Deniz Aytemiz
denizaytemiz@vt.edu

ECE6524-HW3: Logistic Regression

Introduction

In this work logistic regression is examined mathematically, and practically on both simulated and real datasets. The effect of learning rate on log likelihood function along with how the algorithm is coded and evaluated is practised.

Method/ Approach

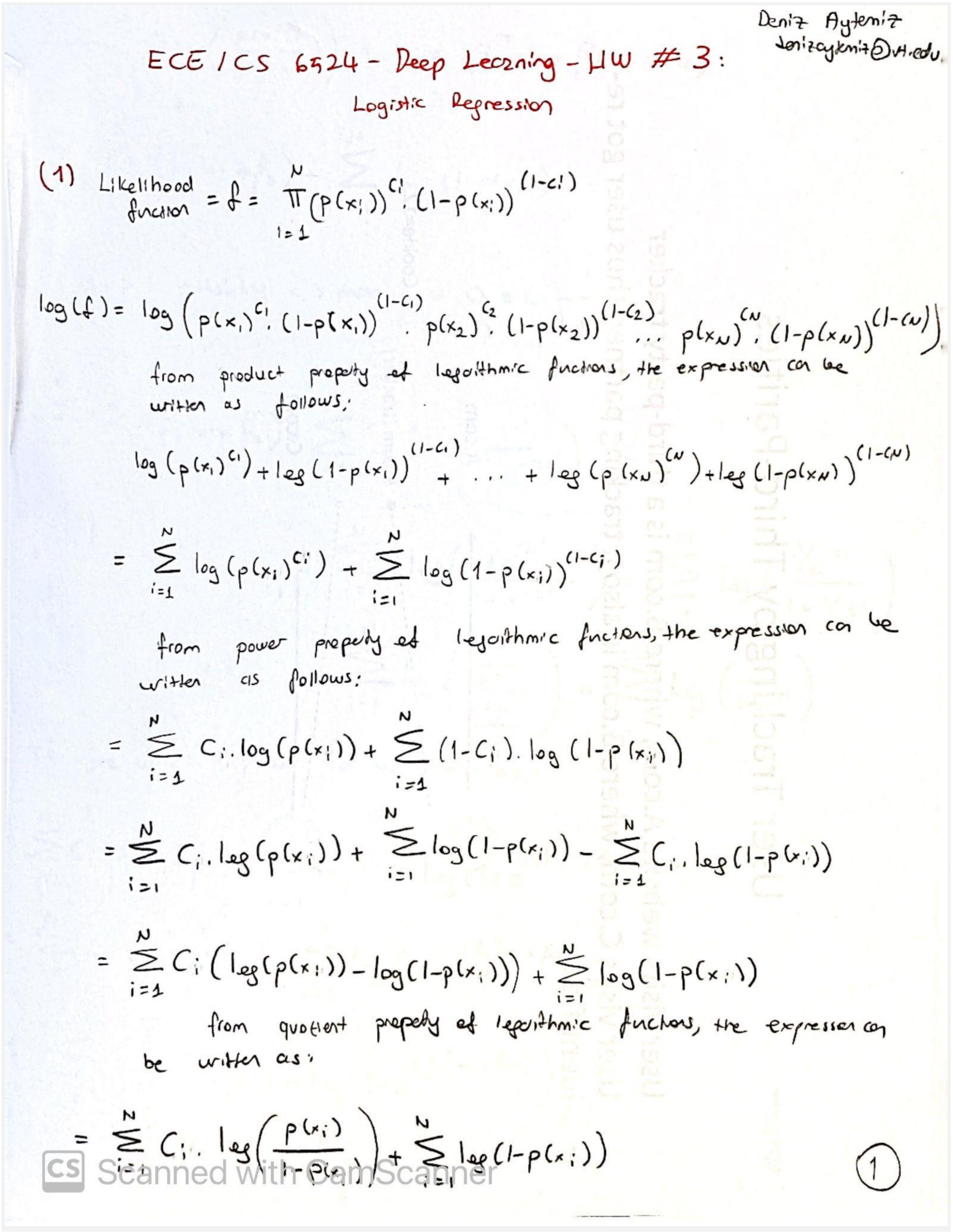
The logistic regression model is build for inputs with 2 features x1 and x2 and 2 classes y = 0 and y = 1. The optimal values for weights and biases are found and the model's accuracy is then tested with those parameters on test sets.

