MACHINE LEARNING
ASSIGNMENT 02

NAME:

MUHAMMAD MUAAZ SHOAIB

REG. NO:

FA20-BCS-074

CLASS:

BCS-6B

SUBMITTED TO:

DR. ZAHOOR

DATE:

10-APR-2023

Part 1: Linear Regression

1. Implement linear regression using gradient descent to predict the MEDV. Your implementation should include the following:
    - A function to calculate the cost function.
    - A function to perform gradient descent.
    - A function to predict the MEDV given a set of input features.

```
# A function to calculate the cost function
import numpy as np

def cost(features, target, theta):
    total_number_of_samples = len(target)
    predictions = features.dot(theta)
    cost = 1/(2 * total_number_of_samples) *
np.sum(np.square(predictions - target))
    return cost

# A function to perform gradient descent.
import numpy as np

def gradient_descent(features, target, theta, alpha,
number_of_iterations):
    total_number_of_samples = len(target)
    cost_history = np.zeros(number_of_iterations)
    for i in range(number_of_iterations):
        predictions = features.dot(theta)
        loss = predictions - target

        theta = theta - alpha * (1 / total_number_of_samples) *
(features.T.dot(loss))
        cost_history[i] = cost(features, target, theta)

    return theta, cost_history

# A function to predict the MEDV given a set of input features.
def predict_medv(features, theta):
     predictions = features.dot(theta)
     return predictions
```

2. Split the dataset into a training set and a test set. Use 80% of the data for training and the remaining 20% for testing.

```
# Downlaod the `Boston Housing Dataset`
!wget
https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing
.csv

import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('BostonHousing.csv')

features = df.drop(['medv'], axis=1)
```

```
target = df['medv']

features_train, features_test, target_train, target_test =
train_test_split(features,
target,
test_size=0.2)

len(features_train), len(features_test), len(target_train),
len(target_test)
```

```
--2023-04-12 20:12:43--  https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 35735 (35K) [text/plain]
Saving to: 'BostonHousing.csv.13'

BostonHousing.csv.1 100%[===================>]  34.90K  --.-KB/s    in 0.003s

2023-04-12 20:12:43 (11.2 MB/s) - 'BostonHousing.csv.13' saved [35735/35735]

(404, 102, 404, 102)
```

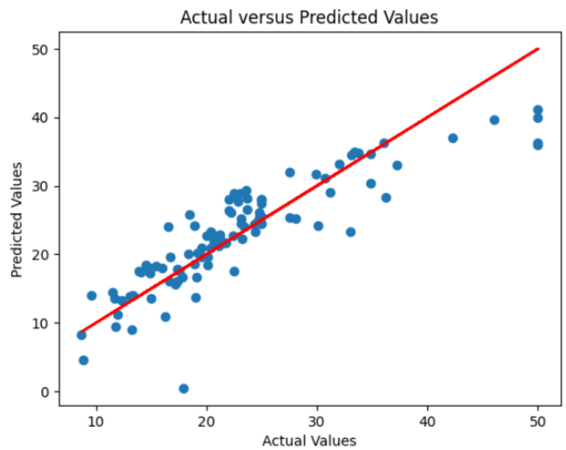
3. Train your linear regression model on the training set and evaluate its performance on the test set using the mean squared error (MSE) metric. Report the MSE value.

```
from sklearn.metrics import mean_squared_error

# Normalize to get the same scale of all values
features_train_normalize = (features_train - np.mean(features_train,
axis=0))/np.std(features_train, axis=0)
features_train_normalize =
np.hstack((np.ones((features_train_normalize.shape[0],1)),features_trai
n_normalize))

theta = np.zeros(features_train_normalize.shape[1])

theta, cost_hist = gradient_descent(features_train_normalize,
                                    target_train,
                                    theta,
                                    alpha=0.01,
                                    number_of_iterations=1000)

# Mean Squared Error
features_test_normalize = (features_test - np.mean(features_train,
axis=0))/np.std(features_train, axis=0)
features_test_normalize =
np.hstack((np.ones((features_test_normalize.shape[0],1)),features_test_
normalize))
target_predictions = features_test_normalize.dot(theta)
mean_squared_error = mean_squared_error(target_test,
target_predictions)

print('Mean Squared Error: ', mean_squared_error)
```

```
Mean Squared Error:  18.990003274355246
```

4. Plot the predicted values vs. the actual values on the test set in a scatter plot.

```
import matplotlib.pyplot as plt

# Plot the predicted values versus the actual values on the test set
plt.scatter(target_test, target_predictions)
plt.plot(target_test, target_test, color='red')
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual versus Predicted Values")
plt.show()
```



Part 2: Logistic Regression

1. Implement logistic regression using gradient descent to predict whether a suburb has a high or low MEDV. To do this, you will need to binarize the MEDV column by setting a threshold value. If the MEDV is greater than or equal to the threshold value, the suburb is classified as having a high MEDV, otherwise it is classified as having a low MEDV.

```
threshold = 21.2
target_train = (target_train >= threshold).astype(int)
target_test = (target_test >= threshold).astype(int)

threshold, len(target_train), len(target_test)
```

2. Split the dataset into a training set and a test set. Use 80% of the data for training and the remaining 20% for testing.

```
# Downlaod the `Boston Housing Dataset`
!wget
https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing
.csv

import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('BostonHousing.csv')
```

```
features = df.drop(['medv'], axis=1)
target = df['medv']

features_train, features_test, target_train, target_test =
train_test_split(features,

target,

test_size=0.2)

len(features_train), len(features_test), len(target_train),
len(target_test)
```

```
--2023-04-12 20:12:43--  https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 35735 (35K) [text/plain]
Saving to: 'BostonHousing.csv.13'

BostonHousing.csv.1 100%[===================>]  34.90K  --.-KB/s    in 0.003s

2023-04-12 20:12:43 (11.2 MB/s) - 'BostonHousing.csv.13' saved [35735/35735]

(404, 102, 404, 102)
```

3. Train your logistic regression model on the training set and evaluate its performance on the test set using the accuracy metric. Report the accuracy value.

```
import numpy as np

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

NUMBER_OF_ITERATIONS = 1000
ALPHA = 0.01

total_number_of_samples = len(target)

# Normalize to get the same scale of all values
features_train_normalize = (features_train - np.mean(features_train,
axis=0))/np.std(features_train, axis=0)
features_train_normalize =
np.hstack((np.ones((features_train_normalize.shape[0],1)),features_trai
n_normalize))

for i in range(NUMBER_OF_ITERATIONS):
  z = np.dot(features_train_normalize, theta)
  h = sigmoid(z)
  gradient = np.dot(features_train_normalize.T, (h - target_train)) /
total_number_of_samples
  theta -= ALPHA * gradient

features_test_normalize = (features_test - np.mean(features_train,
axis=0))/np.std(features_train, axis=0)
features_test_normalize =
np.hstack((np.ones((features_test_normalize.shape[0],1)),features_test_
```

```
normalize))

z_test = np.dot(features_test_normalize, theta)
h_test = sigmoid(z_test)
target_predictions = (h_test >= 0.5).astype(int)

from sklearn.metrics import accuracy_score, confusion_matrix
accuracy = accuracy_score(target_test, target_predictions)
print("Accuracy: ", accuracy)
```

```
Accuracy:  0.5392156862745098
```

4. Plot the decision boundary of your logistic regression model in a scatter plot that shows the data points with different colors for high and low MEDV.

```
import matplotlib.pyplot as plt

# Plot the predicted values versus the actual values on the test set
plt.scatter(z_test, target_predictions)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual versus Predicted Values")
plt.show()
```