University of Regina

ENSE 865: Applied Machine Learning

PROGRAMMING ASSIGNMENT 3

SUBMITTED BY: Hitarthi Gandhi (200428713)

Question1: Implement Binary Logistic Regression from scratch. Write functions for normalizing dataset, predicting output, estimating and printing logistic regression coefficients, and calculating model accuracy.

Answer:

Here, we are implementing binary logistic regression function from the starting. Logistic regression function is statical model and mostly based on binary dependent function. For this we created sigmoid function, cost function, gradient descent function, fit function prediction function, score function and plot function.

Below function takes logistic regression, learning rate and number of iteration as an argument. Learning rate is hyperparameter and it controls the change in the model.

```
def __init__(Logistic_reg, Learning_rate=0.01, number_of_iteration=100):

    Logistic_reg.Learning_rate = Learning_rate

    Logistic_reg.number_iteration = number_of_iteration
```

After initializing the main function we creates function for logistic regression. Below sigmoid function calculating the sigmoid value of the given parameters and return the value as an output. .

```
def sigm_fun(Logistic_reg, x):

    output = 1 / (1 + np.exp(-x))

    return output
```

After calculating the sigmoid value we are defining error function. Cost function is also known as error function. Cost function calculating the cost value of the logistic regression and return cost value.

```
def cost_fun(Logistic_reg,h,model_weights, y): # The fuctions calculates the cost value

    m = len(y)

    cost = (1 / m) * (np.sum(-y.T.dot(np.log(h)) - (1 - y).T.dot(np.log(1 - h))))

    return cost
```

After getting the cost value of the function using gradient descent method below function computing the model weights (Theta). By using Gradient decent function our main goal is to minimizing the error and changing the theta values.

```
def gradient_fun(Logistic_reg,X,h,model_weights,y,m):
    gradient_output = np.dot(X.T, (h - y)) / m
    model_weights -= Logistic_reg.Learning_rate * gradient_output
    return model_weights
```

We need to use predict data to build our model so below function first computing the optimal weights for the model based on predicted data. Every time function takes different values.

```
def fit(Logistic_reg, X, y):
    print("Running...Please Wait")
    Logistic_reg.model_weights = []
    Logistic_reg.cost = []
    X = np.insert(X, 0, 1, axis=1)
    m = len(y)
    for i in np.unique(y):
        y_onevsall = np.where(y == i, 1, 0)
        model_weights = np.zeros(X.shape[1])
        cost = []
        for _ in range(Logistic_reg.number_iteration):
            z = X.dot(model_weights)
            h = Logistic_reg.sigm_fun(z)
            model_weights = Logistic_reg.gradient_fun(X,h,model_weights,y_onevsall,m)
            cost.append(Logistic_reg.cost_fun(h,model_weights,y_onevsall))
        Logistic_reg.model_weights.append((model_weights, i))
        Logistic_reg.cost.append((cost,i))
```

```
    return Logistic_reg,Logistic_reg.model_weights
```

After calcultaing the model weights value we need to classify the invidual features based on our needs.Below function classfied the individal features for predicted value.

```
def predict_fun(Logistic_reg, X):

    X = np.insert(X, 0, 1, axis=1)

    X_predicted = [max((Logistic_reg.sigm_fun(i.dot(model_weights)), c) for model_weights, c in Logistic_reg.model_weights)[1] for i in X ]

    return X_predicted
```

Below function comparing the values between the predicted label and actual label.It gives model Accuracy.To find the model performace based on the data.

```
def score_fun(Logistic_reg,X, y):

    score = sum(Logistic_reg.predict_fun(X) == y) / len(y)

    return score
```

Below funtion plotting cost function graph for its different value.This function plots converge garph.

```
def plt_cost(Logistic_reg,costh):

    for cost,c in costh   :

        plt.plot(range(len(cost)),cost,'g')

        plt.title("Gradient Graph for species -" + str(c) +" vs All")

        plt.xlabel("Number of Iterations")

        plt.ylabel("Cost(Error)")

        plt.show()
```

Below function normalizing our data.

```
    Normalizer = StandardScaler()
```

X= Normalizer.fit_transform(X)

