2023 -2024 Spring Term

CS412 – Machine Learning Course

Homework 3 – Report

Berke Ayyıldızlı – 31018

In this homework, I implemented a logistic regression model using gradient descent. By using a preprocessed version of the Titanic dataset, I predicted if a passenger survived or not, based on their age, sex and class. This report explains the answers I gave to each instruction on the question one by one, using the graphs I obtained and the numerical data to support it.

Q1. Load the dataset and preprocess the data:

In this question, I started by setting the seed to 42, splitting the set to 3 as train, validation and test, having percentages 60, 20 and 20. Than I scaled them using StandardScaler. Here is the result of train.head():

	Survived	Pclass	Sex	Age
570	1	-0.408652	0.737125	2.518697
787	0	0.803682	0.737125	-1.649218
74	1	0.803682	0.737125	0.203189
113	0	0.803682	-1.356623	-0.723015
635	1	-0.408652	-1.356623	-0.105546

Figure 1: Result of train.head()

Q2. Implement the logistic regression model:

In this question, it is asked from us to implement 2 functions and an algorithm to minimize the cost function. I started by initializing the parameters w and bias, then creating a sigmoid function, then creating a cost function which also finds outs the gradients for the algorithm, and another cost function just for the validation set and an algorithm function to update the cost function.

Q3. Set the step size to 0.1. Train your model using the training data. Calculate the loss on the validation data. Plot both the training and validation losses across 100 iterations.

On this question, as asked from us I called the algorithm function with 0.1 as the rate and 100 as the iteration amount. The validation loss and the resulting graph is as follows:

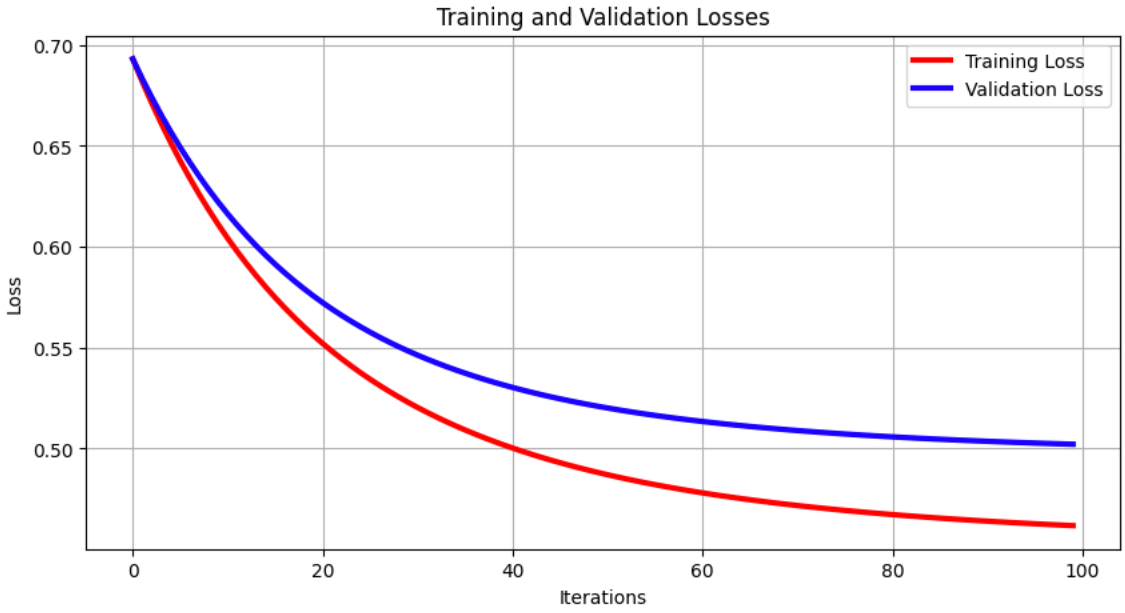

Figure 2 : Training and Validation Losses According to the Iterations

The validation loss after the end of the iterations is : 0.502144073183423

Q4. Now vary your step size and number of iterations, and calculate the validation loss in each case. Pick the one that gives you the best loss. Plot the loss curve across different iterations for the chosen values of these hyperparameters.

On this question, I picked 7 different steps, (0.01, 0.05, 0.1, 0.5, 0.7, 3, 4) with 4 different iterations. (100, 500, 1000, 5000) I computed each validation loss and plotted them. From that graph, I can see that the validation losses are too similar after the rate 0.1 until 1. So I picked 0.50 as the middle of it. Also for the iterations, although again, it does not really matter after 100 iterations with 0.50 step size, I picked 500 to be in the middle again. The respected validation loss curve graph is as follows:

