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

```
df = pd.read_csv("/content/diabetes_prediction_dataset.csv")
```

```
df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

```
df.shape
```

```
(100000, 9)
```

```
df.describe()
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.885856	0.07485	0.039420	27.320767	5.527507	138.058060	0.085000
std	22.516840	0.26315	0.194593	6.636783	1.070672	40.708136	0.278883
min	0.080000	0.00000	0.000000	10.010000	3.500000	80.000000	0.000000
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.000000	0.000000
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.000000	0.000000
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.000000	0.000000
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.000000	1.000000

```
df['diabetes'].value_counts()
```

```
0    91500
1     8500
Name: diabetes, dtype: int64
```

```
df['gender'].value_counts()
```

```
Female    58552
Male      41430
Other        18
Name: gender, dtype: int64
```

```
#Converting categorical variable to numeric

import pandas as pd
gender_mapping = {'Male': 1, 'Female': 0, 'Other': 2}

df['numeric_gender'] = df['gender'].map(gender_mapping)
print(df)
```

```
       gender   age  hypertension  heart_disease smoking_history    bmi  \
0      Female  80.0             0              1           never  25.19
1      Female  54.0             0              0         No Info  27.32
2        Male  28.0             0              0           never  27.32
3      Female  36.0             0              0         current  23.45
4        Male  76.0             1              1         current  20.14
...       ...   ...           ...            ...             ...    ...
99995  Female  80.0             0              0         No Info  27.32
99996  Female   2.0             0              0         No Info  17.37
99997    Male  66.0             0              0          former  27.83
```

```
99998  Female  24.0             0              0          never  35.42
99999  Female  57.0             0              0        current  22.43

       HbA1c_level  blood_glucose_level  diabetes  numeric_gender
0              6.6                  140         0               0
1              6.6                   80         0               0
2              5.7                  158         0               1
3              5.0                  155         0               0
4              4.8                  155         0               1
...            ...                  ...       ...             ...
99995          6.2                   90         0               0
99996          6.5                  100         0               0
99997          5.7                  155         0               1
99998          4.0                  100         0               0
99999          6.6                   90         0               0

[100000 rows x 10 columns]
```

```
df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	

```
df['numeric_gender'].value_counts()
```

```
0    58552
1    41430
2       18
Name: numeric_gender, dtype: int64
```

```
df['smoking_history'].value_counts()
```

```
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: smoking_history, dtype: int64
```

```
#Converting categorical variable to numeric

import pandas as pd
smoking_history_numeric = {'No Info': 0, 'never': 1, 'former': 2, 'current': 3 ,'not current':4, 'ever':5}

df['smoking_history_numeric'] = df['smoking_history'].map(smoking_history_numeric)
print(df)
```

```
       gender   age  hypertension  heart_disease smoking_history    bmi  \
0      Female  80.0             0              1           never  25.19
1      Female  54.0             0              0         No Info  27.32
2        Male  28.0             0              0           never  27.32
3      Female  36.0             0              0         current  23.45
4        Male  76.0             1              1         current  20.14
...       ...   ...           ...            ...             ...    ...
99995  Female  80.0             0              0         No Info  27.32
99996  Female   2.0             0              0         No Info  17.37
99997    Male  66.0             0              0          former  27.83
99998  Female  24.0             0              0           never  35.42
99999  Female  57.0             0              0         current  22.43

       HbA1c_level  blood_glucose_level  diabetes  numeric_gender  \
0              6.6                  140         0               0
1              6.6                   80         0               0
2              5.7                  158         0               1
3              5.0                  155         0               0
4              4.8                  155         0               1
...            ...                  ...       ...             ...
99995          6.2                   90         0               0
99996          6.5                  100         0               0
99997          5.7                  155         0               1
```

```
99998         4.0                 100        0               0
99999         6.6                  90        0               0

       smoking_history_numeric
0                            1
1                            0
2                            1
3                            3
4                            3
...                        ...
99995                        0
99996                        0
99997                        2
99998                        1
99999                        3

