# Assignment3 Logistic Regression

assignment_3.py

C: > Users > sulim > Downloads > assignment_3.py > ...

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

class Logistic_Regression():

    # Constructor
    def __init__(self, dataset_name, learning_rate, no_of_iterations):
        self.dataset_name = dataset_name
        self.learning_rate = learning_rate
        self.no_of_iterations = no_of_iterations
        self.cost_fun = []  # Cost Array

    # Traning
    def fit(self, X, Y):
        self.m, self.n = X.shape
        self.w = np.zeros(self.n)
        self.b = 0
        self.X = X
        self.Y = Y

        for i in range(self.no_of_iterations):
            self.update_weights()
            cost = self.compute_cost()
            self.cost_fun.append(cost)

        print("Final estimates of b and w are: ", self.b, self.w)

        # Cost function
        print('Initial loss\t:', self.cost_fun[0])
        print('Final loss\t:', self.cost_fun[-1])
        self.plot_cost_fun()

    # Gradient Descent
    def update_weights(self):

        z = np.dot(self.X, self.w) + self.b
        predictions = self.sigmoid(z)

        # Differentiation
        dw = (1/self.m) * np.dot(self.X.T, (predictions - self.Y))
        db = (1/self.m) * np.sum(predictions - self.Y)

        #update
        self.w -= self.learning_rate * dw
        self.b -= self.learning_rate * db

    def sigmoid(self, z):
        z = 1 / (1 + np.exp(-z))
        return z

    # Log Loss
    def compute_cost(self):
        z = np.dot(self.X, self.w) + self.b
        predictions = self.sigmoid(z)
        cost = -(1/self.m) * np.sum(self.Y * np.log(predictions) + (1 - self.Y) * np.log(1 - predictions))
        return cost

    def predict(self, X):
        Y_pred = self.sigmoid(np.dot(X, self.w) + self.b)
        Y_pred = np.where(Y_pred > 0.5, 1, 0)
        return Y_pred

```

assignment_3.py

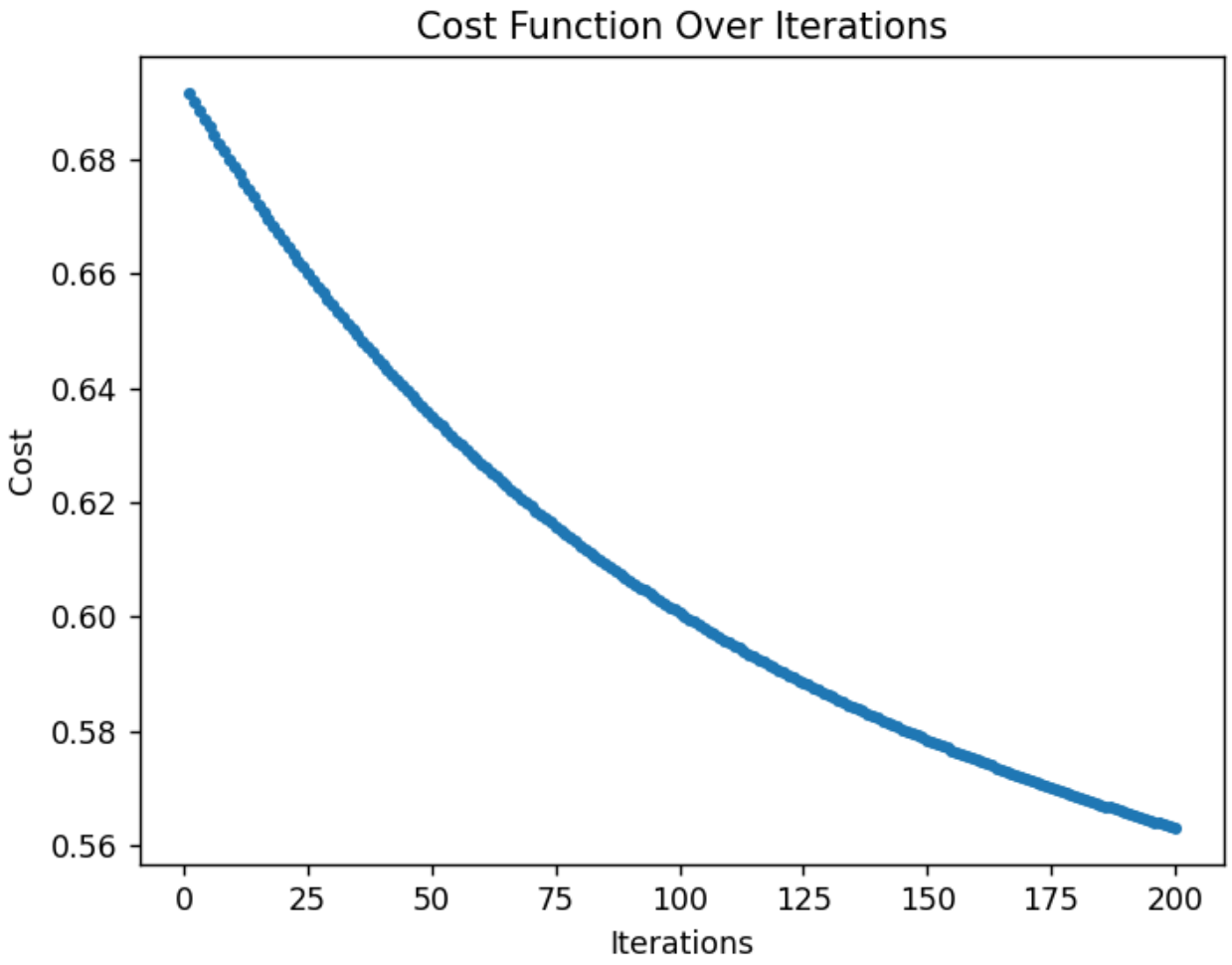
C: > Users > sulim > Downloads > assignment_3.py > ...