As an output of the 1st question we are printing the model coefficient values for all seven species. We already define function for model coefficients.

print("Model Coefficient:", model_coefficients)

Output:

```
  Model Coefficient for Species name Bream:
  [(array([-2.98545503, -0.25564792, -0.5183716 , -0.40370878,  0.467
75187,4.05699127, -0.74824008]), 'Bream'),

  Model Coefficient for Species name Parkki:
  (array([-3.64962137, -0.70504343, -0.34087934, -0.33760876, -0.2967
5485,1.44259392, -0.45825185]), 'Parkki'),

  Model Coefficient for Species name Perch:
  (array([-0.9634782 , -0.68959499, -0.13381211,  0.14499624, -1.7485
496,-1.577866  ,  3.40216198]), 'Perch'),

Model Coefficient for Species name Pike:
  (array([-3.48405046, -0.47328308,  1.58468675,  1.52020102,  1.5212
0044,-1.69566876, -1.22338364]), 'Pike'),

Model Coefficient for Species name Roach:
  (array([-2.24217104, -2.17293762, -0.15876588, -0.21061889,  0.4758
0902,0.17888896,  0.96377876]), 'Roach'),

Model Coefficient for Species name Smelt:
  (array([-4.41074303,  1.1894529 , -0.4354776 , -0.61880896, -0.4947
9979,-1.00564177, -1.94241069]), 'Smelt'),

Model Coefficient for Species name Whitefish:
  (array([-3.1517422 , -0.16729033, -0.3008254 , -0.22663768, -0.2049
7685,-0.0085274 ,  1.4081482 ]), 'Whitefish')]
```

Question2: The data contains Seven species of Fish. Learn Seven 'One vs All' bina ry classification models using functions in Q1. For each model show test/train accuracy. Use followings as input_features: Weight, Length1,Length2,Length3,Width,Height.

Answer:

In this we are using onevsall classification. For this we are using above functions. One-vs-all is a strategy that involves training N distinct binary classifiers, each designed to recognize a specific class. Here we are taking Weight,Lenght1,Length2,Length3,Width and Height as an input features and Every time model plots onevsall graph for one inputfeature and one species.

for _ in range (6):

X_train,X_test,y_train,y_test = train_test_split(X,y_data,test_size = 0.6)

log_reg, model_coefficients= Logistic_Regression(number_of_iteration=10000).fit(X_train, y_train)

prediction_output = log_reg.predict_fun(X_test)

Train_Accuracy = log_reg.score_fun(X_train,y_train)

Test_Accuracy = log_reg.score_fun(X_test,y_test)

print("\n")

print("Train accuracy of the model is: ",Train_Accuracy)

print("Test accuracy of the model is: ",Test_Accuracy)

print("\n")

As an result We are plot the onevsall graph for each species and individual input features.

log_reg.plt_cost(log_reg.cost)

Output:

Train and Test Accuracy for Weight:

```
Train accuracy of the model is:  0.7777777777777778
Test accuracy of the model is:  0.7916666666666666
```

Train and Test Accuracy for Length1:

```
Train accuracy of the model is:  0.8253968253968254
Test accuracy of the model is:  0.7291666666666666
```

Train and Test Accuracy for Length2:

```
Train accuracy of the model is:  0.8253968253968254
Test accuracy of the model is:  0.7395833333333334
```

Train and Test Accuracy for Length3:

```
Train accuracy of the model is:  0.8412698412698413
Test accuracy of the model is:  0.78125
```

Train and Test Accuracy for Height:

```
Train accuracy of the model is:  0.7619047619047619
Test accuracy of the model is:  0.8333333333333334
```

Train and Test Accuracy for Width:

```
Train accuracy of the model is:  0.7777777777777778
Test accuracy of the model is:  0.7395833333333334
```