Experimental Results

The experimental results is given and discussed in each answer section.

Deniz Aytemiz
denizaytemiz@vt.edu

Question 1:



Deniz Aytemiz
denizaytemiz@vt.edu

ECE / CS 6524 - Deep Learning - HW # 3:

Logistic Regression

(1) Likelihood function $= f = \prod_{i=1}^{N} (p(x_i))^{c_i} \cdot (1-p(x_i))^{(1-c_i)}$

$$\log(f) = \log\left(p(x_1)^{c_1} \cdot (1-p(x_1))^{(1-c_1)} \cdot p(x_2)^{c_2} \cdot (1-p(x_2))^{(1-c_2)} \dots p(x_N)^{c_N} \cdot (1-p(x_N))^{(1-c_N)} \right)$$

from product property of logarithmic functions, the expression can be written as follows:

$$\log(p(x_1)^{c_1}) + \log(1-p(x_1))^{(1-c_1)} + \dots + \log(p(x_N)^{c_N}) + \log(1-p(x_N))^{(1-c_N)}$$

$$= \sum_{i=1}^{N} \log(p(x_i)^{c_i}) + \sum_{i=1}^{N} \log(1-p(x_i))^{(1-c_i)}$$

from power property of logarithmic functions, the expression can be written as follows:

$$= \sum_{i=1}^{N} c_i \cdot \log(p(x_i)) + \sum_{i=1}^{N} (1-c_i) \cdot \log(1-p(x_i))$$

$$= \sum_{i=1}^{N} c_i \cdot \log(p(x_i)) + \sum_{i=1}^{N} \log(1-p(x_i)) - \sum_{i=1}^{N} c_i \cdot \log(1-p(x_i))$$

$$= \sum_{i=1}^{N} c_i \left(\log(p(x_i)) - \log(1-p(x_i)) \right) + \sum_{i=1}^{N} \log(1-p(x_i))$$

from quotient property of logarithmic functions, the expression can be written as:

$$= \sum_{i=1}^{N} c_i \cdot \log\left(\frac{p(x_i)}{1-p(x_i)} \right) + \sum_{i=1}^{N} \log(1-p(x_i))$$

Deniz Aytemiz
denizaytemiz@vt.edu

we knew $\log\left(\frac{p(x_i)}{1-p(x_i)}\right) = y_i = \beta_1^T x_i + \beta_0$, hence,

$$= \sum_{i=1}^{N} C_i . (\beta_1^T x_i + \beta_0) + \sum_{i=1}^{N} \log(1-p(x_i))$$

$1-p$ = probability x_i's from class $0 = 1 - \frac{1}{1+e^{-y_i}}$

$$= \sum_{i=1}^{N} C_i . (\beta_1^T x_i + \beta_0) + \underbrace{\sum_{i=1}^{N} \log\left(1 - \frac{1}{1+e^{-y_i}}\right)}$$

$$= \sum_{i=1}^{N} \log\left(\frac{\cancel{1}+e^{-y_i}-\cancel{1}}{1+e^{-y_i}}\right) = \sum_{i=1}^{N} \log\left(\frac{1/e^{y_i}}{1+1/e^{y_i}}\right)$$

$$= \sum_{i=1}^{N} \log\left(\frac{\frac{1}{e^{y_i}}}{\frac{e^{y_i}+1}{e^{y_i}}}\right) = \sum_{i=1} \log\left(\frac{1}{1+e^{y_i}}\right)$$