```python

    # Log Loss
    def compute_cost(self):
        z = np.dot(self.X, self.w) + self.b
        predictions = self.sigmoid(z)
        cost = -(1/self.m) * np.sum(self.Y * np.log(predictions) + (1 - self.Y) * np.log(1 - predictions))
        return cost

    def predict(self, X):
        Y_pred = self.sigmoid(np.dot(X, self.w) + self.b)
        Y_pred = np.where(Y_pred > 0.5, 1, 0)
        return Y_pred

    def load_dataset(self):
        heart_dataset = pd.read_csv(self.dataset_name)
        heart_dataset.famhist.replace(('Present', 'Absent'), (0, 1), inplace=True)
        X = heart_dataset.drop(columns=['row.names', 'chd'], axis=1)
        y = heart_dataset["chd"]
        X = X.values
        y = y.values
        return X, y

    def sklearn_LR(self, X_train, X_test, Y_train, Y_test):
        lr = LogisticRegression()
        lr.fit(X_train, Y_train)
        y_pred = lr.predict(X_test)
        testing_data_accuracy = accuracy_score(Y_test, y_pred)
        print('sklearn classifier: Accuracy score of the testing data : ', testing_data_accuracy)

    def plot_cost_fun(self):
        plt.plot(range(1, self.no_of_iterations + 1), self.cost_fun, marker='.')
        plt.title('Cost Function Over Iterations')
        plt.xlabel('Iterations')
        plt.ylabel('Cost')
        plt.show()


if __name__ == "__main__":
    dataset_name = "c:\\Users\\sulim\\Downloads\\Heart.csv"
    learning_rate = 0.01
    no_of_iterations = 200
    classifier = Logistic_Regression(dataset_name, learning_rate, no_of_iterations)
    X, y = classifier.load_dataset()
    scaler = StandardScaler()
    scaler.fit(X)
    X = scaler.transform(X)
    X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=2)

    classifier.fit(X_train, Y_train)
    y_pred = classifier.predict(X_test)
    testing_data_accuracy = accuracy_score(Y_test, y_pred)
    print('Accuracy score of the testing data : ', testing_data_accuracy)

    classifier.sklearn_LR(X_train, X_test, Y_train, Y_test)

    input_data = [124, 4.00, 12.42, 31.29, 1, 54, 23.23, 2.06, 42]
    input_data_as_numpy_array = np.asarray(input_data)
    input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)
    standardized_data = scaler.transform(input_data_reshaped)

    prediction = classifier.predict(standardized_data)

