

Stone Barrett

CSE 590: Intro to Machine Learning

Exam 1

Problem 1a:

For this problem, I decided to create a dataset centered around predicting the breed of dogs based on height and weight. Height and weight serve as the two dimensions, chihuahua and rottweiler serve as the two classes, and there are 30 samples.

Classes:	Chihuahua	Rottweiler	Features:	Height (in)	Weight (lbs)
			Dog 1	17	85
			Dog 2	6	3.3
			Dog 3	17.3	90
			Dog 4	6.1	3.4
			Dog 5	24.2	112
			Dog 6	6.2	3.5
			Dog 7	24.3	113
			Dog 8	6.3	3.6
			Dog 9	25	114
			Dog 10	6.4	3.7
			Dog 11	26	115
			Dog 12	6.5	3.8
			Dog 13	26.1	116
			Dog 14	14.5	24
			Dog 15	25.3	117
			Dog 16	15	23
			Dog 17	25.1	118
			Dog 18	6.6	3.9
			Dog 19	24.9	117
			Dog 20	6.5	3.8
			Dog 21	24.8	116
			Dog 22	6.4	3.7
			Dog 23	24.5	115
			Dog 24	6.3	3.6
			Dog 25	26.7	114
			Dog 26	6.2	3.5
			Dog 27	26.9	113
			Dog 28	6.1	3.4
			Dog 29	27.1	112
			Dog 30	6	3.3

Figure E1.11

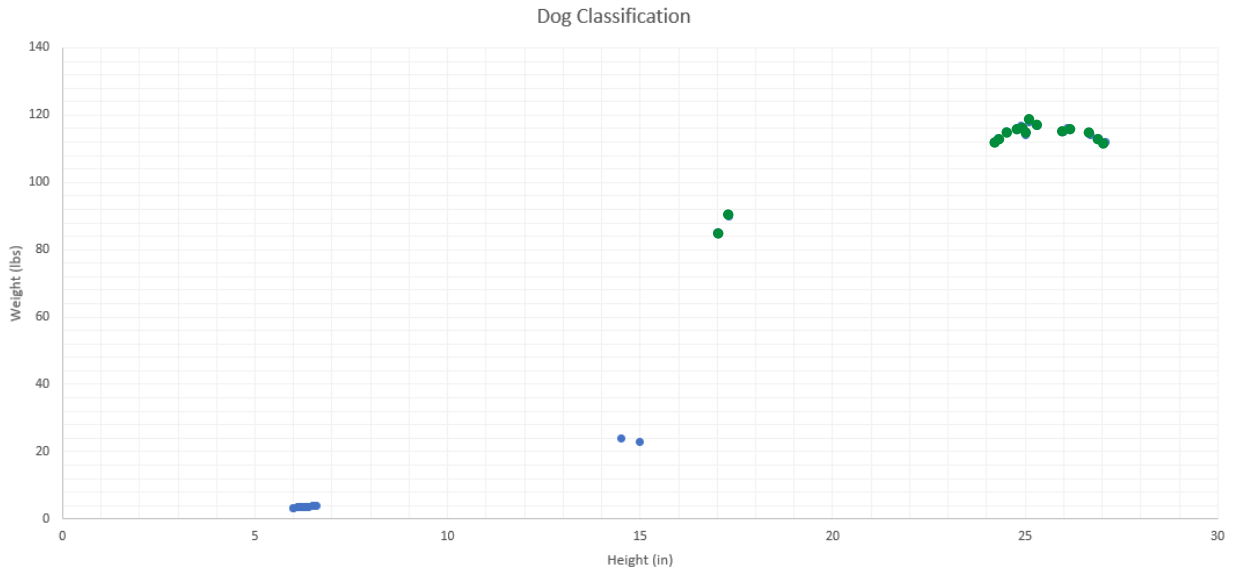


Figure E1.12

Problem 1b:

Figure E1.12 displays a scatterplot of the dataset. This scatterplot clearly shows the capability of this data to be correctly classified by a Logistic Regression classifier.