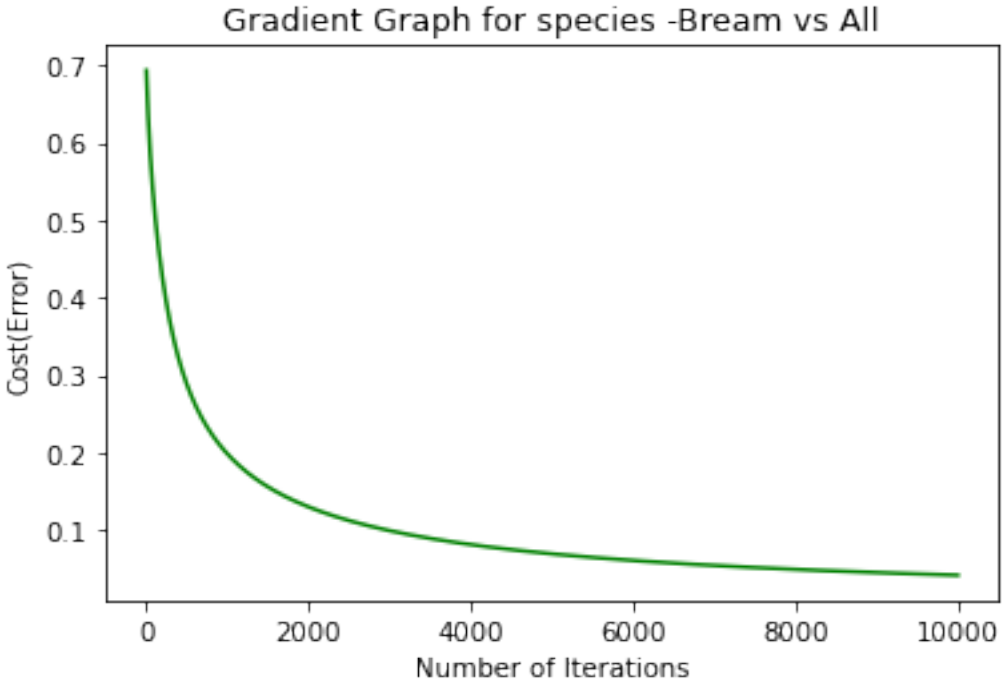

Fig1. Bream Vs All

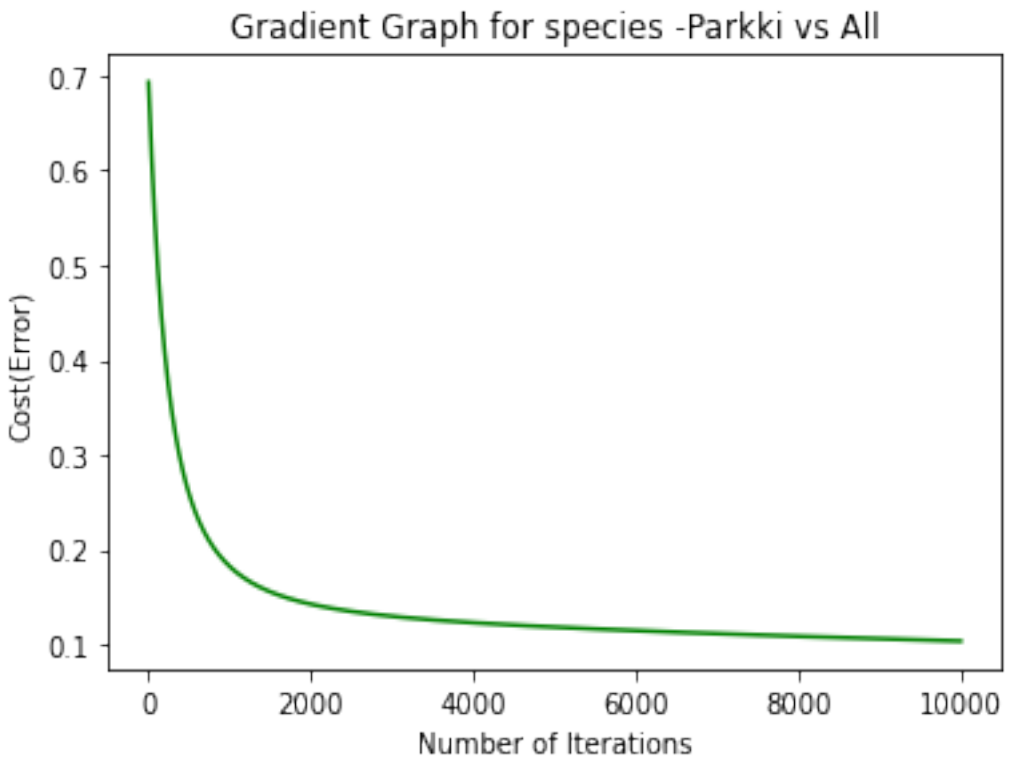