    if prediction[0] == 0:
        print('The person is not coronary heart disease')
    else:
        print('The person has coronary heart disease')
```

# Best inputs from the question:

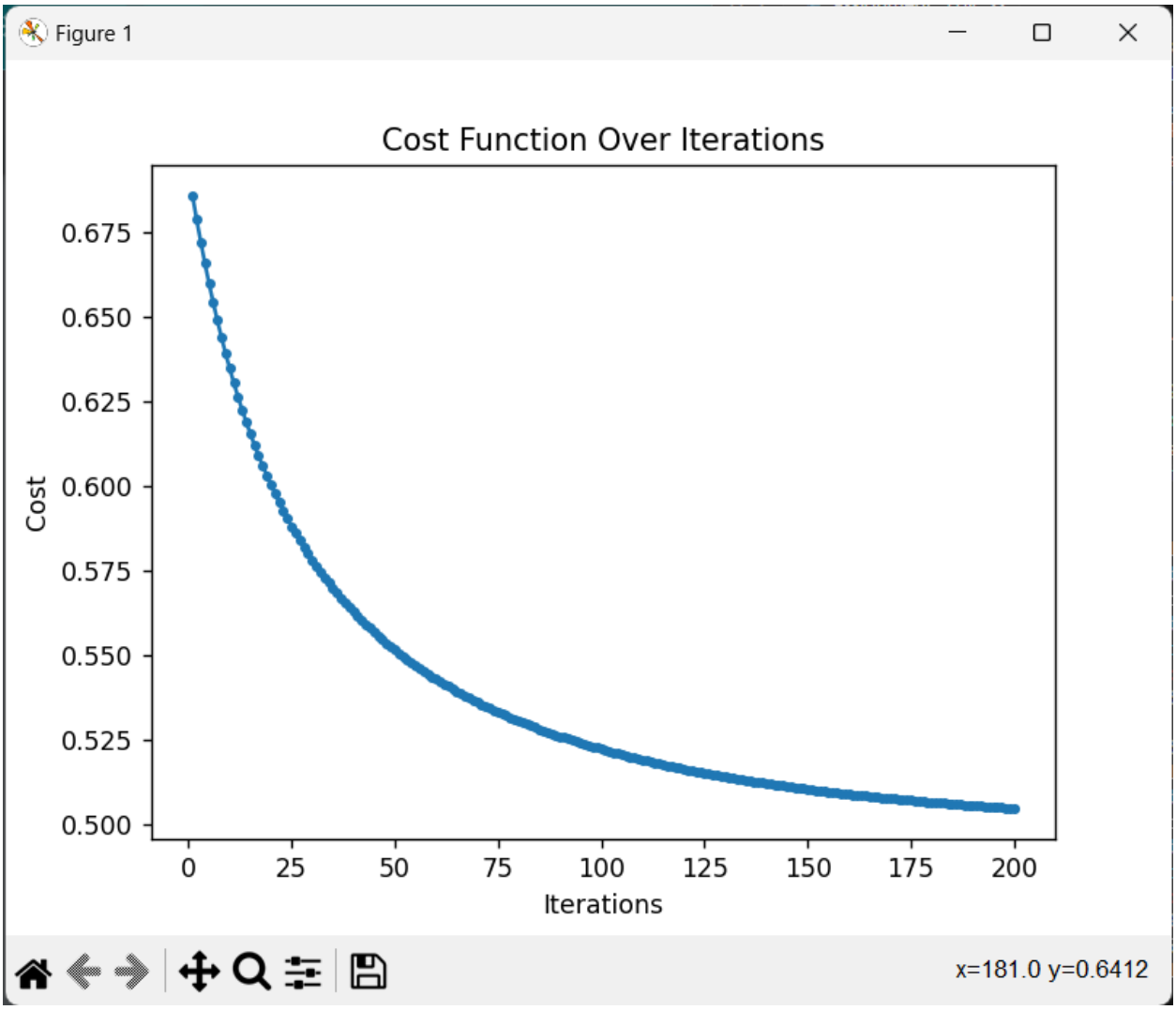
learning_rate = 0.01

no_of_iterations = 200



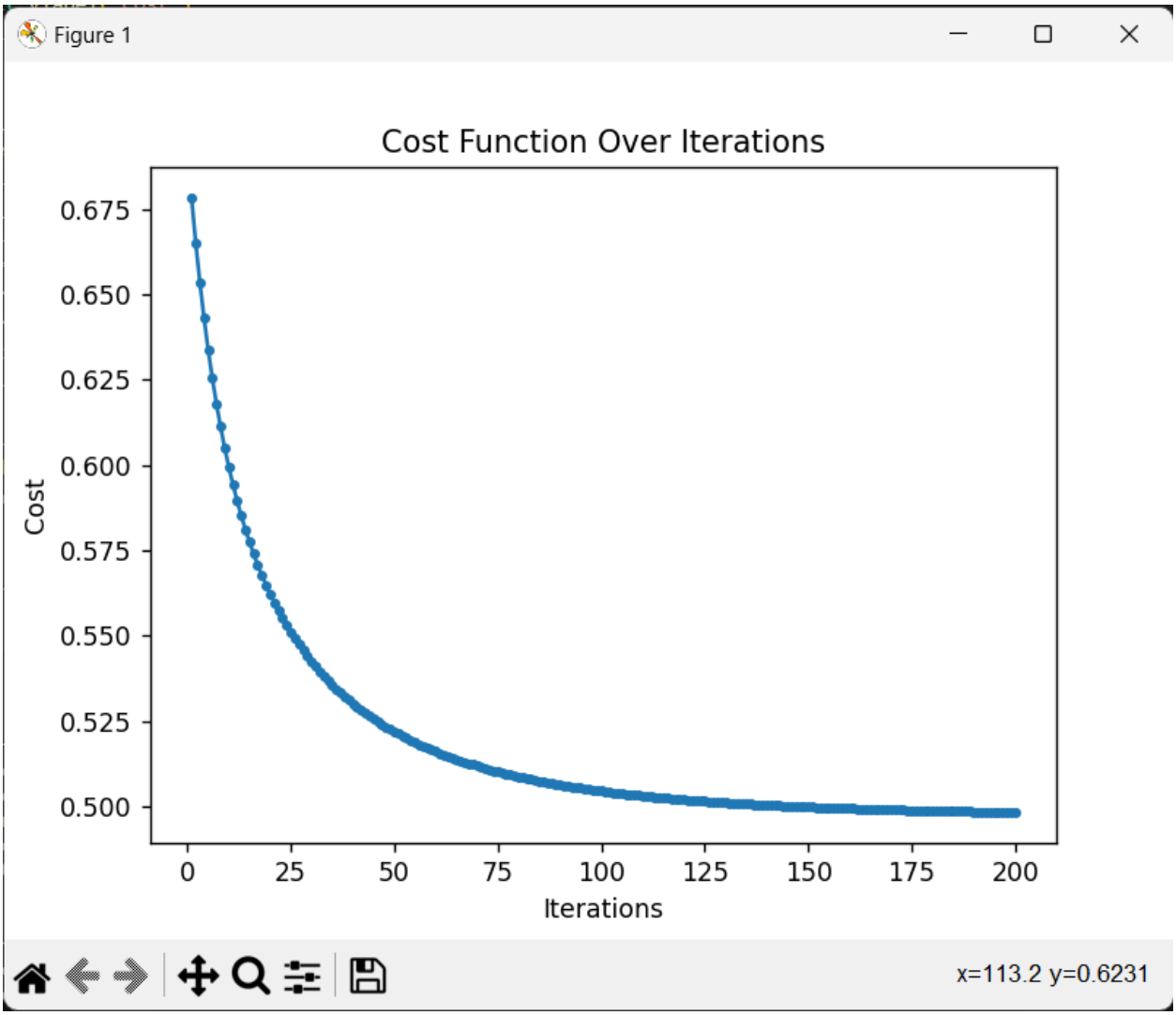

