<u>U18ISI6204 – Machine Learning Techniques</u>

<u>Lab Experiment – 5</u>

<u>NAME:</u> Aaron Mathew

<u>ROLL_NO:</u> 20BIS001

Use a sample dataset and with help of Support Vector Machine, classify the subject whether it has cancer or not

INTRODUCTION

In this experiment, we have to perform Support vector machine on the cancer dataset.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non- linear data and classifier used is called as Non-linear SVM classifier.

Support Vectors:

- The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

OBJECTIVE OF THE EXERCISE/EXPERIMENT

To perform Support vector Machine on the given dataset, using scikit library

STEP 2:

ACQUISITION

PROCEDURE:

STEP-1: Start the program.

STEP-2: import all the necessary libraries

iv) Numpy – array manipulation
v) Pandas – dataframe manipulation
vi) Matplotlib and seaborn – for data visualization

vii) Sklearn.model_selection – train test data split
viii) Sklearn.metrics –confusion matrix and classification report.
ix) Sklearn,svm– for support vector regression
x) Sklearn.decomposition – for PCA
xi) Sklearn.preprocessing – for Normalisation

STEP-3: Loading the dataset using read_csv method in pandas module.

STEP-4: Analyze the dataset using info method, which gives its data types and number of non-null values in each columns.

STEP-5: Perform basic statistic operation using describe() method.

STEP-6: Use heatmaps, correlation matrix, regression plots and pairplots in seaborn to find the relationship between features.

STEP-7: Normalize the data points

STEP-8: Using selective feature, perform PCA in order to reduce number of feature from 30 to 11.

STEP-9: Implement SVM with 11 PCA variable and calculate classification report and confusion matrix.

STEP-10: Stop the program.

PROGRAM:

Importing libraries

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
```

from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

In [1]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Loading dataset

df = pd.read_csv('C:\\Users\\spd85\\Downloads\\cancer\\data.csv')
df.head()

In [4]:
```
df = pd.read_csv('C:\\Users\\spd85\\Downloads\\cancer\\data.csv')
df.head()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	tex
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 33 columns

Basic statistics operations

df.dtypes

df[df.duplicated()].shape

df.describe()

df.columns

In [15]:
```
df.dtypes
```

Out[15]:
```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
```

```
In [5]: df[df.duplicated()].shape
Out[5]: (0, 33)
```

```
In [6]: df.describe()
Out[6]:
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetr
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0

8 rows × 32 columns

```
In [7]: df.columns
Out[7]: Index(['id', 'diagnosis', '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_se',
       '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', 'Unnamed: 32'],
      dtype='object')
```

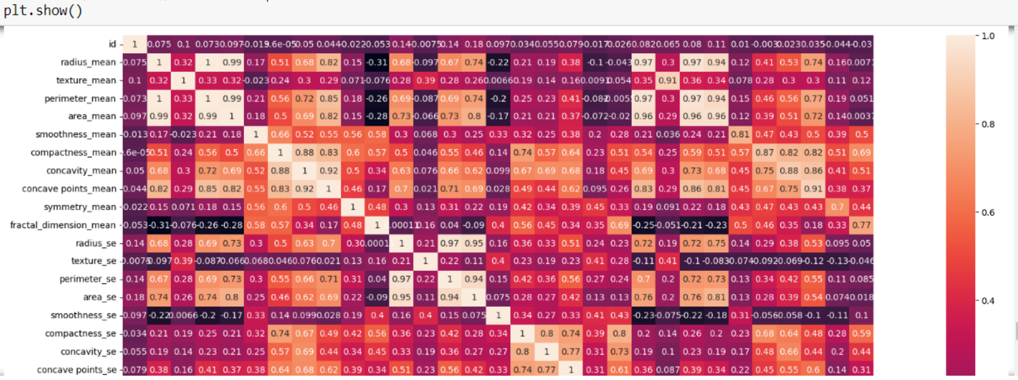
Correlation between columns

plt.figure(figsize=(20,12))

sns.heatmap(df.corr(),annot=True)

plt.show()

```
In [39]: plt.figure(figsize=(20,12))
         sns.heatmap(df.corr(),annot=True)
         plt.show()
```



Normalization and PCA.

scaler = StandardScaler( )

X_scaled = pd.DataFrame(scaler.fit_transform(X))

X_scaled_drop = X_scaled.drop(X_scaled.columns[[2,3,12,13,22,23]],axis=1)

pca = PCA(n_components=0.95)

x_pca = pca.fit_transform(X_scaled_drop)

x_pca = pd.DataFrame(x_pca)

print("Before PCA, X dataframe shape = ",X.shape,"\nAfter PCA, x_pca dataframe shape = ",x_pca.shape)

print(pca.explained_variance_ratio_)

print(pca.explained_variance_ratio_.sum())

y = df.diagnosis

print(y.shape)

y.head()

print(x_pca.shape)

print(y.shape)

```
In [16]: X = df.iloc[:,2:32]
         print(X.shape)
         X.head()

(569, 30)

Out[16]:
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dime
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	

5 rows × 30 columns

```
In [18]: scaler = StandardScaler()
         X_scaled = pd.DataFrame(scaler.fit_transform(X))
         X_scaled_drop = X_scaled.drop(X_scaled.columns[[2,3,12,13,22,23]],axis=1)

In [19]: pca = PCA(n_components=0.95)
         x_pca = pca.fit_transform(X_scaled_drop)
         x_pca = pd.DataFrame(x_pca)

         print("Before PCA, X dataframe shape = ",X.shape,"\nAfter PCA, x_pca dataframe shape = ",x_pca.shape)

         Before PCA, X dataframe shape =  (569, 30)
         After PCA, x_pca dataframe shape =  (569, 11)

In [21]: print(pca.explained_variance_ratio_)
         print(pca.explained_variance_ratio_.sum())

         [0.42661046 0.15932139 0.10294428 0.07788731 0.06489774 0.05015242
          0.02145044 0.0187846  0.01505759 0.01197751 0.01117206]
         0.960255820189289
```

In [25]:
```
y = df.diagnosis
print(y.shape)
y.head()
```

```
(569,)
```

Out[25]:
```
0    M
1    M
2    M
3    M
4    M
Name: diagnosis, dtype: object
```

In [26]:
```
print(x_pca.shape)
print(y.shape)
```

```
(569, 11)
(569,)
```

Train test split, SVM model and model evaluation:

```
X_train, X_test, y_train, y_test = train_test_split(x_pca, y, test_size=0.25, random_state=0)
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("Confusion matrix:\n",cm)
report = classification_report(y_test, y_pred)
print("Classification report:\n",report)
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(x_pca, y, test_size=0.25, random_state=0)
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
```

In [28]:
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion matrix:\n",cm)
report = classification_report(y_test, y_pred)
print("Classification report:\n",report)
```

```
Confusion matrix:
 [[89  1]
 [ 4 49]]
Classification report:
               precision    recall  f1-score   support

           B       0.96      0.99      0.97        90
           M       0.98      0.92      0.95        53

    accuracy                           0.97       143
   macro avg       0.97      0.96      0.96       143
weighted avg       0.97      0.97      0.96       143
```