**U18ISI6204 – Machine Learning Techniques**

**Lab Experiment – 5**

|  |  |
| --- | --- |
| **NAME** | **J.SANDHIYA** |
| **ROLL NO** | **20BIS036** |

**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

1. Numpy – array manipulation
2. Pandas – dataframe manipulation
3. Matplotlib and seaborn – for data visualization
4. Sklearn.model\_selection – train test data split
5. Sklearn.metrics –confusion matrix and classification report.
6. Sklearn,svm– for support vector regression
7. Sklearn.decomposition – for PCA
8. 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

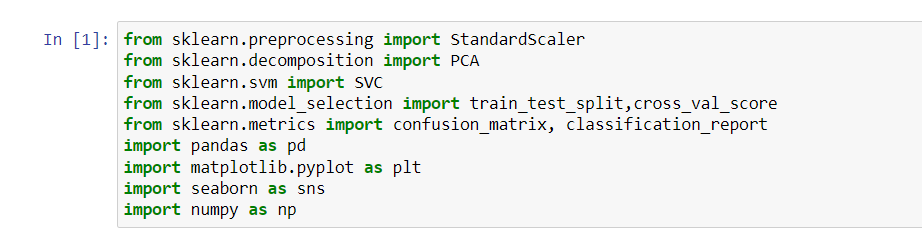
from sklearn.metrics import confusion\_matrix, classification\_report