Fig2. Parkki vs All

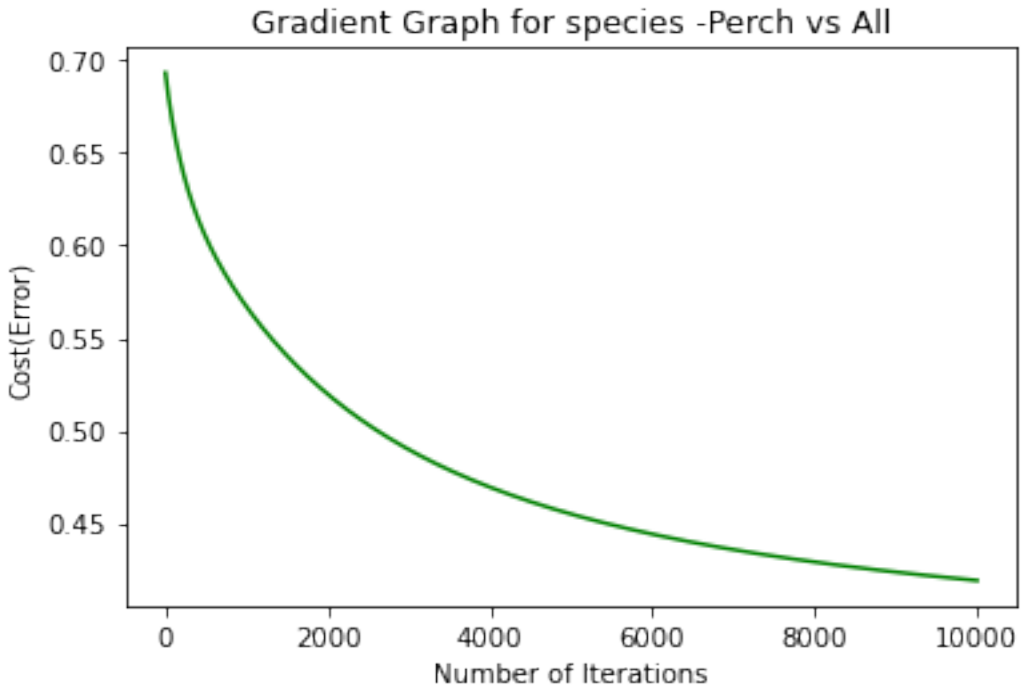

Fig3. Perch vs All

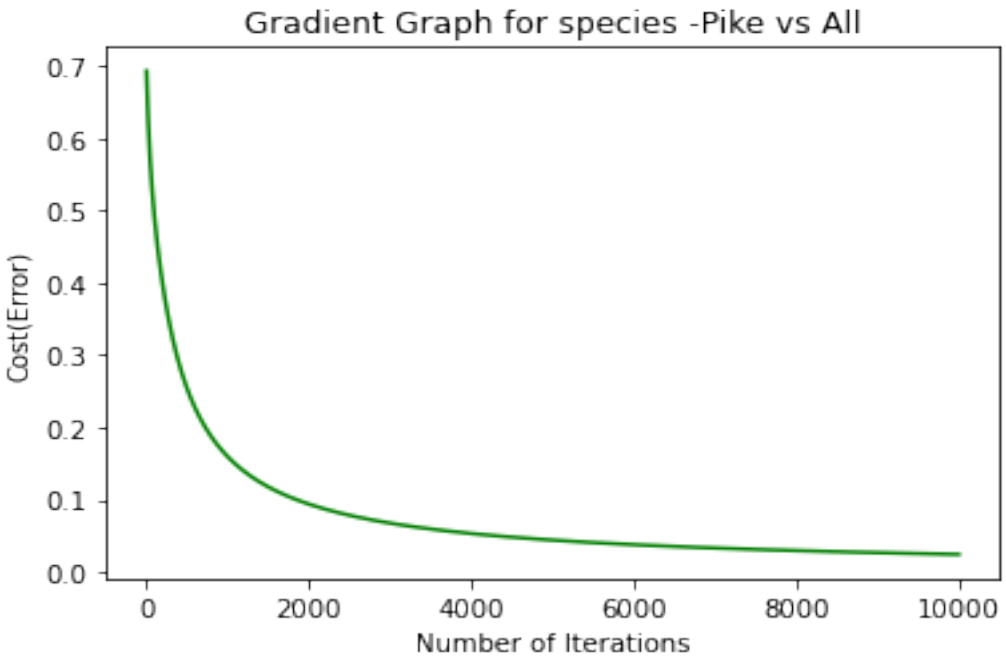

