In [2]:

```
#Import python packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import svm #Import svm model
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [3]:

```
#Import the heart data
data = pd.read_csv("heart.csv")
```

In [4]:

```
#Display first 5 lines of heart data
data.head()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [5]:
```
#Display basic info about the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [6]:
```
#Separate Feature and Target Matrix
x = data.drop('target',axis = 1)
y = data.target
```

In [7]:
```
# Split dataset into training set and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_state=109) # 70% training and
```

In [8]:

```
# Create a pipeline with feature scaling and SVM classifier
pipeline = make_pipeline(StandardScaler(), svm.SVC(kernel='linear'))

# Define the parameter grid for hyperparameter tuning
param_grid = {'svc__C': [0.1, 1, 5, 10, 50], 'svc__gamma': [1, 0.1, 0.01, 0.001]}

# Perform GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(x_train, y_train)

# Get the best model from GridSearchCV
best_model = grid_search.best_estimator_

# Train the best model on the entire training set
best_model.fit(x_train, y_train)

# Predict the response for the test dataset
y_pred = best_model.predict(x_test)

# Calculate the accuracy of the best model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Evaluate the model
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8636363636363636
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.81      0.85       150
           1       0.83      0.92      0.87       158

    accuracy                           0.86       308
   macro avg       0.87      0.86      0.86       308
weighted avg       0.87      0.86      0.86       308
```

In [20]:

```python
#Create a svm Classifier
ml = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
ml.fit(x_train, y_train)

#Predict the response for test dataset
y_pred = ml.predict(x_test)
```

In [21]:

```python
# Model Accuracy: how often is the classifier correct?
ml.score(x_test,y_test)
```

Out[21]:

```
0.8733766233766234
```

In [ ]:

```python
```

In [ ]:

```python
```