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

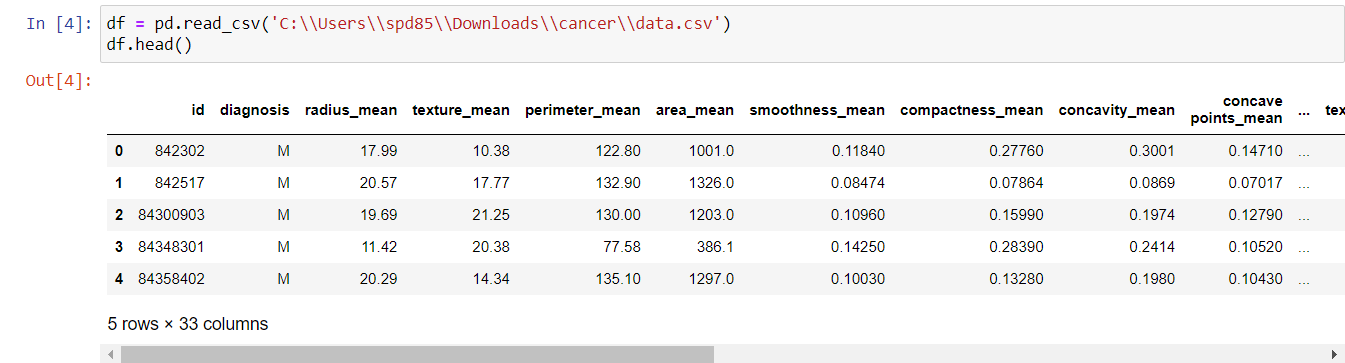
import numpy as np

****

# Loading dataset

# df = pd.read\_csv('C:\\Users\\spd85\\Downloads\\cancer\\data.csv')

# df.head()

****

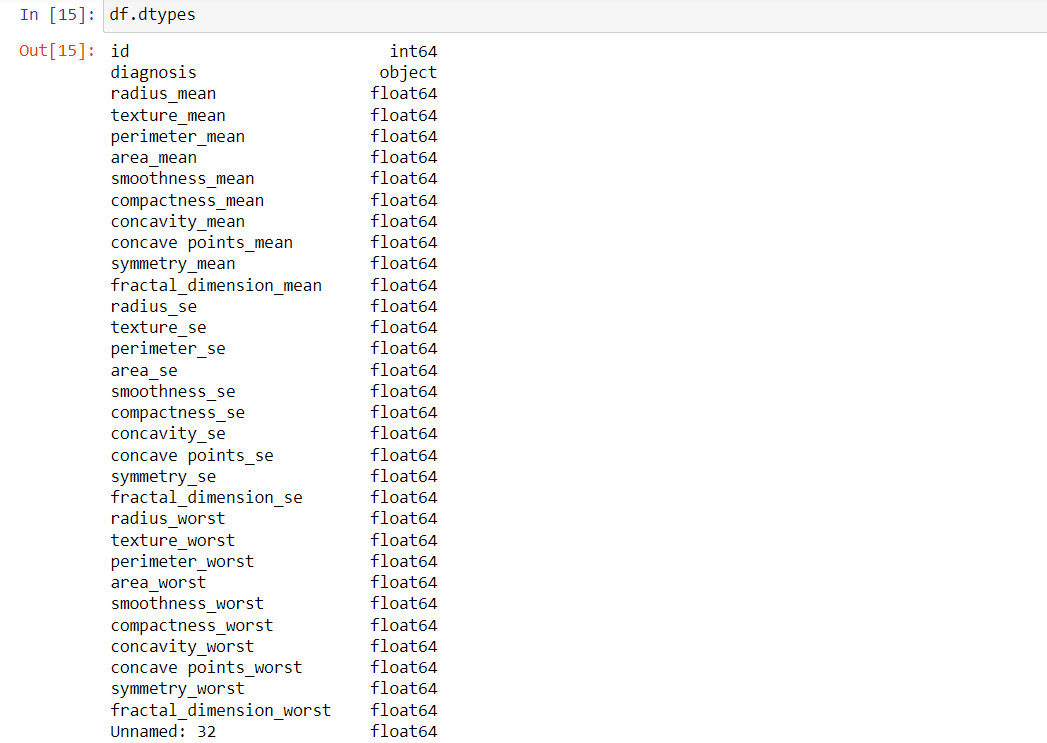
**Basic statistics operations**

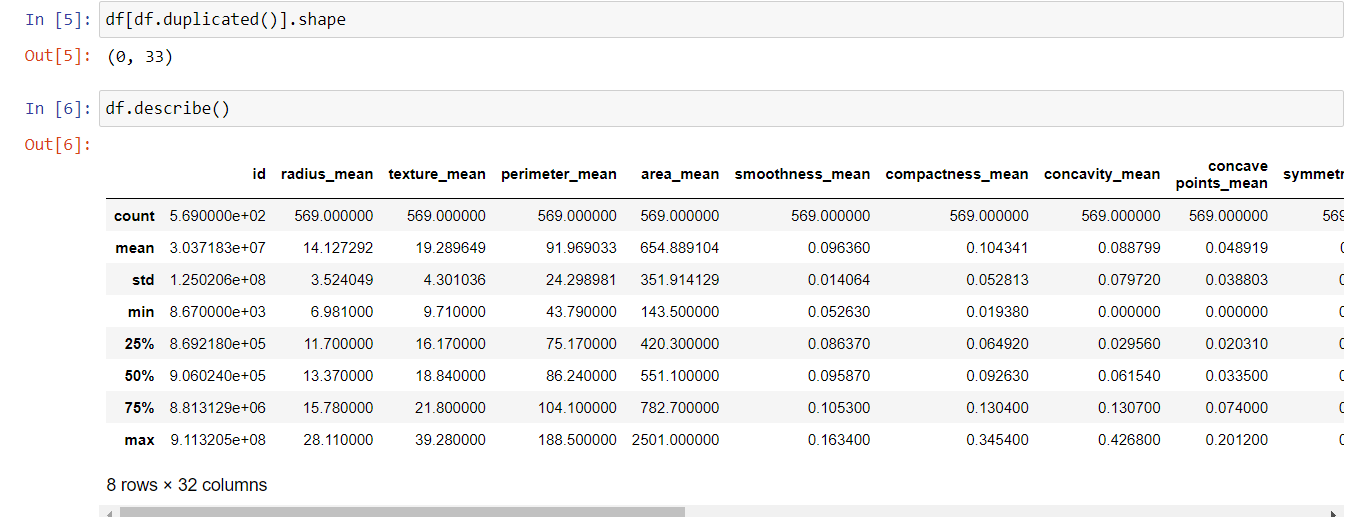
df.dtypes

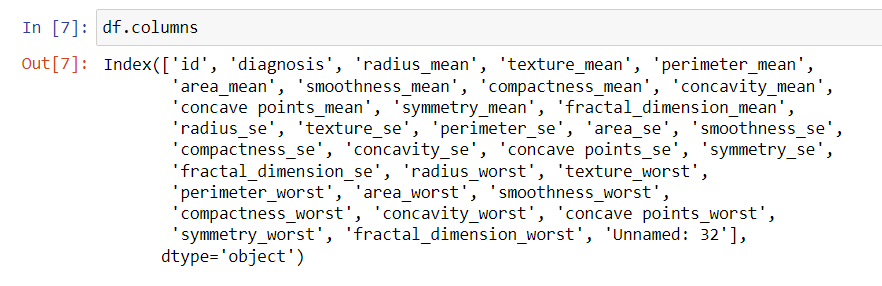
df[df.duplicated()].shape

df.describe()