```
PS C:\Users\sulim> python -u "c:\Users\sulim\Downloads\assignment_3.py"
Final estimates of b and w are:  -0.2322948376833924 [ 0.08459305  0.21284346  0.16957313  0.13797504 -0.168603    0.07177381
  0.02002443  0.02174723  0.22796475]
Initial loss    : 0.6916306519720461
Final loss      : 0.5630774601264
Accuracy score of the testing data :  0.6559139784946236
sklearn classifier: Accuracy score of the testing data :  0.6451612903225806
The person has coronary heart disease
```

learning rate = 0.01



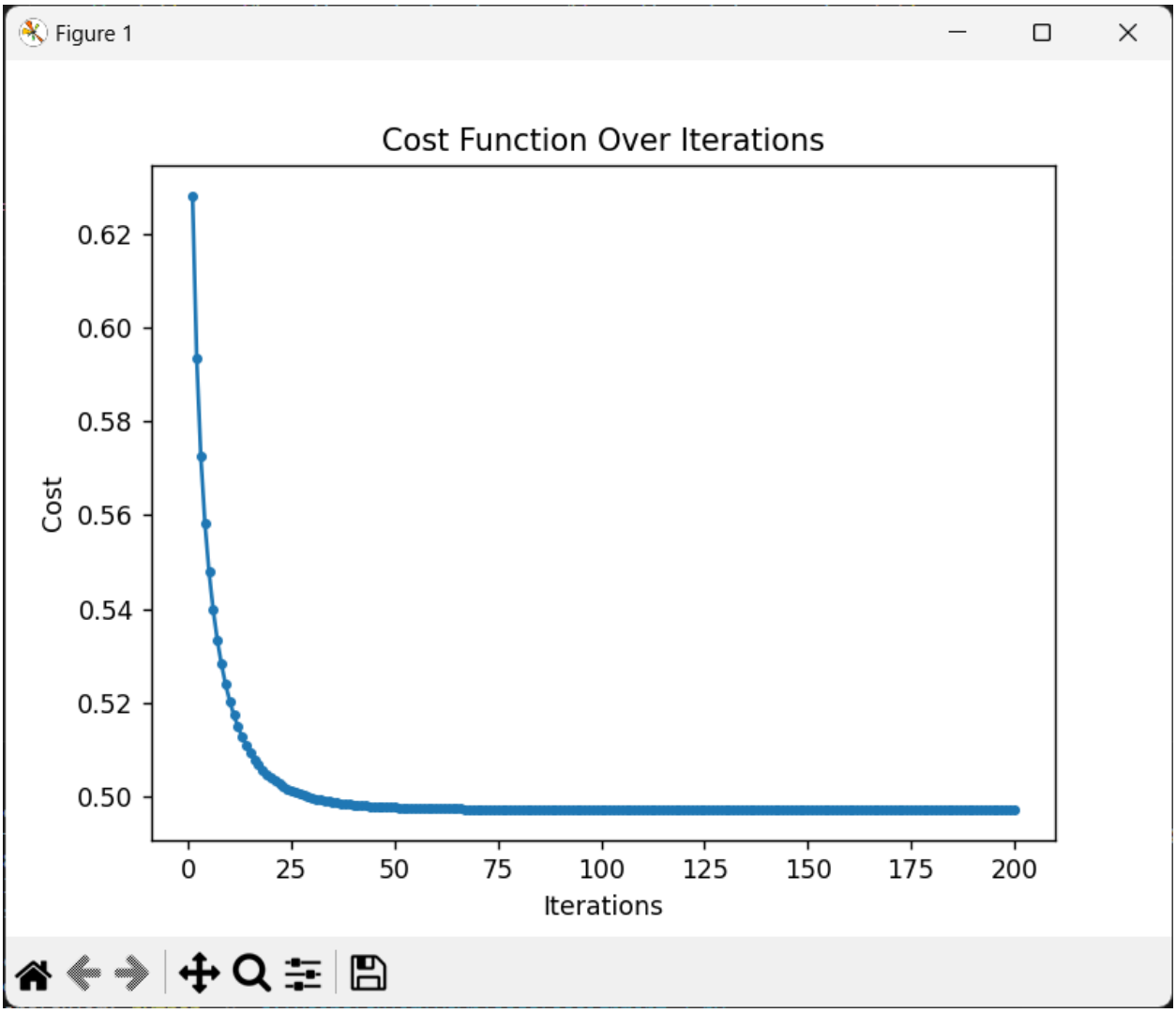

