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  SkinThickness  Insulin
BMI  \
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


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
# 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  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
```

```
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

```
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  BloodPressure  SkinThickness
Insulin  \
Outcome

0           3.298000  109.980000      68.184000      19.664000
68.792000
1           4.865672  141.257463      70.824627      22.164179
100.335821

              BMI  DiabetesPedigreeFunction        Age
Outcome
```

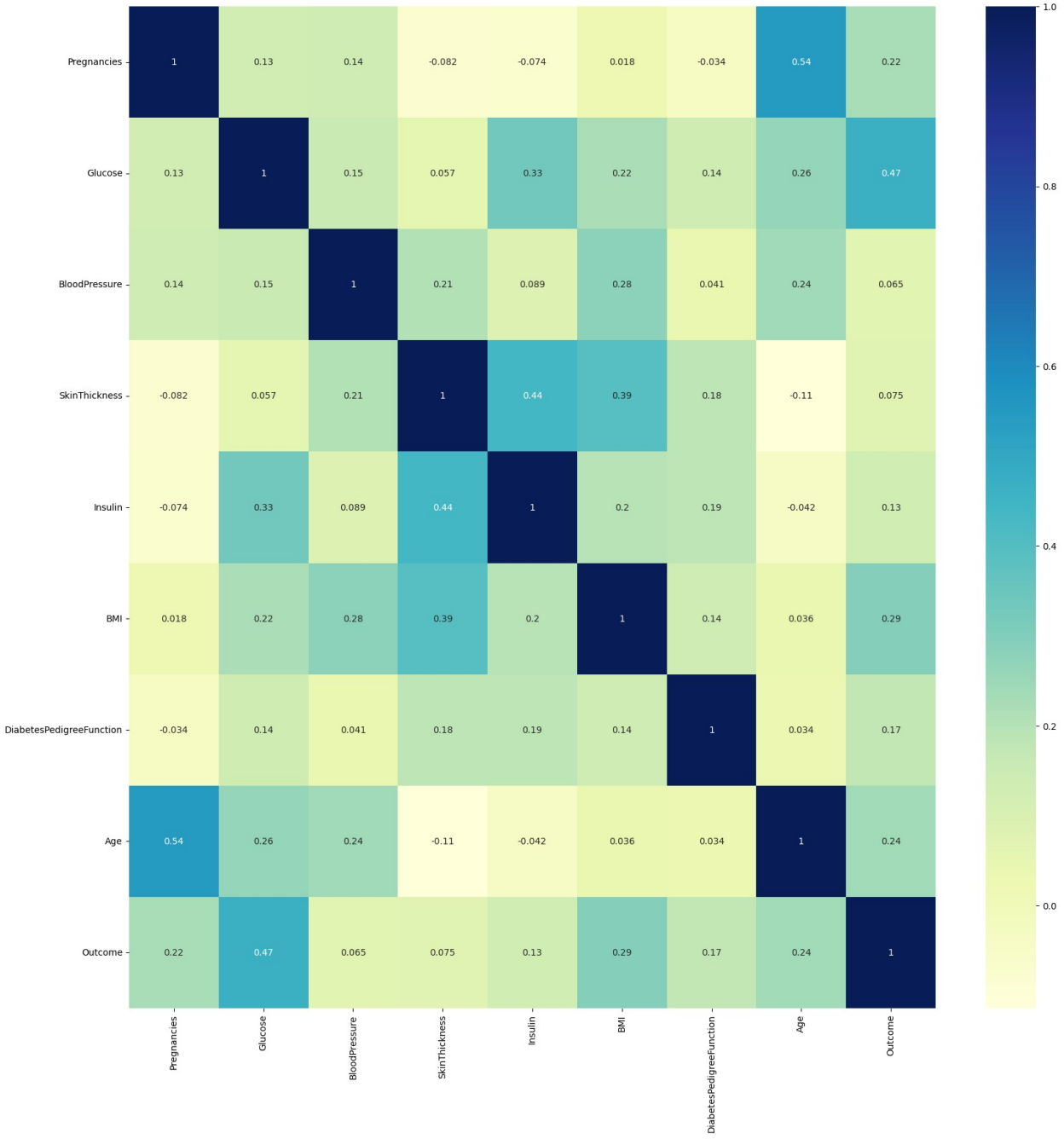
```
0        30.304200                  0.429734  31.190000
1        35.142537                  0.550500  37.067164
```

```
# To find the corelation in data set we use corr()
import matplotlib.pyplot as plt
import seaborn as sns
corrMatrix=diabetes_dataset.corr()
corrMatrix
```

```
                          Pregnancies   Glucose  BloodPressure
SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -
0.081672
Glucose                      0.129459  1.000000       0.152590
0.057328
BloodPressure                0.141282  0.152590       1.000000
0.207371
SkinThickness               -0.081672  0.057328       0.207371
1.000000
Insulin                     -0.073535  0.331357       0.088933
0.436783
BMI                          0.017683  0.221071       0.281805
0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265
0.183928
Age                          0.544341  0.263514       0.239528      -
0.113970
Outcome                      0.221898  0.466581       0.065068
0.074752

                            Insulin       BMI  DiabetesPedigreeFunction
\
Pregnancies               -0.073535  0.017683                 -0.033523

Glucose                    0.331357  0.221071                  0.137337

BloodPressure              0.088933  0.281805                  0.041265

SkinThickness              0.436783  0.392573                  0.183928

Insulin                    1.000000  0.197859                  0.185071

BMI                        0.197859  1.000000                  0.140647

DiabetesPedigreeFunction   0.185071  0.140647                  1.000000

Age                       -0.042163  0.036242                  0.033561

Outcome                    0.130548  0.292695                  0.173844
```

```
                                Age   Outcome
Pregnancies                0.544341  0.221898
Glucose                    0.263514  0.466581
BloodPressure              0.239528  0.065068
SkinThickness             -0.113970  0.074752
Insulin                   -0.042163  0.130548
BMI                        0.036242  0.292695
DiabetesPedigreeFunction   0.033561  0.173844
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

```
import seaborn as sns
sns.heatmap(corrMatrix,cmap="YlGnBu",annot=True)
plt.gcf().set_size_inches(20, 20)
```




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

print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
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

..             ...      ...             ...             ...          ...   ...

763             10      101              76              48          180  32.9

764              2      122              70              27            0  36.8

765              5      121              72              23          112  26.2

766              1      126              60               0            0  30.1

767              1       93              70              31            0  30.4


     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     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/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but SVC was fitted
with feature names
  warnings.warn(
```

Saving the trained model

```
import pickle

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

# loading the saved model
loaded_model = pickle.load(open('diabetes_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/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but SVC was fitted
with feature names
  warnings.warn(
```

```
for column in X.columns:
  print(column)
```

```
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
```