Therefore whole expression can be written as,

$$= \sum_{i=1}^{N} C_i(\beta_1^T x_i + \beta_0) + \sum_{i=1}^{N} \overset{0}{\cancel{\log(1)}} - \log(1+e^{(\beta_1^T x_i + \beta_0)})$$

$$= \sum_{i=1}^{N} C_i(\beta_1^T x_i + \beta_0) - \sum_{i=1}^{N} \log(1+e^{(\beta_1^T x_i + \beta_0)})$$

$$= \sum_{i=1}^{N} \left(C_i(\beta_1^T x_i + \beta_0) - \log(1+e^{(\beta_1^T x_i + \beta_0)})\right)$$

Deniz Aytemiz
denizaytemiz@vt.edu

$$\text{log likelihood} = \sum_{i=1}^{N} C_i . (\beta_1^T x_i + \beta_0) - \log\left(1 + e^{(\beta_1^T x_i + \beta_0)}\right)$$

$$\beta_1^T x_i + \beta_0 = \underline{\beta}^T . \underline{x} = [\beta_0 \; \beta_1 \dots \; \beta_m] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{bmatrix} \xrightarrow{} = 1 \quad = \beta_0 + \beta_1 x_1 + \dots \beta_m . x_m = y$$

$$\text{log likelihood} = \sum_{i=1}^{N} C_i . \underline{\beta}^T . \underline{x} - \log\left(1 + e^{\underline{\beta}^T . \underline{x}}\right)$$

$$\frac{\partial(\text{log likelihood})}{\partial \beta} = \sum_{i=1}^{N} C_i . \underline{x} - \sum_{i=1}^{N} \underbrace{\frac{\partial}{\partial \beta} \left(\log\left(1 + e^{\underline{\beta}^T . \underline{x}}\right)\right)}$$

$$\downarrow$$

$$\frac{\partial}{\partial \beta}\left(\ln\left(1 + e^{\beta^T \underline{x}}\right)\right)$$

$$= \frac{\frac{\partial}{\partial \beta} . (1 + e^{\beta^T x})}{1 + e^{\beta^T x}}$$

$$= \frac{\frac{\partial}{\partial \beta} . (e^{\beta^T x})}{1 + e^{\beta^T x}}$$

$$= \frac{e^{\beta^T x} . \overbrace{\frac{\partial}{\partial \beta} (\beta^T x)}^{x}}{1 + e^{\beta^T x}}$$

$$\frac{\partial \text{ log likelihood}}{\partial \beta} = \sum_{i=1}^{N} C_i . \underline{x} - \sum_{i=1}^{N} \frac{x . e^{\beta^T \underline{x}}}{1 + e^{\beta^T \underline{x}}}$$

Deniz Aytemiz
denizaytemiz@vt.edu

Question 2:

From 2 normal distribution with means [0,0] and [0.5 ,4], 200 points from each are sampled. The means are selected as such to have some overlap but not that much. The covariance matrix is same for both to apply same sampling behaviours. The distributed points with mean [0,0] are labeled as Y = 0 and the ones with mean = [0.5, 4] are labeled as Y = 1. These sampled points are concatenated in a feature matrix with size 400x2 and the corresponding labels are put into array as class_label with size 400x1. Then both feature matrix and label array are shuffled before train and test split. 80% of dataset is reserved for training and rest is for testing. The sklearns logistic regression model trained with the training set outputs the following weights, w1 and w2 respectively,

```
[[-0.533276  4.182721]]
```

For the bias w0, it outputs `[-8.602465]`
It gives an accuracy of `0.9375`

Question 3:

