Ονοματεπώνυμο: Παντελεήμων Μαλέκας
A.M: 1115201600268

1. The partial derivatives of the ridge regression loss function are given by the following equation: $\frac{2}{m}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}^{(i)} - y^{(i)})\,x_j^{(i)} + \alpha w_j$

Explanation: First we compute the partial derivatives with respect to w_i for the Mean Squared Error. As we saw in the lecture's slides, this is given by:

$$\frac{\partial}{\partial w_j} MSE(\mathbf{w}) = \frac{2}{m}\sum_{i=1}^{m}\left(\mathbf{w}\cdot\mathbf{x}^{(i)} - y^{(i)}\right)x_j^{(i)}$$

Next we compute the partial derivatives for the second term of the function, meaning:

$$\alpha\frac{1}{2}\sum_{i=1}^{n} w_i^2.$$

To find the partial derivatives, we will use the following proof: Let n = 2

This gives us: $\frac{\alpha}{2}(w_1^2 + w_2^2)$ and we have: $\frac{\partial}{\partial w1} = \frac{\alpha}{2} 2w_1 = \alpha w_1$ and we also have: $\frac{\partial}{\partial w2} = \frac{\alpha}{2} 2w_2 = \alpha w_2$ and so on for every other n. So, the partial derivatives with respect to w_i for the second term is given by: αw_j

2. For the second question I used the Batch Gradient Descent code that was given in Piazza and I came up with my own ideas for the Stochastic/Mini-batch Gradient Descent implementation.

After defining the gradient descent functions, I loaded the HousingData.csv (using drive.mount) and I created the X and y arrays. The MinMaxScaler module was used to perform scaling on the X array.

Then I executed the Gradient Descent function with 1000, 5000 and 10000 steps, showing the change in the error function with each run. As we can notice, each time the steps increase the loss in the error function becomes more accurate.

For the Stochastic/Mini-batch gradient descent executions I defined some necessary functions for the creation of the batches. Setting batch size = 1 is the equivalent for Stochastic Gradient Descent. I executed the SGD algorithm with steps 3, 5 and 10, showing again the values of the error function for each change. It should be noted that step sizes are way smaller than the ones in Gradient Descent. This is done because the gradient is computed for every single batch. So

the iterations ought to be less for Stochastic/Mini-batch. As we can see, the R2 score in SGD is about the same as in Gradient Descent. Yet, the error function behaves far differently.

So, I also showed some examples for the Mini-batch executions. Again step sizes are the same as the ones in SGD, though the learning rate was tweaked in a few cases. Examples are shown for batch sizes 8, 32 and 128. As we can see, batch size = 8 gives us error values similar to the ones in SGD executions. With batch size = 32 the error starts to resemble the one in Gradient Descent. Finally, with batch size = 128 we get somewhat close to the error of Gradient Descent. We can see that every time the batch size increases we get more accurate values for the error function.

Additional information may be seen at the comments of the program.

3. For the third question I implemented the sentiment classifier using LogisticRegression from sklearn. The data is loaded using drive.mount and after preprocessing the 'text' column, the TF-IDF vectorizer is initialized to fit the data. Finally, the LogisticRegression model is initialized and after making the predictions, the classification report is given, showing scores from precision, recall and F-measure metrics.

Specifically, after loading in the data, every character is converted to lowercase and then some unwanted characters are removed with the sub() function provided by the re module. I chose to remove URLs, escape characters (like \n, \x) and every non-alphabetic character.

After preprocessing, the dataframe is split to 80% train/20% test sets. Next, stemming and lemmatization is performed on the text columns. The TF-IDF vectorizer is initialized using unigrams and bigrams and the stemmed/lemmatized text is fit/transformed (I also tried using stop words but it gave worse accuracy for some reason). Finally we save the target columns as Y.

Finally, the classification is performed and after that, we get the classification report showing precision, recall and F1 metrics. As a bonus, cross validation is also performed on the data, showing the given scores. (I should also note that I increased the max iterations of the algorithm because the default value gave some convergence errors)

Links that were used for the first question:

https://eclass.uoa.gr/modules/document/file.php/DI517/%CE%94%CE%B9%CE%B1%CF%86%CE%AC%CE%BD%CE%B5%CE%B9%CE%B5%CF%82/2_regression.pdf

https://math.stackexchange.com/questions/2605546/partial-derivative-of-a-summation