Roll No: RBT21CB013

SVM implementation on diabetes dataset

SVM or Support Vector Machines are used in machine learning and pattern recognition for classification and regression problems, especially when dealing with large datasets. They are relatively simple to understand and use, but also very powerful and effective. In this article, we are going to classify the Iris dataset using different SVM kernels using Python's Scikit-Learn package. To keep it simple and understandable we will only use features from the dataset — glucose,blood pressure and skinthickness.

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

# Importing the datasets

df = pd.read_csv('/content/diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.shape
```

```
(768, 9)
```

```
X = df.iloc[:, [1,2,3]]
Y = df.iloc[:, 8]
```

```
X.head()
```

	Glucose	BloodPressure	SkinThickness
0	148	72	35
1	85	66	29
2	183	64	0
3	89	66	23
4	137	40	35

```
Y.head()
```

```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

Feature Engineering

Although there were no missing values found in the dataset, there still needs to be some feature engineering done before implementing the SVM model. The three features Glucose, Blood pressure, and Skin thickness have minimum values of 0. However, these features cannot be equal to zero because humans can't survive with zero glucose, blood pressure, or skin thickness. So to solve this issue all values equal to zero in each of those three features were turned into null values and they were just ignored for simplicity

Double-click (or enter) to edit

```
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size = 0.25, random_state = 0)


print("Training data : ",X_Train.shape)
print("Training data : ",X_Test.shape)
```

```
Training data :  (576, 3)
Training data :  (192, 3)
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_Train = sc_X.fit_transform(X_Train)
X_Test = sc_X.transform(X_Test)


from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_Train, Y_Train)

# Predicting the test set results
Y_Pred = classifier.predict(X_Test)


Y_Pred
```

```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0])
```

```
from sklearn import metrics
print('Accuracy Score: with linear kernel')

print(metrics.accuracy_score(Y_Test,Y_Pred))
```

```
Accuracy Score: with linear kernel
0.7552083333333334
```

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf')
classifier.fit(X_Train, Y_Train)

# Predicting the test set results
Y_Pred = classifier.predict(X_Test)

print('Accuracy Score: with default rbf kernel')
print(metrics.accuracy_score(Y_Test,Y_Pred))
```

```
Accuracy Score: with default rbf kernel
0.765625
```

```
from sklearn.svm import SVC
#classifier = SVC(kernel = 'rbf', gamma = 10,  random_state=0)  # 93% accuracy
classifier = SVC(kernel = 'rbf', gamma = 15, C=7,  random_state=0)
classifier.fit(X_Train, Y_Train)

# Predicting the test set results

Y_Pred = classifier.predict(X_Test)

print('Accuracy Score On Test Data: with default rbf kernel')
print(metrics.accuracy_score(Y_Test,Y_Pred))
```

```
Accuracy Score On Test Data: with default rbf kernel
0.6614583333333334
```

```
svc=SVC(kernel='poly', degree = 4)
svc.fit(X_Train,Y_Train)

y_pred=svc.predict(X_Test)
print('Accuracy Score:with poly kernel and degree ')
print(metrics.accuracy_score(Y_Test,Y_Pred))
```

```
Accuracy Score:with poly kernel and degree
0.6614583333333334
```

```
import matplotlib.pyplot as plt

plt.scatter(X_Train[:, 0], X_Train[:, 1],c=Y_Train)
plt.xlabel('Glucose')
plt.ylabel('blood pressure')
plt.title('Diabetes detection')
plt.show()
```