The logistic regression function coded firstly initializes weights and biases, then it expands the feature arrays from [x1 x2] to [1 x1 x2] where x1 and x2 are features, in order to facilitate calculation of w0+w1*x1+w2*x2. Then this linear combination is calculated for each data point in training set and it is passed through a sigmoid function resulting in the p value that is equal to P(Y=1|X=x). Then the gradient is calculated as (p-y)*x1 for gradient with respect to w1 and so on. After the gradients for w0,w1 adn w2 are calculated the weights are updated by multiplying the gradients with the learning rate and nudging the weight values towards some local minima. This is done for all points in training dataset in each iteration where I set maximum value of iteration to 100 when testing my model. However in each iteration if the gradient vectors norm is below some tolerance value selected as 0.0001 for this function, the calculation and update of the weights are stopped and function returns the current weights stored. This function is trained with training set and the resulting bias is `[-4.074575302907977]`, and the w1 and w2 values are `[-0.17280742193702137, 2.279493819626461]`
We can say weights are close to the sklearns models weight parameters however bias is a bit different. I initiliazed the bias to value 1 though and rest to 0. This differences should be due to initialization of the parameters, learning rate and maximum iterations. Still the logistic_regression function gives good enough accuracy with these weights and bias.

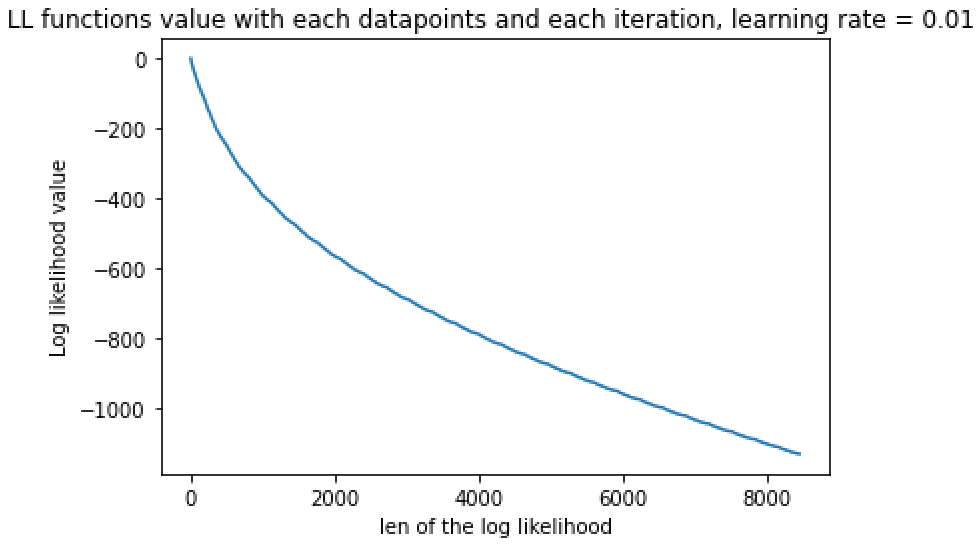
Question 4:

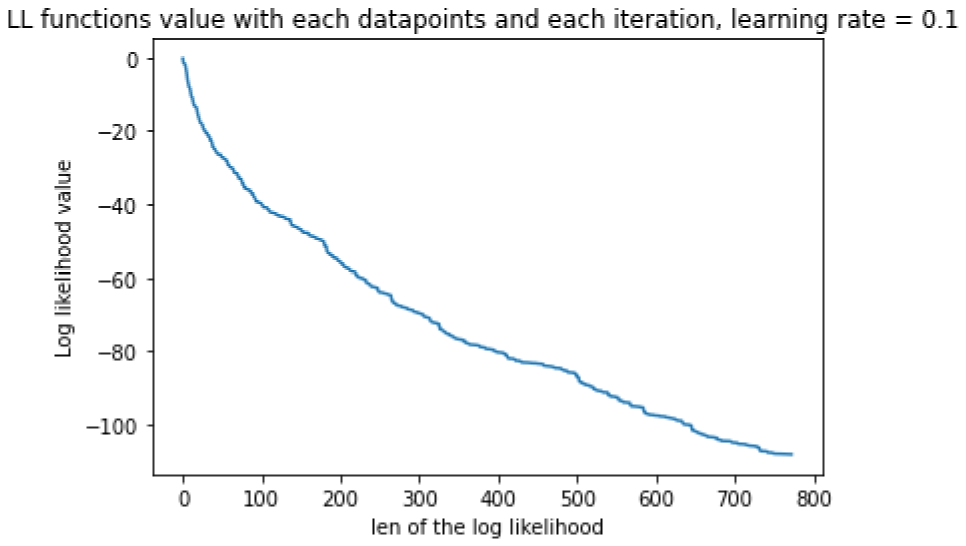
In order to understand how learning rate affects log likelihood function, I defined plot_LL function which is same with logistic_regression but I added log likelihood calculation as