Fig3. Pike vs All

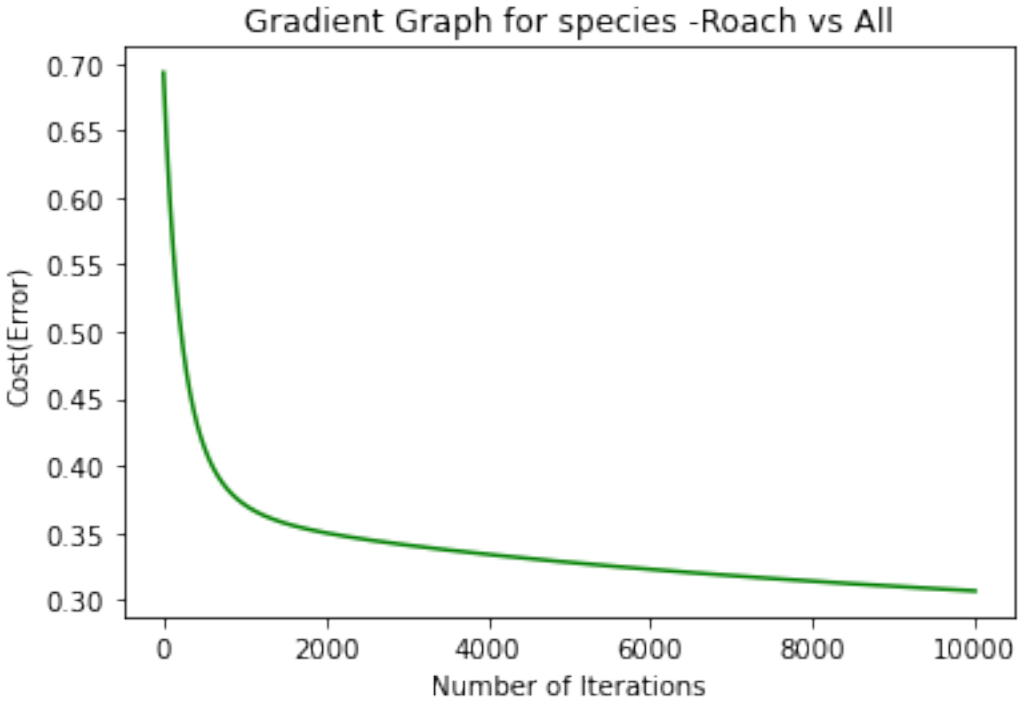

Fig4. Roach vs All