[100000 rows x 11 columns]
```

```
df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	

```
# Drop the specified columns
columns_to_remove = ['gender', 'smoking_history']
df = df.drop(columns=columns_to_remove)
```

```
df.head()
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
0	80.0	0	1	25.19	6.6	140	0
1	54.0	0	0	27.32	6.6	80	0
2	28.0	0	0	27.32	5.7	158	0
3	36.0	0	0	23.45	5.0	155	0
4	76.0	1	1	20.14	4.8	155	0

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

Data Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
standardized_data = scaler.fit_transform(X)
```

```
print(standardized_data)
```

```
[[ 1.69270354 -0.28443945  4.93637859 ...  0.04770422 -0.84104674
  -0.19575171]
 [ 0.53800643 -0.28443945 -0.20257766 ... -1.42620999 -0.84104674
  -0.90848322]
 [-0.61669069 -0.28443945 -0.20257766 ...  0.48987848  1.18723364
  -0.19575171]
 ...
 [ 1.07094356 -0.28443945 -0.20257766 ...  0.41618277  1.18723364
   0.5169798 ]
 [-0.7943364  -0.28443945 -0.20257766 ... -0.93490525 -0.84104674
  -0.19575171]
 [ 0.67124071 -0.28443945 -0.20257766 ... -1.18055762 -0.84104674
   1.22971132]]
```

```
X = standardized_data
Y= df['diabetes']
```

Train Test Split

```
from sklearn.model_selection import 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)
```

```
(100000, 8) (80000, 8) (20000, 8)
```

Training the Model

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

```
# Initialize SVM classifier
svm_classifier = SVC(kernel='linear', C=1.0, random_state=42)
```

```
svm_classifier.fit(X_train, Y_train)
```

```
▾                 SVC
SVC(kernel='linear', random_state=42)
```

```
# Make predictions on the testing set
y_pred = svm_classifier.predict(X_test)
```

```
# accuracy score on the training data
X_train_prediction = svm_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 :  0.96045
```

```
# accuracy score on the testing data
X_test_prediction = svm_classifier.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the testing data : ', testing_data_accuracy)
```

```
Accuracy score of the testing data :  0.96165
```

```
# Evaluate the performance of the classifier
accuracy = accuracy_score(Y_test, y_pred)
classification_report_output = classification_report(Y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report_output)
```

```
Accuracy: 0.96
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     18300
           1       0.92      0.60      0.73      1700

    accuracy                           0.96     20000
   macro avg       0.94      0.80      0.85     20000
weighted avg       0.96      0.96      0.96     20000
```

Making a predictive System

```
df.head(10)
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
0	80.0	0	1	25.19	6.6	140	0
1	54.0	0	0	27.32	6.6	80	0
2	28.0	0	0	27.32	5.7	158	0
3	36.0	0	0	23.45	5.0	155	0
4	76.0	1	1	20.14	4.8	155	0
5	20.0	0	0	27.32	6.6	85	0
6	44.0	0	0	19.31	6.5	200	1
7	79.0	0	0	23.86	5.7	85	0
8	42.0	0	0	33.64	4.8	145	0
9	32.0	0	0	27.32	5.0	100	0

```
input_data = (44.0,1,1,19.31,6.5,200,1,1)


# 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)


# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)
```

```
[[ 0.09389215  3.51568677  4.93637859 -1.20703158  0.90830598  1.52161842
   1.18723364 -0.19575171]]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was f
  warnings.warn(
```

```
prediction = svm_classifier.predict(std_data)
print(prediction)
```

```
[1]
```

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

```
    The person is diabetic
```

Saving the trained model

```
import pickle
```

```
filename ='trained_model.sav'
```

```
pickle.dump(svm_classifier, open(filename,'wb'))
```

Loading the saved model

```
loaded_model =  pickle.load(open('trained_model.sav','rb'))
```

```
input_data = (44.0,1,1,19.31,6.5,200,1,1)

# 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 = svm_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
```