a = y*log(p+0.0001)+(1-y)*log(1-p+0.0001),

Deniz Aytemiz
denizaytemiz@vt.edu

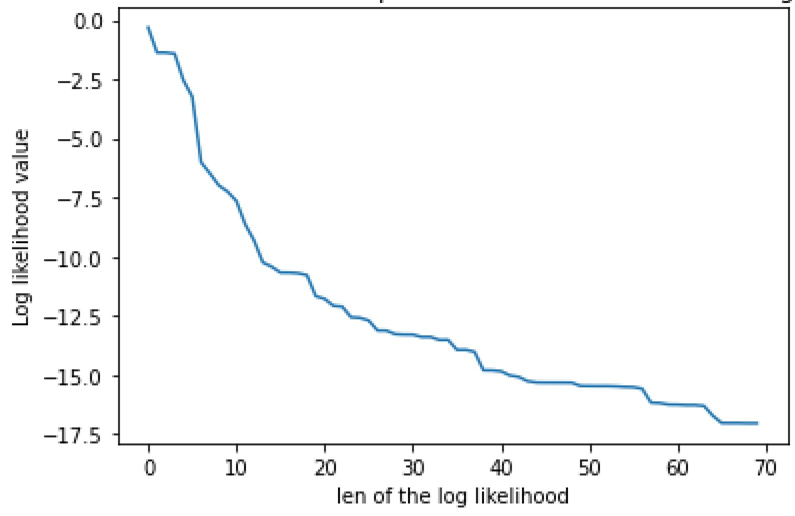
0.0001 is added into log in order to avoid inside of log being 0 and logarithmic function diverging. Variable a is calculated for each data point in training set and for each iteration and these a values area appended to a list. This list is plotted with different learning rates and teh results are as follows:



