```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection, Formatting and Analysis

```
# Read and save data
data = pd.read_csv("heart.csv")
data.head()
```

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

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

```
# Remove duplicate values
data.drop_duplicates(inplace=True)
```

```
# Display counts of all target values (0- disease present, 1- disease
not present)
data["target"].value_counts()
```

```
target
1    164
0    138
Name: count, dtype: int64
```

```
data.shape
```

```
(302, 14)
```

```
# Display positive and negative correlations
data.corr()
```

```
                age       sex        cp  trestbps      chol
fbs  \
age        1.000000 -0.094962 -0.063107  0.283121  0.207216  0.119492

sex       -0.094962  1.000000 -0.051740 -0.057647 -0.195571  0.046022

cp        -0.063107 -0.051740  1.000000  0.046486 -0.072682  0.096018

trestbps  0.283121 -0.057647  0.046486  1.000000  0.125256  0.178125

chol       0.207216 -0.195571 -0.072682  0.125256  1.000000  0.011428

fbs        0.119492  0.046022  0.096018  0.178125  0.011428  1.000000

restecg   -0.111590 -0.060351  0.041561 -0.115367 -0.147602 -0.083081

thalach   -0.395235 -0.046439  0.293367 -0.048023 -0.005308 -0.007169

exang      0.093216  0.143460 -0.392937  0.068526  0.064099  0.024729

oldpeak    0.206040  0.098322 -0.146692  0.194600  0.050086  0.004514

slope     -0.164124 -0.032990  0.116854 -0.122873  0.000417 -0.058654

ca         0.302261  0.113060 -0.195356  0.099248  0.086878  0.144935

thal       0.065317  0.211452 -0.160370  0.062870  0.096810 -0.032752

target    -0.221476 -0.283609  0.432080 -0.146269 -0.081437 -0.026826


            restecg   thalach     exang   oldpeak     slope
ca  \
age       -0.111590 -0.395235  0.093216  0.206040 -0.164124  0.302261

sex       -0.060351 -0.046439  0.143460  0.098322 -0.032990  0.113060

cp         0.041561  0.293367 -0.392937 -0.146692  0.116854 -0.195356

trestbps -0.115367 -0.048023  0.068526  0.194600 -0.122873  0.099248

chol      -0.147602 -0.005308  0.064099  0.050086  0.000417  0.086878

fbs       -0.083081 -0.007169  0.024729  0.004514 -0.058654  0.144935

restecg    1.000000  0.041210 -0.068807 -0.056251  0.090402 -0.083112

thalach    0.041210  1.000000 -0.377411 -0.342201  0.384754 -0.228311

exang     -0.068807 -0.377411  1.000000  0.286766 -0.256106  0.125377

oldpeak   -0.056251 -0.342201  0.286766  1.000000 -0.576314  0.236560
```

```
slope     0.090402  0.384754 -0.256106 -0.576314  1.000000 -0.092236

ca       -0.083112 -0.228311  0.125377  0.236560 -0.092236  1.000000

thal     -0.010473 -0.094910  0.205826  0.209090 -0.103314  0.160085

target    0.134874  0.419955 -0.435601 -0.429146  0.343940 -0.408992


                thal    target
age         0.065317 -0.221476
sex         0.211452 -0.283609
cp         -0.160370  0.432080
trestbps    0.062870 -0.146269
chol        0.096810 -0.081437
fbs        -0.032752 -0.026826
restecg    -0.010473  0.134874
thalach    -0.094910  0.419955
exang       0.205826 -0.435601
oldpeak     0.209090 -0.429146
slope      -0.103314  0.343940
ca          0.160085 -0.408992
thal        1.000000 -0.343101
target     -0.343101  1.000000
```

```
# Display statistics for data
data.describe()
```

```
              age         sex          cp    trestbps        chol
fbs  \
count  302.00000  302.000000  302.000000  302.000000  302.000000
302.000000
mean    54.42053    0.682119    0.963576  131.602649  246.500000
0.149007
std      9.04797    0.466426    1.032044   17.563394   51.753489
0.356686
min     29.00000    0.000000    0.000000   94.000000  126.000000
0.000000
25%     48.00000    0.000000    0.000000  120.000000  211.000000
0.000000
50%     55.50000    1.000000    1.000000  130.000000  240.500000
0.000000
75%     61.00000    1.000000    2.000000  140.000000  274.750000
0.000000
max     77.00000    1.000000    3.000000  200.000000  564.000000
1.000000

          restecg     thalach       exang     oldpeak       slope
ca  \
```

```
count  302.000000  302.000000  302.000000  302.000000  302.000000
302.000000
mean     0.526490  149.569536    0.327815    1.043046    1.397351
0.718543
std      0.526027   22.903527    0.470196    1.161452    0.616274
1.006748
min      0.000000   71.000000    0.000000    0.000000    0.000000
0.000000
25%      0.000000  133.250000    0.000000    0.000000    1.000000
0.000000
50%      1.000000  152.500000    0.000000    0.800000    1.000000
0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000
1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000
4.000000

             thal      target
count  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

Splitting Data into Training and Testing Sets

```
# Set independent and dependent variables
x = data.drop("target", axis=1)
y = data["target"]

x.shape
```

```
(302, 13)
```

```
# Split Data into training and testing sets using
sklearn.model_selection
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=40)

x_train.shape
```

```
(241, 13)
```

```
x_test.shape
```

```
(61, 13)
```

Feature Scaling

```
sc = StandardScaler()

sc.fit(x_train)
```

```
StandardScaler()
```

```
# Set x_train and x_test to have a standardized distribution
x_train = sc.transform(x_train)
x_test = sc.transform(x_test)

x_train
```

```
array([[-0.76158321,  0.70490738,  0.96436106, ...,  0.95315324,
        -0.67698227, -0.51037721],
       [-0.76158321,  0.70490738, -0.9723974 , ...,  0.95315324,
        -0.67698227, -0.51037721],
       [-1.73786806,  0.70490738,  0.96436106, ...,  0.95315324,
         3.40183588, -0.51037721],
       ...,
       [-0.8700593 ,  0.70490738, -0.9723974 , ..., -0.71141148,
         1.36242681,  1.17456673],
       [ 0.10622555,  0.70490738, -0.9723974 , ..., -0.71141148,
         0.34272227,  1.17456673],
       [ 0.97403431, -1.4186261 , -0.9723974 , ..., -0.71141148,
         1.36242681, -0.51037721]])
```

Creating a Training Model

```
lr = LogisticRegression()

lr.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
# Create predictions using logistic regression from sklearn
y_predictions = lr.predict(x_test)

y_predictions
```

```
array([1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
1,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1],
dtype=int64)
```

```
# The percent accuracy of the training model
accuracy_score(y_test, y_predictions)
```

```
0.9016393442622951
```

Generating Results

```
x_train[1]
```

```
array([-0.76158321,  0.70490738, -0.9723974 , -1.08803518, -
0.77822762,
       -0.42587856,  0.85167763, -0.2877988 , -0.6917569 , -
0.79207469,
        0.95315324, -0.67698227, -0.51037721])
```

```
input_values = (52, 1, 0, 125, 212, 0, 1, 168, 0, 1, 2, 2, 3)

df = np.asarray(input_values)
pred = lr.predict(df.reshape(1, -1))

if pred[0] == 1:
    result = True
else:
    result = False

result
```

```
True
```

```
import pickle
pickle.dump(lr, open('ML_Model.pkl', 'wb'))
```