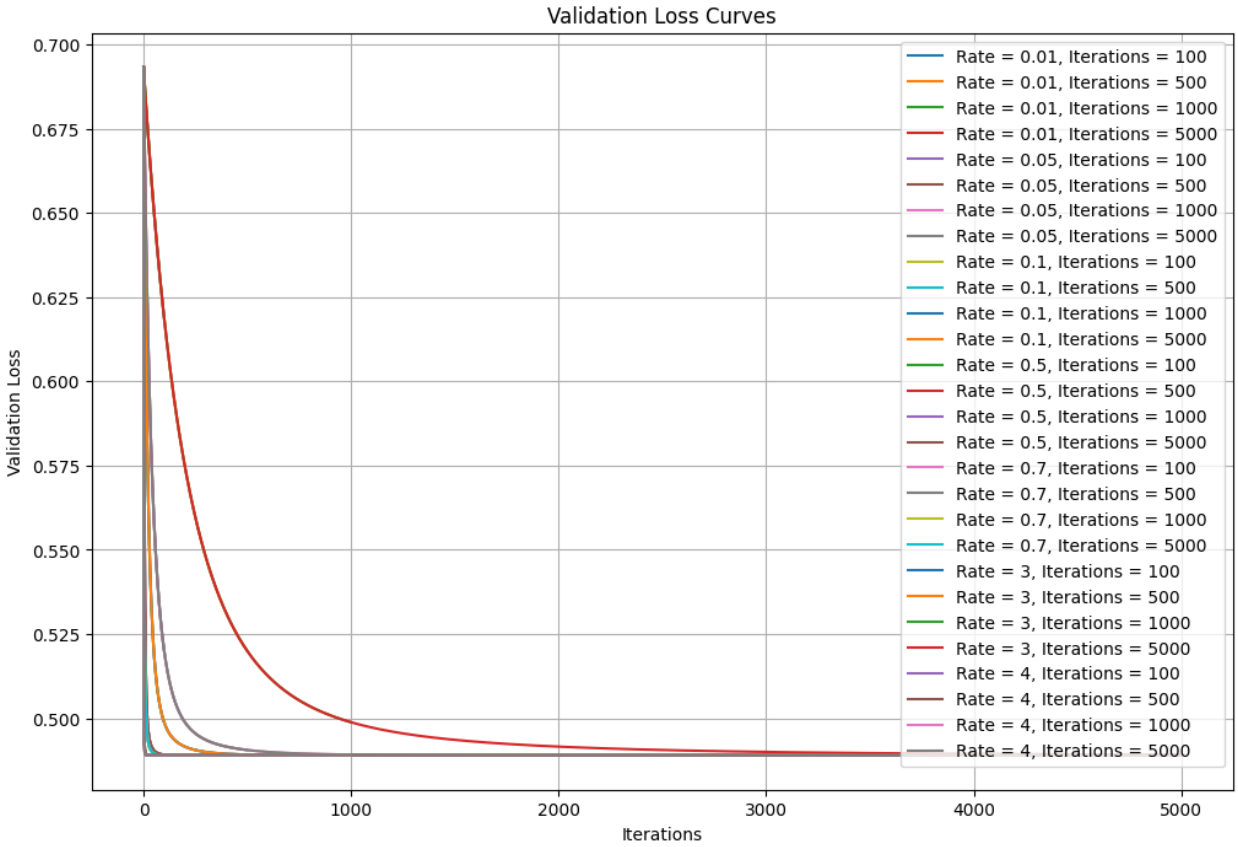

Figure 3: Validation Loss Curves WRT Number of iterations

In my code, I also printed out each validation loss for each of the rates and iterations, for the sake of simplicity, I am just showing the one with rate = 0.5 and iterations = 500.

```
Trained with rate = 0.5, iterations = 500, final validation loss = 0.4890339997274691
```

Q5. Combine the validation and training data and retrain the final model with the chosen hyperparameters.

In this question, first, like we have done on the previous homework and recitations, concatenated the train and validation datasets. Then, again using our training algorithm function, trained it using the recently created dataset with my selected hyperparameters, rate as 0.5 and iterations as 500. I plotted the training costs to a graph to better see the declare.

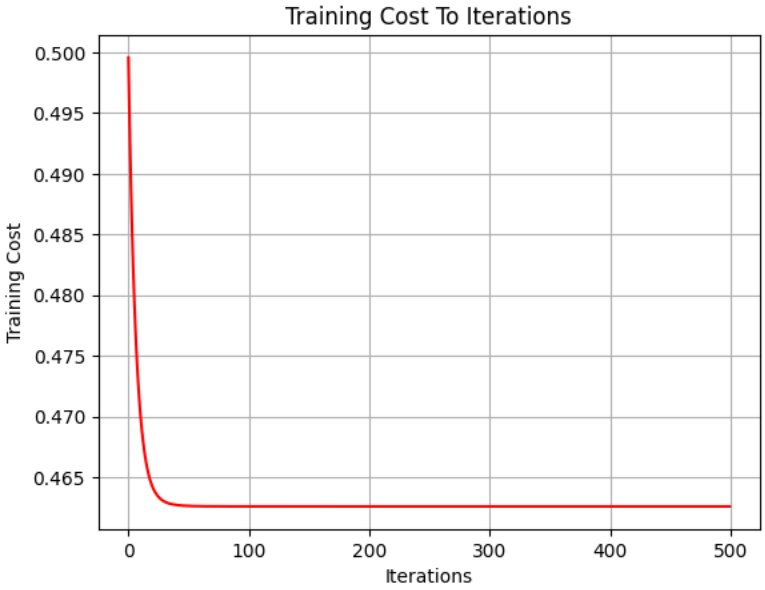

Figure 4: Training Cost Curve WRT Number of Iterations

The final training cost is : 0.46259167436911136

Q6. Evaluate the accuracy of your model on the testing data and report the results.

In this question, after inputting the sigmoid functions, I found the test accuracy as 82.68%. This result indicates that the model that I trained, has the ability to predict whether a passenger has survived the crash by the error rate of nearly 18 percent. This low amount of error rate mainly comes from the correct selection of the step size and the iteration count, as selecting less than 0.50 and 500 leads to a lack of training and values more than 0.50 and 500 leads to an overtrain of the model, further decreasing the percentage to correctly predict. Also dividing the main test data into validation also causes the algorithm to correct itself on the go, as we can directly see the validation test errors.