Diabetese prediction using support vector machine

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

Importing Data and analysing

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

```
diabetes_dataset.head()
diabetes_dataset.shape
```

```
(768, 9)
```

```
diabetes_dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000

```
diabetes_dataset['Outcome'].value_counts()

# data is not properly distributed hence we have to use standart scaler to make all featues of almost same range an
# test and train dataset get same length of both categories
# 0    500  (non-diabatic)
# 1    268  (diabatic)
# Name: Outcome, dtype: int64
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

```
diabetes_dataset.groupby('Outcome').mean()
```

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537

Separating features and label

```
x = diabetes_dataset.drop('Outcome', axis=1)
x.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

```
y = diabetes_dataset.Outcome
y.head()
```

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

Data standardisation

(as data is highly deviated and it would cause lowering in the accuracy of of our model prediction)

```
scaler = StandardScaler()
```

```
scaler.fit(x)

# The .fit() method then takes the feature matrix X_train and target variable y_train as arguments and fits the
# model to the training data. During this process, the model learns the coefficients and intercept that define the de
```

```
#  The transform method, is used to apply the learned transformation to new data.
std_x = scaler.transform(x)
```

```
std_x[2]  #numpy_array
```

```
array([ 1.23388019,  1.94372388, -0.26394125, -1.28821221, -0.69289057,
       -1.10325546,  0.60439732, -0.10558415])
```

```
x = std_x
```

Splitting data into training and testing part

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)
```

```
len(x_train)
len(x_test)
print(len(x_train)/(len(x_test)+len(x_train)))
```

```
0.7994791666666666
```

```
print(y_train.value_counts())
print(y_test.value_counts())

# stratified on y
```

```
0    400
1    214
Name: Outcome, dtype: int64
0    100
1     54
Name: Outcome, dtype: int64
```

Training Model

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

```
# training our svc
classifier.fit(x_train, y_train)
```

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

Model Evaluation

```
# accuracy score on my training data
x_train_prediction = classifier.predict(x_train)
x_train_acc  = accuracy_score(x_train_prediction, y_train)
```

```
print('Accuracy score of our training data : ', x_train_acc)
# Accuracy score of our training data :  0.7915309446254072
```

```
Accuracy score of our training data :  0.7915309446254072
```

```
# accuracy score of testing data
x_test_prediction = classifier.predict(x_test)
x_test_acc  = accuracy_score(x_test_prediction, y_test)
print('Accuracy score of testing data : ', x_test_acc)
# Accuracy score of testing data :  0.7207792207792207
```

```
Accuracy score of testing data :  0.7207792207792207
```