COMSATS University Attock Campus
Department of Computer Science
Spring 2023 Assignment-2 **Course: - ML**

Program: BS-CS / BS-SE **Due Dated: 11 Apr 2023**

Name:- _____Reg. No:- _____

Note:- Must come with your laptop on April 11, 2023 with the implementation

Programming Assignment: Linear and Logistic Regression

In this assignment, you will be implementing linear regression and logistic regression from scratch using Python. You will be provided with a dataset and you will use these models to perform regression analysis on the dataset.

Dataset

The dataset that you will be using is the Boston Housing Dataset. This dataset contains information about the housing values in suburbs of Boston. Each row in the dataset represents a suburb, and there are 14 columns:

- **CRIM:** Per capita crime rate by town.
- **ZN:** Proportion of residential land zoned for lots over 25,000 square feet.
- **INDUS:** Proportion of non-retail business acres per town.
- **CHAS:** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- **NOX:** Nitric oxides concentration (parts per 10 million).
- **RM:** Average number of rooms per dwelling.
- **AGE:** Proportion of owner-occupied units built prior to 1940.
- **DIS:** Weighted distances to five Boston employment centres.
- **RAD:** Index of accessibility to radial highways.
- **TAX:** Full-value property-tax rate per $10,000.
- **PTRATIO:** Pupil-teacher ratio by town.
- **B:** 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- **LSTAT:** % lower status of the population.
- **MEDV:** Median value of owner-occupied homes in $1000's.

The goal of this assignment is to predict the median value of owner-occupied homes (MEDV) using the other 13 features.

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.
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.
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.

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

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.
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.
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.
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.

Submission

You should submit a report that includes the following:

1. A brief introduction to linear and logistic regression.
2. Your code for linear regression and logistic regression.
3. A description of the dataset and the preprocessing steps that you have performed.

Solution to Programming Assignment: Linear and Logistic Regression

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.

```
python
def compute_cost(X, y, theta):
    m = len(y)
    predictions = X.dot(theta)
    square_errors = (predictions - y)**2
    J = 1/(2*m) * np.sum(square_errors)
    return J
```

A function to perform gradient descent.

```
python
def gradient_descent(X, y, theta, alpha, num_iters):
    m = len(y)
    J_history = np.zeros((num_iters,1))

    for i in range(num_iters):
        predictions = X.dot(theta)
        errors = predictions - y
        delta = (1/m) * (X.T.dot(errors))
        theta = theta - alpha * delta
        J_history[i] = compute_cost(X, y, theta)

    return theta, J_history
```

A function to predict the MEDV given a set of input features.

```
python
def predict_medv(X, theta):
    predictions = X.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.

```
python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

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.

```
python
m = len(y_train)
n = X_train.shape[1]
theta = np.zeros((n,1))

alpha = 0.01
num_iters = 1000

X_train = np.hstack((np.ones((m,1)), X_train))

theta, J_history = gradient_descent(X_train, y_train, theta, alpha, num_iters)

X_test = np.hstack((np.ones((len(y_test),1)), X_test))
predictions = predict_medv(X_test, theta)

mse = np.mean((predictions - y_test)**2)
print("Mean Squared Error:", mse)
```

Output:

```
javascript
Mean Squared Error: 33.44897999767678
```

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

```
python
import matplotlib.pyplot as plt

plt.scatter(y_test, predictions)
plt.xlabel("Actual MEDV")
plt.ylabel("Predicted MEDV")
plt.show()
```

Output:

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.

```
python
threshold = 21.2
y_train = (y_train >= threshold).astype(int)
y_test = (y_test >= threshold).astype(int)
```

A function to calculate the sigmoid function.

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