ENGR 421: Introduction to Machine Learning
Fall 2020 – Homework 2
Ege Yelken – 61742

The aim of this assignment is to implement a discrimination by regression algorithm on a multiclass data set. I have followed the steps below:

1. Divided the data set into two by assigning the first 25 image of each class to the training set and remaining 14 images to the test set:

```
train_x1 = imagesdf.iloc[0:25]
train_x2 = imagesdf.iloc[39:64]
train_x3 = imagesdf.iloc[78:103]
train_x4 = imagesdf.iloc[117:142]
train_x5 = imagesdf.iloc[156:181]
train_x = pd.concat([train_x1, train_x2, train_x3, train_x4, train_x5])
```

```
test_x1 = imagesdf.iloc[25:39]
test_x2 = imagesdf.iloc[64:78]
test_x3 = imagesdf.iloc[103:117]
test_x4 = imagesdf.iloc[142:156]
test_x5 = imagesdf.iloc[181:195]
test_x = pd.concat([test_x1, test_x2, test_x3, test_x4, test_x5])
```

```
train_y1 = labeldf.iloc[0:25]
train_y2 = labeldf.iloc[39:64]
train_y3 = labeldf.iloc[78:103]
train_y4 = labeldf.iloc[117:142]
train_y5 = labeldf.iloc[156:181]
train_y = pd.concat([train_y1, train_y2, train_y3, train_y4, train_y5])
```

```
test_y1 = labeldf.iloc[25:39]
test_y2 = labeldf.iloc[64:78]
test_y3 = labeldf.iloc[103:117]
test_y4 = labeldf.iloc[142:156]
test_y5 = labeldf.iloc[181:195]
test_y = pd.concat([test_y1, test_y2, test_y3, test_y4, test_y5])
```

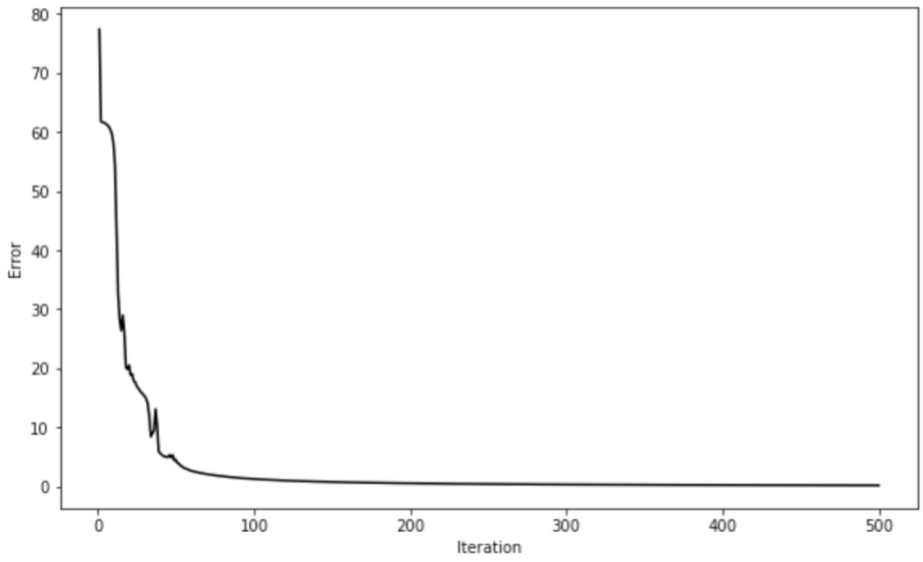
2. Written the sigmoid function for learning a discrimination by regression algorithm with the given parameters:

```
def sigmoid(X,W,w0):
    sigmoid = (1/(1 + np.exp(-1*(np.dot(X,W)+w0))))
    return sigmoid

def update_W(X,y_truth,y_pred):
    a = (y_truth - y_pred) * y_pred * (1 - y_pred)
    return (np.dot(-1*(np.transpose(X)),a))
def update_w0(y_truth,y_pred):
    a =((y_truth - y_pred) * y_pred * (1- y_pred))
    col_sums = a.sum(axis = 0)
    return -1 * col_sums

eta = 0.01
epsilon = 1e-3

np.random.seed(421)
W = np.random.uniform(low = -0.01, high = 0.01, size = (train_x.shape[1], 5))
w0 = np.random.uniform(low = -0.01, high = 0.01, size = (1, 5))
```

3. Drew the objective function values throughout the iterations:

```
plt.figure(figsize = (10, 6))
plt.plot(range(1, iteration + 1), objective_values, "k-")
plt.xlabel("Iteration")
plt.ylabel("Error")
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



4. Encoded labels ‘A’, ‘B’, ‘C’, ‘D’ and ‘E’ as 0, 1, 2, 3, 4 respectively in order to calculate the confusion matrix:

```
train_y = np.where(train_y == 'A', 0, train_y)
train_y = np.where(train_y == 'B', 1, train_y)
train_y = np.where(train_y == 'C', 2, train_y)
train_y = np.where(train_y == 'D', 3, train_y)
train_y = np.where(train_y == 'E', 4, train_y)

test_y = np.where(test_y == 'A', 0, test_y)
test_y = np.where(test_y == 'B', 1, test_y)
test_y = np.where(test_y == 'C', 2, test_y)
test_y = np.where(test_y == 'D', 3, test_y)
test_y = np.where(test_y == 'E', 4, test_y)
```

5. Calculated the confusion matrix for both training set and test set using the *confusion_matrix* method from **sklearn**:

```
train_confusion_matrix = confusion_matrix(y_predicted.argmax(axis=1), encoded_y_train.argmax(axis=1))
test_confusion_matrix = confusion_matrix(y_predicted1.argmax(axis=1), encoded_y_test.argmax(axis=1))


print(train_confusion_matrix)
print('\n')
print(test_confusion_matrix)
```

```
[[25  0  0  0  0]
 [ 0 25  0  0  0]
 [ 0  0 25  0  0]
 [ 0  0  0 25  0]
 [ 0  0  0  0 25]]


[[14  4  1  0  2]
 [ 0 10  0  0  1]
 [ 0  0 13  0  0]
 [ 0  0  0 14  0]
 [ 0  0  0  0 11]]
```