```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data Collection and Analysis, Loading Breast Cancer Wisconsin (Diagnostic) dataset.

```
cancer_dataset = pd.read_csv("/content/data.csv")
```

```
# printing the first 5 rows of the dataset
cancer_dataset.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 32 columns

```
print(cancer_dataset)
```

```
565    926682         M        20.13         28.25          131.20     1261.0
566    926954         M        16.60         28.08          108.30      858.1
567    927241         M        20.60         29.33          140.10     1265.0
568     92751         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave_points_mean  \
0            0.11840           0.27760         0.30010              0.14710   
1            0.08474           0.07864         0.08690              0.07017   
2            0.10960           0.15990         0.19740              0.12790   
3            0.14250           0.28390         0.24140              0.10520   
4            0.10030           0.13280         0.19800              0.10430   
..               ...               ...             ...                  ...   
564          0.11100           0.11590         0.24390              0.13890   
565          0.09780           0.10340         0.14400              0.09791   
566          0.08455           0.10230         0.09251              0.05302   
567          0.11780           0.27700         0.35140              0.15200   
568          0.05263           0.04362         0.00000              0.00000   

     ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0    ...        25.380          17.33           184.60      2019.0   
1    ...        24.990          23.41           158.80      1956.0   
2    ...        23.570          25.53           152.50      1709.0   
3    ...        14.910          26.50            98.87       567.7   
4    ...        22.540          16.67           152.20      1575.0   
..   ...           ...            ...              ...         ...   
564  ...        25.450          26.40           166.10      2027.0   
565  ...        23.690          38.25           155.00      1731.0   
566  ...        18.980          34.12           126.70      1124.0   
567  ...        25.740          39.42           184.60      1821.0   
568  ...         9.456          30.37            59.16       268.6   

     smoothness_worst  compactness_worst  concavity_worst  \
0             0.16220            0.66560           0.7119   
1             0.12380            0.18660           0.2416   
2             0.14440            0.42450           0.4504   
3             0.20980            0.86630           0.6869   
4             0.13740            0.20500           0.4000   
..                ...                ...              ...   
564           0.14100            0.21130           0.4107   
565           0.11660            0.19220           0.3215   
566           0.11390            0.30940           0.3403   
567           0.16500            0.86810           0.9387   
568           0.08996            0.06444           0.0000   
```

```
566                0.1418          0.2218                  0.07820
567                0.2650          0.4087                  0.12400
568                0.0000          0.2871                  0.07039

[569 rows x 32 columns]
```

```
# number of rows and Columns in this dataset
cancer_dataset.shape
```

```
(569, 32)
```

```
# getting the statistical measures of the data
cancer_dataset.describe()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.

8 rows × 31 columns

```
# counting the number of missing values in the dataset
cancer_dataset.isnull().sum()
```

```
id                         0
diagnosis                  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
dtype: int64
```

```
cancer_dataset["diagnosis"].value_counts()
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
# load the Label Encoder Function
label_encode = LabelEncoder()
```

```
labels = label_encode.fit_transform(cancer_dataset.diagnosis)
```

```
# appending the labels to the dataframe
cancer_dataset["diagnosis"]=labels
```

```
cancer_dataset.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothn
0	842302	1	17.99	10.38	122.80	1001.0	
1	842517	1	20.57	17.77	132.90	1326.0	
2	84300903	1	19.69	21.25	130.00	1203.0	
3	84348301	1	11.42	20.38	77.58	386.1	
4	84358402	1	20.29	14.34	135.10	1297.0	

