Importing the Dependencies

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

Data Collection and Analysis

PIMA Diabetes Dataset

```
# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
```

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

```
   Pregnancies  Glucose  BloodPressure  ...  DiabetesPedigreeFunction
Age  Outcome
0            6      148             72  ...                     0.627
50        1
1            1       85             66  ...                     0.351
31        0
2            8      183             64  ...                     0.672
32        1
3            1       89             66  ...                     0.167
21        0
4            0      137             40  ...                     2.288
33        1

[5 rows x 9 columns]
```

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

```
(768, 9)
```

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

```
       Pregnancies     Glucose  ...         Age     Outcome
count   768.000000  768.000000  ...  768.000000  768.000000
mean      3.845052  120.894531  ...   33.240885    0.348958
std       3.369578   31.972618  ...   11.760232    0.476951
min       0.000000    0.000000  ...   21.000000    0.000000
25%       1.000000   99.000000  ...   24.000000    0.000000
50%       3.000000  117.000000  ...   29.000000    0.000000
75%       6.000000  140.250000  ...   41.000000    1.000000
max      17.000000  199.000000  ...   81.000000    1.000000
```

```
[8 rows x 9 columns]
```

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

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

0 --> Non-Diabetic

1 --> Diabetic

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

```
         Pregnancies     Glucose  ...  DiabetesPedigreeFunction
Age
Outcome                           ...

0           3.298000  109.980000  ...                  0.429734
31.190000
1           4.865672  141.257463  ...                  0.550500
37.067164

[2 rows x 8 columns]
```

```
# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

```
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  ...   BMI
DiabetesPedigreeFunction  Age
0              6      148             72  ...  33.6
0.627   50
1              1       85             66  ...  26.6
0.351   31
2              8      183             64  ...  23.3
0.672   32
3              1       89             66  ...  28.1
0.167   21
4              0      137             40  ...  43.1
2.288   33
..           ...      ...            ...  ...   ...
...  ...
763           10      101             76  ...  32.9
0.171   63
764            2      122             70  ...  36.8
0.340   27
765            5      121             72  ...  26.2
0.245   30
```

```
766            1      126             60  ...  30.1
0.349   47
767            1       93             70  ...  30.4
0.315   23

[768 rows x 8 columns]
```

```
print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =
0.2, stratify=Y, random_state=2)
```

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

```
(768, 8) (614, 8) (154, 8)
```

Training the Model

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

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

```
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 :  0.7833876221498371
```

```
# 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 :  0.7727272727272727
```

Making a Predictive System

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

# 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)

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

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

```
[1]
The person is diabetic

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446:
UserWarning: X does not have valid feature names, but SVC was fitted
with feature names
  "X does not have valid feature names, but"
```

Saving the trained model

```
import pickle

filename = 'trained_model.sav'
pickle.dump(classifier, open(filename, 'wb'))

# loading the saved model
loaded_model = pickle.load(open('trained_model.sav', 'rb'))

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

# 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)
```

```
prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

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

```
[1]
The person is diabetic
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446:
UserWarning: X does not have valid feature names, but SVC was fitted
with feature names
  "X does not have valid feature names, but"
```