      pm      am       cm      cnm      cpm
0    17.99  122.80  1001.0  0.27760  0.30010  0.14710
1    20.57  132.90  1326.0  0.07864  0.08690  0.07017
2    19.69  130.00  1203.0  0.15990  0.19740  0.12790
3    11.42   77.58   386.1  0.28390  0.24140  0.10520
4    20.29  135.10  1297.0  0.13280  0.19800  0.10430
..     ...     ...     ...      ...      ...      ...
564  21.56  142.00  1479.0  0.11590  0.24390  0.13890
565  20.13  131.20  1261.0  0.10340  0.14400  0.09791
566  16.60  108.30   858.1  0.10230  0.09251  0.05302
567  20.60  140.10  1265.0  0.27700  0.35140  0.15200
568   7.76   47.92   181.0  0.04362  0.00000  0.00000

[569 rows x 6 columns]
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: Outcome, Length: 569, dtype: int64
```

Data Standardization :

```
scaler = StandardScaler()
```

```
scaler.fit(X)
X=scaler.transform(X)

print(X)
```

```
[[ 1.09706398  1.26993369  0.9843749   3.28351467  2.65287398  2.53247522]
 [ 1.82982061  1.68595471  1.90870825 -0.48707167 -0.02384586  0.54814416]
 [ 1.57988811  1.56650313  1.55888363  1.05292554  1.36347845  2.03723076]
 ...
 [ 0.70228425  0.67267578  0.57795264 -0.03867967  0.04658753  0.10577736]
 [ 1.83834103  1.98252415  1.73521799  3.27214378  3.296944    2.65886573]
 [-1.80840125 -1.81438851 -1.34778924 -1.15075248 -1.11487284 -1.26181958]]
```

Splitting Data :

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, stratify = Y, random_state = 42)
```

```
print(X.shape, x_test.shape, x_train.shape)
```

```
(569, 6) (114, 6) (455, 6)
```

Training Model :

```
classifier = svm.SVC(kernel='linear')
```

```
classifier.fit(x_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

Checking Accuracy :

```
x_train_prediction = classifier.predict(x_train)
train_accuracy = accuracy_score(x_train_prediction, y_train)
print('Accuracy Score for SVC on training data is : ', train_accuracy)
```

```
Accuracy Score for SVC on training data is :  0.9164835164835164
```

```
x_test_prediction = classifier.predict(x_test)
```

```
test_accuracy = accuracy_score(x_test_prediction, y_test)
print('Accuracy Score for SVC on testing data is : ', test_accuracy)
```

```
Accuracy Score for SVC on testing data is :  0.9035087719298246
```

Creating Model file :

```
from joblib import dump

filename = 'breastcancer_model.joblib'
dump(classifier, filename)
```

```
['breastcancer_model.joblib']
```

Loading Model file:

```
from joblib import load
model = load('/content/breastcancer_model.joblib')
```

Checking the execution of model :

```
input_data = (17.99,  122.80,  1001.0,  0.27760,  0.30010,  0.14710)

# Changing to a numpy array
input_numpyarray = np.asarray(input_data)

# Reshaping
input_reshaped = input_numpyarray.reshape(1, -1)

# Fit the scaler on your training data (if available)
# You should replace X_train with your actual training data
# scaler.fit(X_train)

# Standardize the input data
inp = scaler.transform(input_reshaped)
y_pred = model.predict(inp)

if int(y_pred[0]) == 1:
    print('Melignant')
else:
    print('Benign')
```

```
Melignant
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was f
  warnings.warn(
```

```
input_data2 = (7.76,   47.92,   181.0,  0.04362,  0.00000,  0.00000)

# Changing to a numpy array
input_numpyarray2 = np.asarray(input_data2)

# Reshaping
input_reshaped2 = input_numpyarray2.reshape(1, -1)

# Fit the scaler on your training data (if available)
# You should replace X_train with your actual training data
# scaler.fit(X_train)

# Standardize the input data
inp2 = scaler.transform(input_reshaped2)
y_pred = model.predict(inp2)

if int(y_pred[0]) == 1:
    print('Melignant')
else:
    print('Benign')
```

```
Benign
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was f
  warnings.warn(
```