5 rows × 32 columns

0 ----Benign 1-----Malignant

```
cancer_dataset["diagnosis"].value_counts()
```

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

Data Standardization

The process of standardizing the data to a common format and common range

```
X = cancer_dataset.drop(columns = 'diagnosis', axis=1)
Y = cancer_dataset["diagnosis"]
```

```
print(X)
```

```
565    926682        20.13         28.25          131.20     1261.0
566    926954        16.60         28.08          108.30      858.1
567    927241        20.60         29.33          140.10     1265.0
568     92751         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave_points_mean  \
0            0.11840           0.27760         0.30010              0.14710
1            0.08474           0.07864         0.08690              0.07017
2            0.10960           0.15990         0.19740              0.12790
3            0.14250           0.28390         0.24140              0.10520
4            0.10030           0.13280         0.19800              0.10430
..               ...               ...             ...                  ...
564          0.11100           0.11590         0.24390              0.13890
565          0.09780           0.10340         0.14400              0.09791
566          0.08455           0.10230         0.09251              0.05302
567          0.11780           0.27700         0.35140              0.15200
568          0.05263           0.04362         0.00000              0.00000

     symmetry_mean  ...  radius_worst  texture_worst  perimeter_worst  \
0           0.2419  ...        25.380          17.33           184.60
1           0.1812  ...        24.990          23.41           158.80
2           0.2069  ...        23.570          25.53           152.50
3           0.2597  ...        14.910          26.50            98.87
4           0.1809  ...        22.540          16.67           152.20
..             ...  ...           ...            ...              ...
564         0.1726  ...        25.450          26.40           166.10
565         0.1752  ...        23.690          38.25           155.00
566         0.1590  ...        18.980          34.12           126.70
567         0.2397  ...        25.740          39.42           184.60
568         0.1587  ...         9.456          30.37            59.16

     area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0        2019.0           0.16220            0.66560           0.7119
1        1956.0           0.12380            0.18660           0.2416
2        1709.0           0.14440            0.42450           0.4504
3         567.7           0.20980            0.86630           0.6869
```

```
     concave_points_worst  symmetry_worst  fractal_dimension_worst
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
2                  0.2430          0.3613                  0.08758
3                  0.2575          0.6638                  0.17300
4                  0.1625          0.2364                  0.07678
..                    ...             ...                      ...
564                0.2216          0.2060                  0.07115
565                0.1628          0.2572                  0.06637
566                0.1418          0.2218                  0.07820
567                0.2650          0.4087                  0.12400
568                0.0000          0.2871                  0.07039

[569 rows x 31 columns]
```

```
print(Y)
```

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

Splitting the data into training data and test data

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =0.2,random_state = 3)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(569, 31) (455, 31) (114, 31)
```

Standardize the Data

```
scaler = StandardScaler()
```

```
scaler.fit(X_train)
```

```
▾ StandardScaler
StandardScaler()
```

```
X_train_standardized = scaler.transform(X_train)
```

```
print(X_train_standardized)
```

```
[[-0.17447005  1.40381088  1.79283426 ...  1.044121    0.52295995
   0.64990763]
 [-0.24176487  1.16565505 -0.14461158 ...  0.5940779   0.44153782
  -0.85281516]
 [-0.24147181 -0.0307278  -0.77271123 ... -0.64047556 -0.31161687
  -0.69292805]
 ...
 [-0.24184096  1.06478904  0.20084323 ...  0.01694621  3.06583565
  -1.29952679]
 [ 0.49310842  1.51308238  2.3170559  ...  1.14728703 -0.16599653
   0.82816016]
 [-0.24167719 -0.73678981 -1.02636686 ... -0.31826862 -0.40713129
  -0.38233653]]
```

```
X_test_standardized = scaler.transform(X_test)
```

```
print(X_test_standardized)
```

```
[[-0.1744851  -0.99455847 -0.05522817 ... -0.5697545   0.02503231
  -0.50225186]
 [-0.24145565  0.10656204  0.03898678 ... -0.32036185 -0.70933265
  -0.5692316 ]
 [-0.24149426 -0.72278064 -0.03348626 ... -0.75844367  0.17378428
  -0.12576093]
 ...
 [-0.24132334  0.68654154  2.15036791 ...  0.36531844 -1.10548262
  -0.37639478]
 [-0.24900942 -0.35293864 -1.46362085 ... -0.95251542 -0.9629939
```

```
   -0.86523882]
  [-0.2415074  -0.50703947 -1.02153533 ... -1.03818808 -0.98021781
   -1.37352859]]
```

```
print(X_train_standardized.std())
print(X_test_standardized.std())
```

```
     1.0
     0.8792879780993126
```

```
X_train = X_train_standardized
X_test = X_test_standardized
```

Training the Model

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

```
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

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

Model Evaluation

Accuracy Score

```
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
     Accuracy score of the training data :  0.9868131868131869
```

Generate Using ... a slider using jupyter widgets Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro ×

```
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
     Accuracy score of the test data :  0.9649122807017544
```

Making a Predictive System

```
input_data = (5,166,72,19,175,25.8,0.587,51,5,166,72,19,175,25.8,0.587,51,5,166,72,19,175,25.8,0.587,51,5,166,72,19,175,25.8,0.587)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
     [[-2.49083997e-01  4.25459319e+01  1.27604205e+01 -2.97027778e+00
       -1.34977587e+00  1.83546997e+03  8.85416480e+00  6.21848777e+02
        1.25170926e+02  5.82096758e+03  9.83207351e+03  6.41278010e+01
        3.19264924e+02  1.07744947e+01 -8.42511354e-01  1.62347294e+04
```

```
   2.65423937e+02  5.18243679e+03  1.13296981e+04  2.17446131e+03
   6.22557510e+04  1.91844145e+00 -4.16189885e+00 -1.65406544e+00
  -1.51059807e+00  7.37061887e+03  4.47498457e+02  8.86875232e+01
   2.61477486e+03  3.99400572e+02  2.71069035e+01]]
[0]
The person is not diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler wa
  warnings.warn(
```

Start coding or generate with AI.