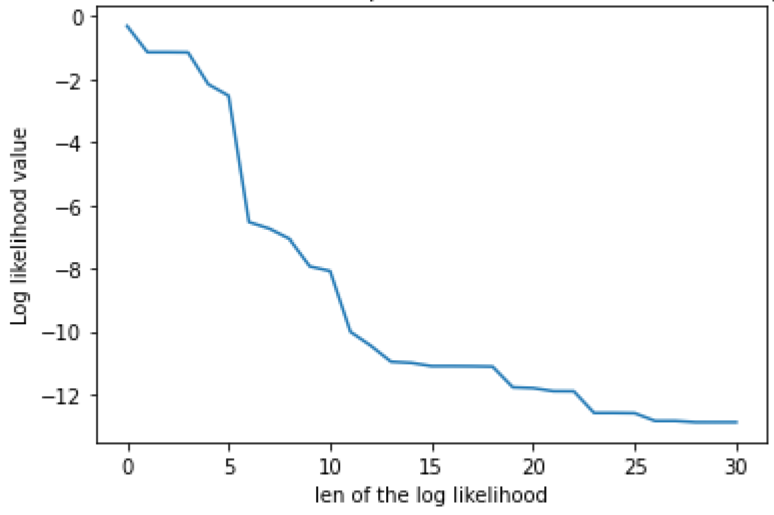


Deniz Aytemiz
denizaytemiz@vt.edu

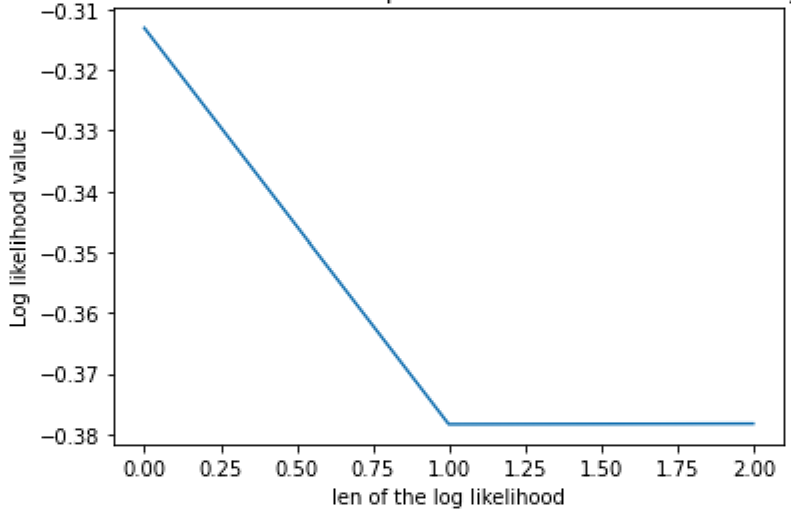
LL functions value with each datapoints and each iteration, learning rate = 0.5



LL functions value with each datapoints and each iteration, learning rate = 1



LL functions value with each datapoints and each iteration, learning rate = 5



Deniz Aytemiz
denizaytemiz@vt.edu

Comparing by the log likelihood value, we know the values closer to 0 is better then the ones converging in very small values. The log likelihood values are closer to 0 when learning rate increase but this is mainly because when learning rate inceraese it converges much faster and loop does not iterayte much. For instance for learning rate 5, the log likelihood array has only 2 elements meaning its gradient falls below tolerance value after calculation of second datapoint and the algorithm exits both loops so does not allow the log likelihood and weights to change. Considering much smaller learning rates, 0.01 for instance the log likelihood converges around -1200 which is too low. Since the step size is too small the function looks more smooth with respect to learning rates of 0.1 and 0.5 and also results in abigger array of log likelihood since the leap of values are very small. Learning rate 0.1 gives much better results and good amount of iteration too. The function looks more crooked since step size increased. Learning rate 0.5 seems the best since the value of log likelihood is closer to 0 and also doesn't leap to convergence point immediately but allows for good amount of iteration.

Question 5:

The accuracy with the logistic_regression model with test data results in 0.95, which is very good.

Question 6:

I have used IRIS data set with 5 features and 3 class. Uploaded the dataset into panda dataframe and experimented with classes [0,1],[1,2],[2,0] by dropping the rows with target value 2, 0 and 1 respectively. In order to use the logistic_regression function I have created I also dropped 3 features from the dataset to be able to update only 2 weights and a bias. The results look somewhat random, with accuracies 1.0, 0.55 and 0.05. I think the problems are firstly dropping the 3 features that were relevant. I should have modified the code instead to outcome 5 wights but I did not have enough time. Also the dataset with 2 classes are always 100 rows and I separated 80 of them for training and 20 of them for testing. And 20 test points are not enough to judge performance since all of them being correct for instance is high probability if the model has 90% accuracy. Then the overall accuracy will be 100. I believe the logistic_regression model works alright since it scored good with the previous dataset and a more suitable dataset for this model should be found but due to time constraints I am leaving it at it is.

Question 7:

In order to classify data with k classes, we can use one vs rest method. For k classes one can build k different logistic regression models to find probability that datapoint belongs to that class. Then the class with highest probability would be assigned to that datapoint. This calculation can be done with softmax regression.

Deniz Aytemiz
denizaytemiz@vt.edu

Conclusion: The logistic regression model is a binary classifier model that works with probabilistic approach and optimization of parameters. It is mostly used with classification for 2 classes though, it can be applied to multiclass datasets too by forming multiple models. The parameter tuning is very important for this model, especially learning rate determines how the loss function converges which directly affects the weights and biases we test our model on. Those can differ a lot by the selection of initial parameters and learning rate.