Pawel Linek
Jesse Hazard
Assignment2 Report
11/5/2019

Our implementation of the multiple linear regression algorithm utilizes normalizing the data with the built in MinMaxScaler function from sklearn.preprocessing. This brings our data to the range of 0-1 where we then look through the data and see if each column will be relevant to training. If we find that a given column is 98% the same, we drop that column in the training and test data. This will leave our model to train off of columns which have relevant and useful data for training. We also penalize high weight values using a PolynomialFeature function also imported from sklearn. We then select 1/5th of the data and apply the closed form for linear regression. We also tried to regularize the data by simulating an appropriately sized identity matrix and applying the formula ((Xt*X)+gamma*I)^-1*Y.

We use a learning rate of 0.01 and look at 20,000 rows of data in each step, running it until we look at most, if not all of the data. For example, if we have 100 rows of data, we will look at 20 rows, get a weight vector, look at the next 20 rows update the weight vector, and so on. We do this until we look at all of the data. We also have the potential to run this for numerous epochs, but because we are looking at the entire data set, after the first epoch our accuracy doesn't really improve.

We had training set accuracy of 96.4%, which is a little high. After submitting it on kaggle we found that our algorithm performed rather poorly. We received a root mean squared error of 850ish, which is slightly worse than random submission but considering previous error scores of 2000, 1400. This shows improvement and with a little more fiddling around with the code I believe we could have improved the accuracy to be better than the random submission.csv.

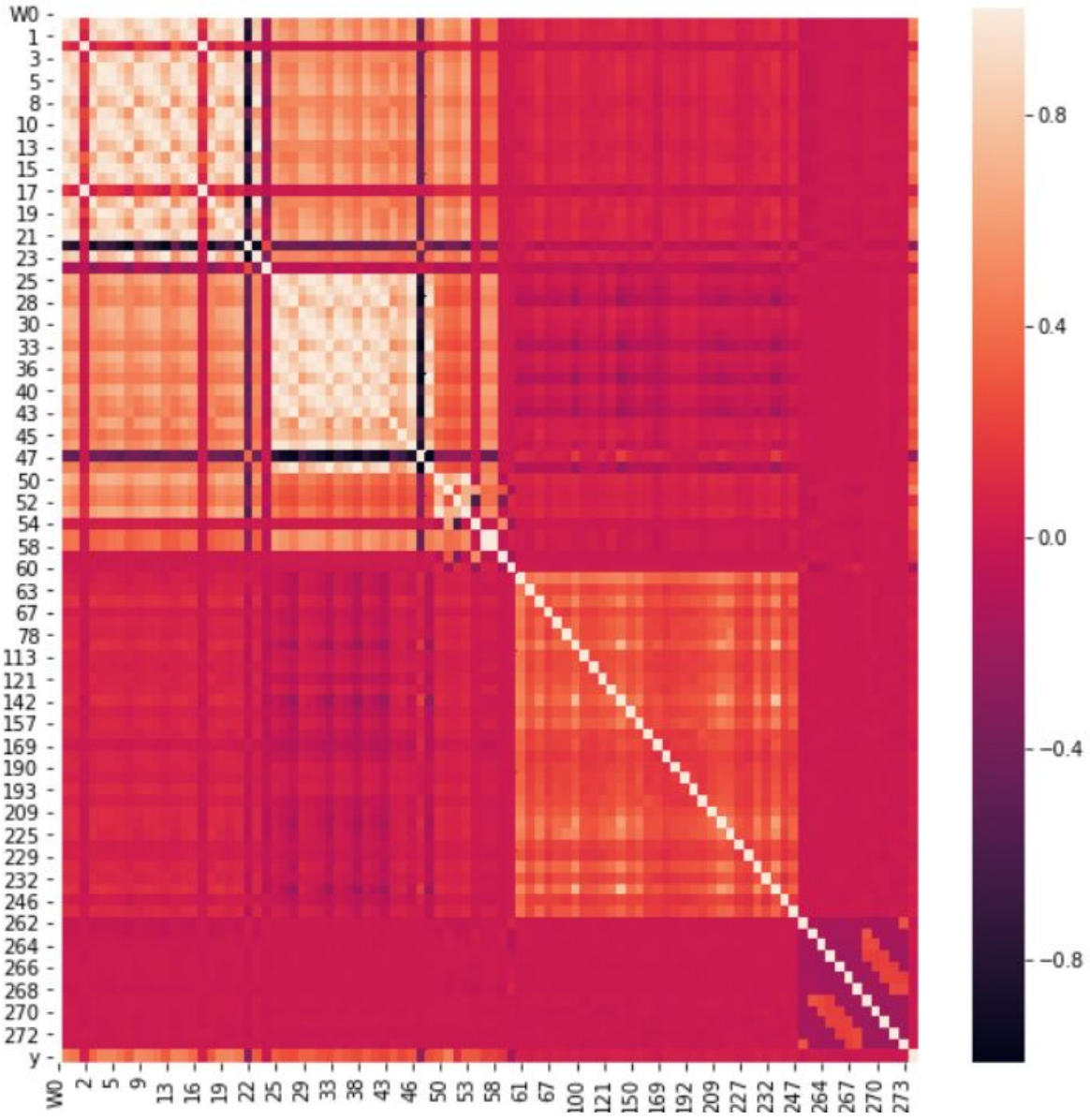
Below is a list of the mean absolute error score after each epoch:

[0.0383, 0.0380, 0.03825, 0.03828, 0.03839] … and so on,

As can be seen, after the first epoch we don't have much accuracy improvement.

As a result of a high training set accuracy and a low test set accuracy we conclude that our algorithm is overfitting.

Below is a heatmap of all the variables after being dropped:


W0
1
3
5
8
10
13
15
17
19
21
23
25
28
30
33
36
40
43
45
47
50
52
54
58
60
63
67
78
113
121
142
157
169
190
193
209
225
229
232
246
262
264
266
268
270
272
y
W0
2
5
9
13
16
19
22
25
29
33
38
43
46
50
53
58
61
67
100
121
150
169
192
209
227
232
247
264
267
270
273
0.8
0.4
0.0
−0.4
−0.8