df.columns

****

****

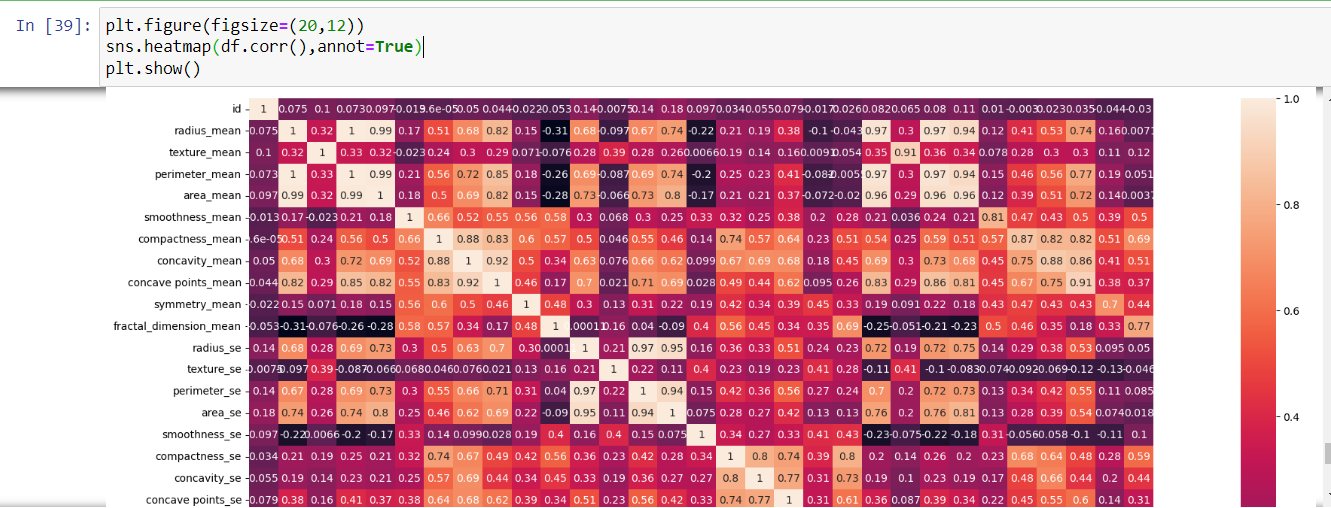
****

# Correlation between columns

# 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())

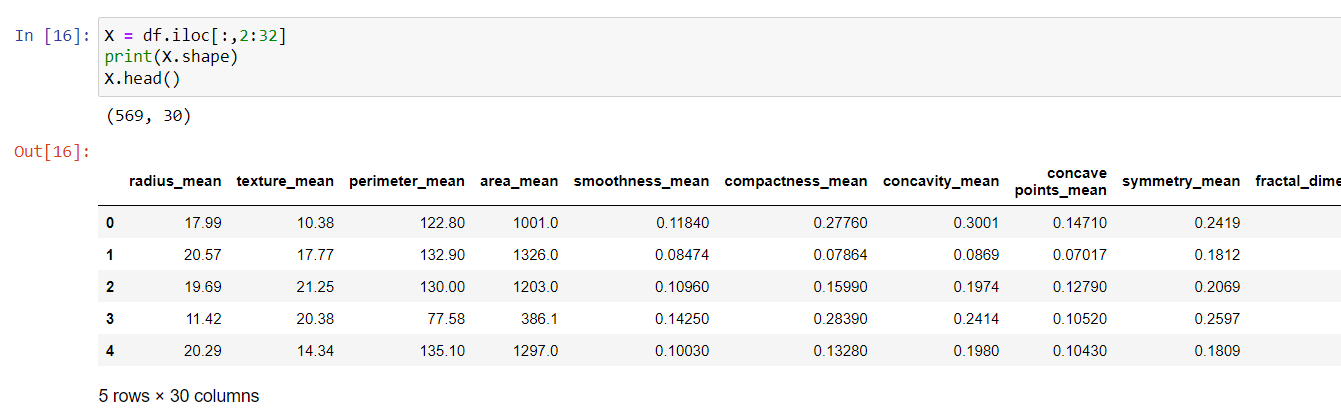
y = df.diagnosis

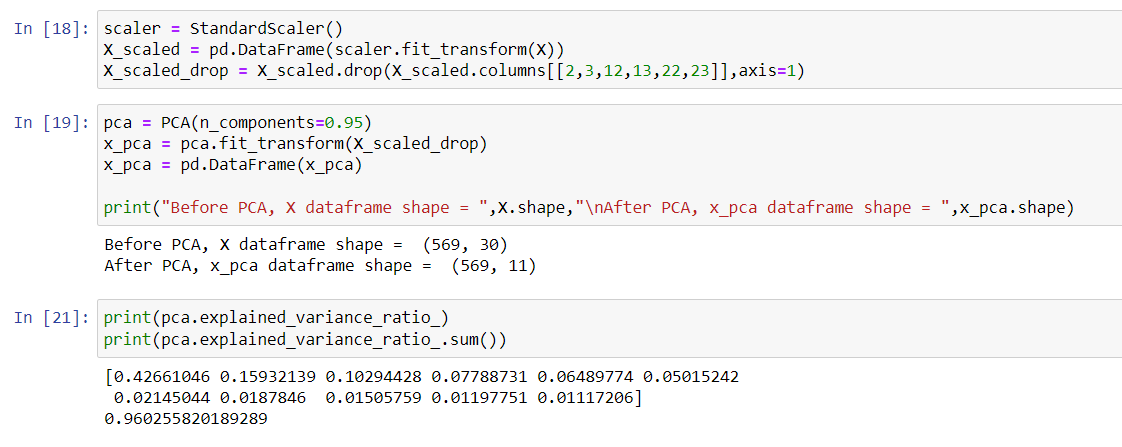
print(y.shape)

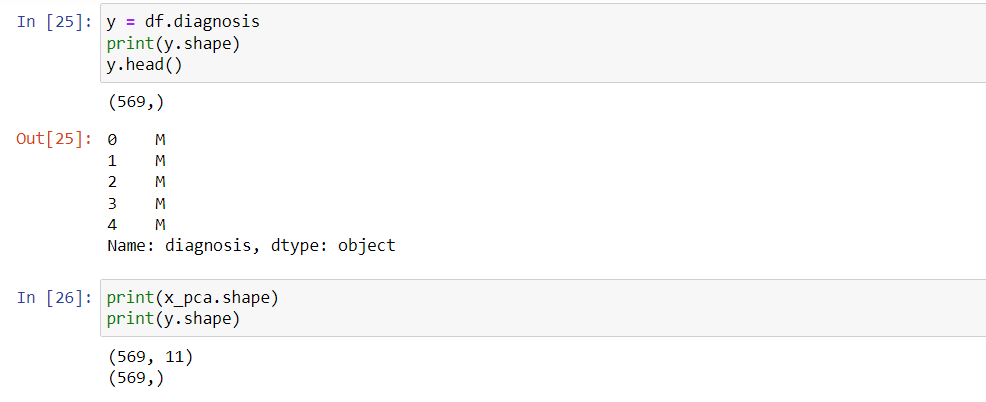
y.head()

print(x\_pca.shape)

