I. Importing essential libraries

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

II. Importing and understanding our dataset

```
heart_data = pd.read_csv('/content/heart_disease_data.csv')
```

Verifying it as a 'dataframe' object in pandas

```
heart_data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Shape of dataset

```
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Printing out a few columns

```
dataset.head(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
heart_data.shape
```

```
(303, 14)
```

```
heart_data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
trestbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
thalach	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exang	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slope	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
ca	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thal	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
target	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

```
heart_data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
heart_data['target'].value_counts()
```

```
target
1    165
0    138
Name: count, dtype: int64
```

1 ---> Heart Disease 0 ---> Healthy Heart

```
heart_data.groupby('target').mean()
```

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212

```
X = heart_data.drop(columns='target', axis=1)
y = heart_data['target']
```

```
print(X)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

```
print(y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, 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)
```

```
(303, 13) (242, 13) (61, 13)
```

Model Training

```
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, y_train)
```

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

Model Prediction and Evaluation

```
# Accuracy on train data
train_pred = classifier.predict(X_train)
acc = accuracy_score(y_train, train_pred)
print("Training accuracy of SVM Model is {}".format(acc))
```

```
Training accuracy of SVM Model is 0.8553719008264463
```

```
# Accuracy on test data
prediction = classifier.predict(X_test)
acc = accuracy_score(y_test, prediction)
print("Accuracy of SVM Model is {}".format(acc))

prec = precision_score(y_test, prediction)
print("Precision of Model is {}".format(prec))

rec = recall_score(y_test, prediction)
print("Recall of Model is {}".format(rec))

f1 = f1_score(y_test, prediction)
print("F1-Score of Model is {}".format(f1))
```

```
Accuracy of SVM Model is 0.819672131147541
Precision of Model is 0.8055555555555556
Recall of Model is 0.8787878787878788
F1-Score of Model is 0.8405797101449276
```

Heart Disease Prediction System

```
# use any data instance from heart disease dataset
input_data = (58,0,0,170,225,1,0,146,1,2.8,1,2,1)

# changing the input_data to numpy array
input_numpy_array = np.asarray(input_data)

# reshape the array for predicting one instance
input_data_reshaped = input_numpy_array.reshape(1,-1)

prediction = classifier.predict(input_data_reshaped)
print("Predicted Label: ", prediction)

if (prediction[0] == 0):
    print('The person does not have a heart disease')
else:
    print('The person have a heart disease')
```

```
Predicted Label:  [0]
The person does not have a heart disease
```

```
import pickle


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


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

Start coding or generate with AI.