```
Final estimates of b and w are:  -0.6336274079860791 [ 0.09369933  0.45793394  0.3304137   0.18732974 -0.35489961  0.25657642
 -0.12704601  0.00642576  0.46937039]
Initial loss    : 0.6856541361532598
Final loss      : 0.5045538057185711
Accuracy score of the testing data :  0.6559139784946236
sklearn classifier: Accuracy score of the testing data :  0.6451612903225806
The person has coronary heart disease
```

learning rate = 0.05



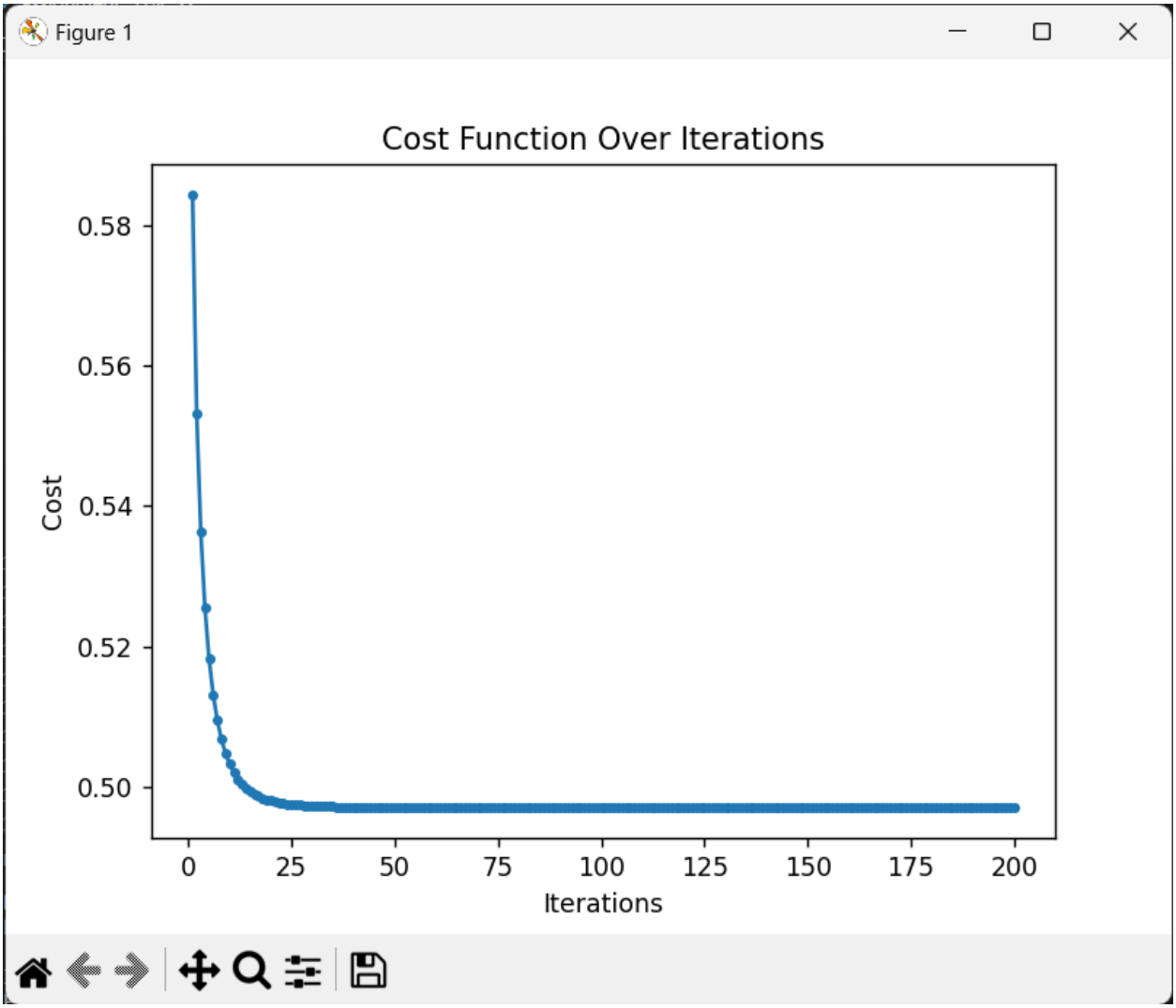