print(y.shape)

****

****

****

**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)

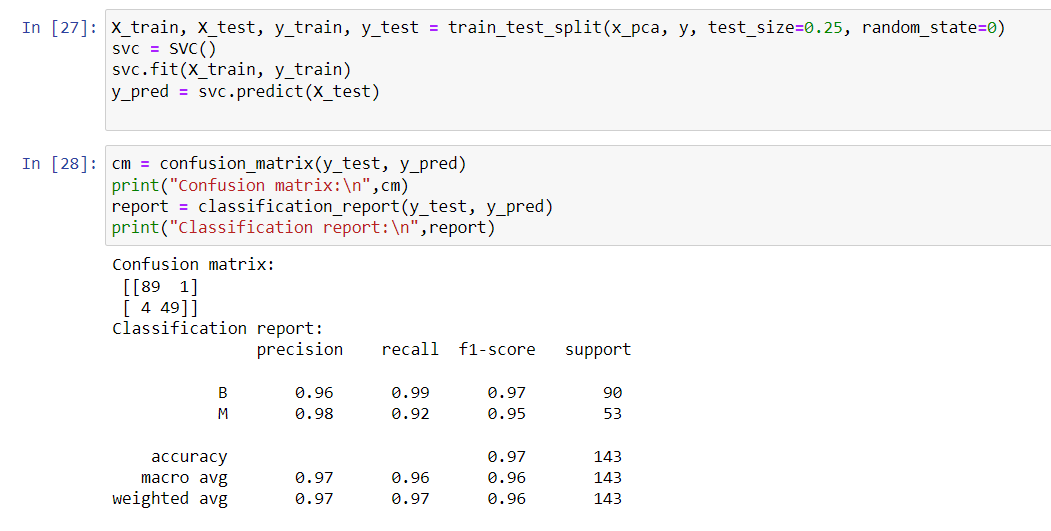
# y\_pred = svc.predict(X\_test)

# cm = confusion\_matrix(y\_test, y\_pred)

# print("Confusion matrix:\n",cm)

# report = classification\_report(y\_test, y\_pred)

# print("Classification report:\n",report)

****