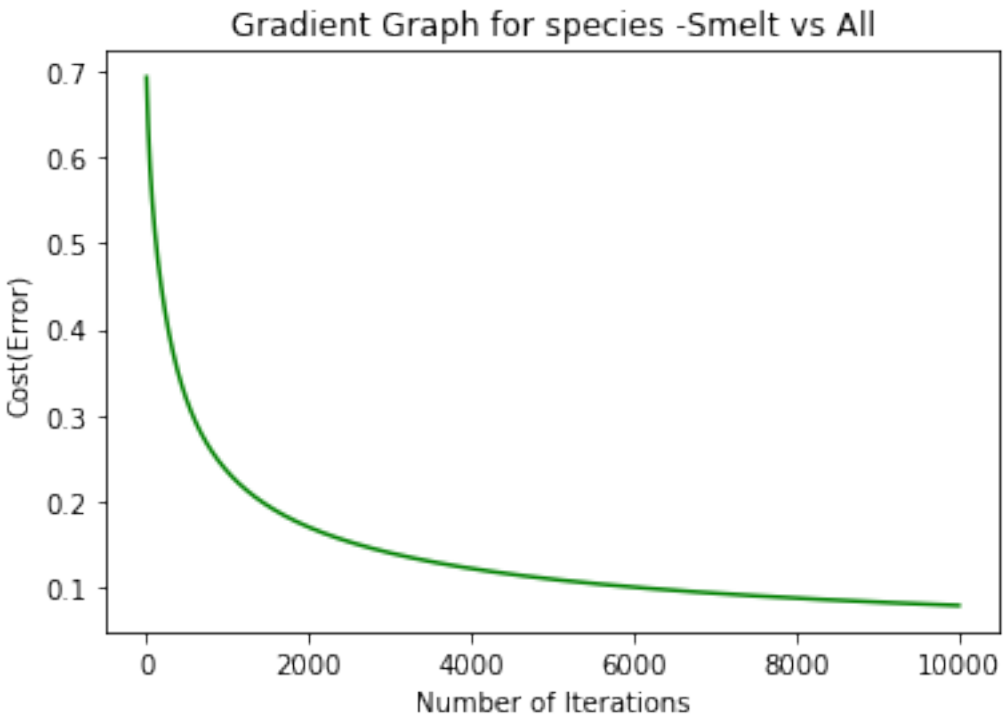

Fig5. Smelt vs All

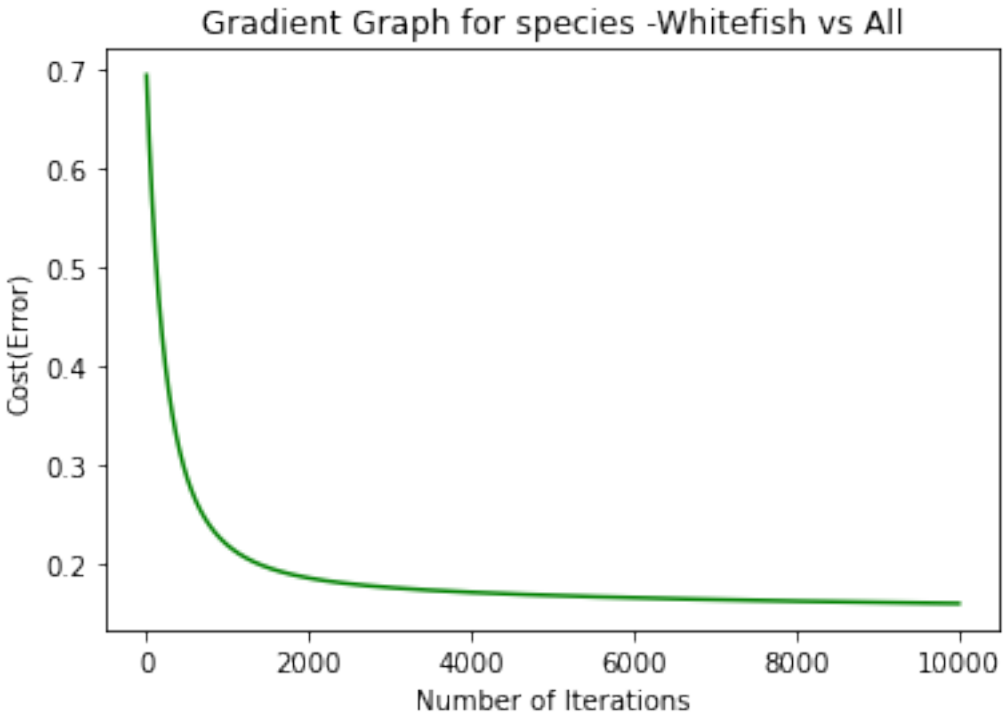

Fig6. Whitefish vs All