```
Final estimates of b and w are:  -0.853475051264791 [ 0.05821073  0.52707969  0.40063137  0.27720767 -0.41194364  0.43391011
 -0.29914796  0.00879877  0.66069812]
Initial loss     : 0.6281958211874307
Final loss       : 0.49707485591133244
Accuracy score of the testing data :  0.6559139784946236
sklearn classifier: Accuracy score of the testing data :  0.6451612903225806
The person has coronary heart disease
```

learning rate = 0.5




```
Final estimates of b and w are:  -0.7734509602241763 [ 7.92998250e-02  5.21037559e-01  3.78212746e-01  2.11533465e-01
 -3.99405503e-01  3.58784004e-01 -2.12066373e-01  2.41819935e-04
  5.76643936e-01]
Initial loss    : 0.6783849683017921
Final loss      : 0.4982401711142727
Accuracy score of the testing data :  0.6559139784946236
sklearn classifier: Accuracy score of the testing data :  0.6451612903225806
The person has coronary heart disease
```

learning rate = 0.1




```
Final estimates of b and w are:  -0.8535507577795765 [ 0.05817623  0.52728662  0.40021322  0.2836978  -0.412031    0.43433799
 -0.30378521  0.00878373  0.65776339]
Initial loss    : 0.584308274533989
Final loss      : 0.497073785464207
Accuracy score of the testing data :  0.6559139784946236
sklearn classifier: Accuracy score of the testing data :  0.6451612903225806
The person has coronary heart disease
```

learning rate = 1