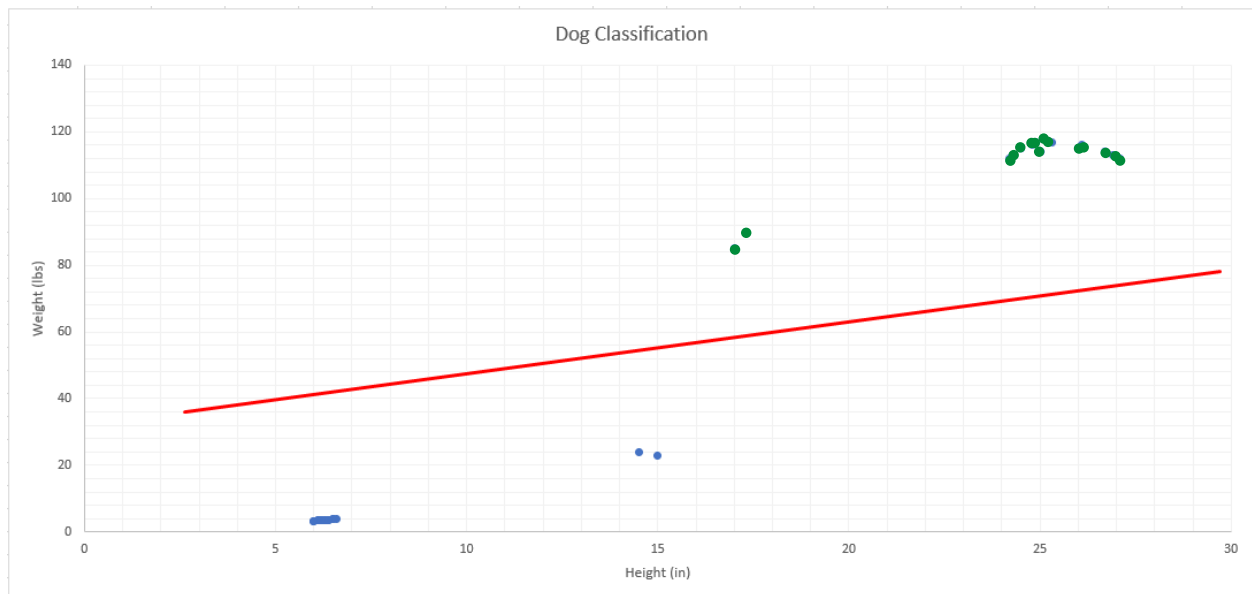


Figure E1.13

Figure E1.13 shows what the scatterplot would look like if the line determined by a Logistic Regression classifier were present on it. All samples of chihuahua and rottweiler are correctly grouped together, respectively.

Problem 1c:

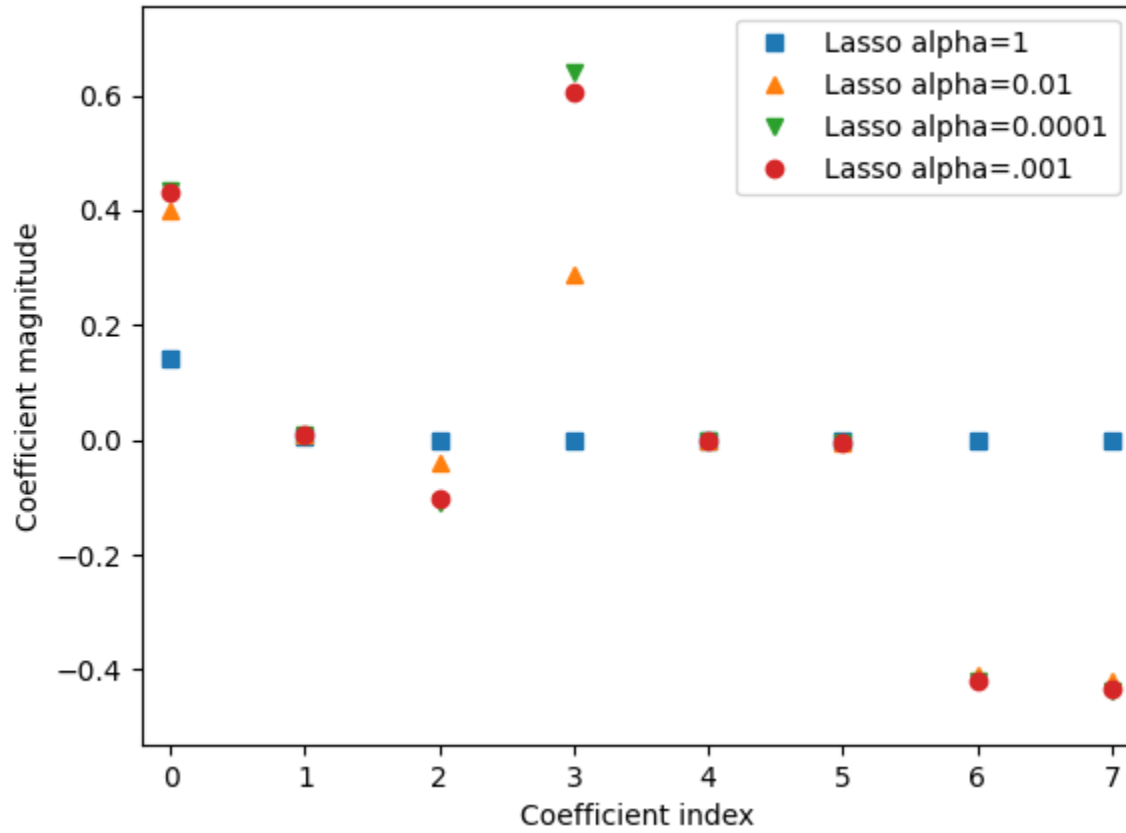
Figure E1.11 lists all the samples and their attributes, as well as highlights four of them. The two yellow highlighted samples are the two chihuahuas that would be misclassified by a KNN classifier. The two blue highlighted samples are the two rottweilers that would be misclassified by a KNN classifier. These samples would be misclassified by KNN because they are anomalies – the rottweilers are shorter and lighter than their average and the chihuahuas are taller and heavier than their average. With a k-value of 3, any one of these data points would be closer to each other than any data point in their average grouping. This would cause the classifier to assume they are of similar class, skewing the prediction.

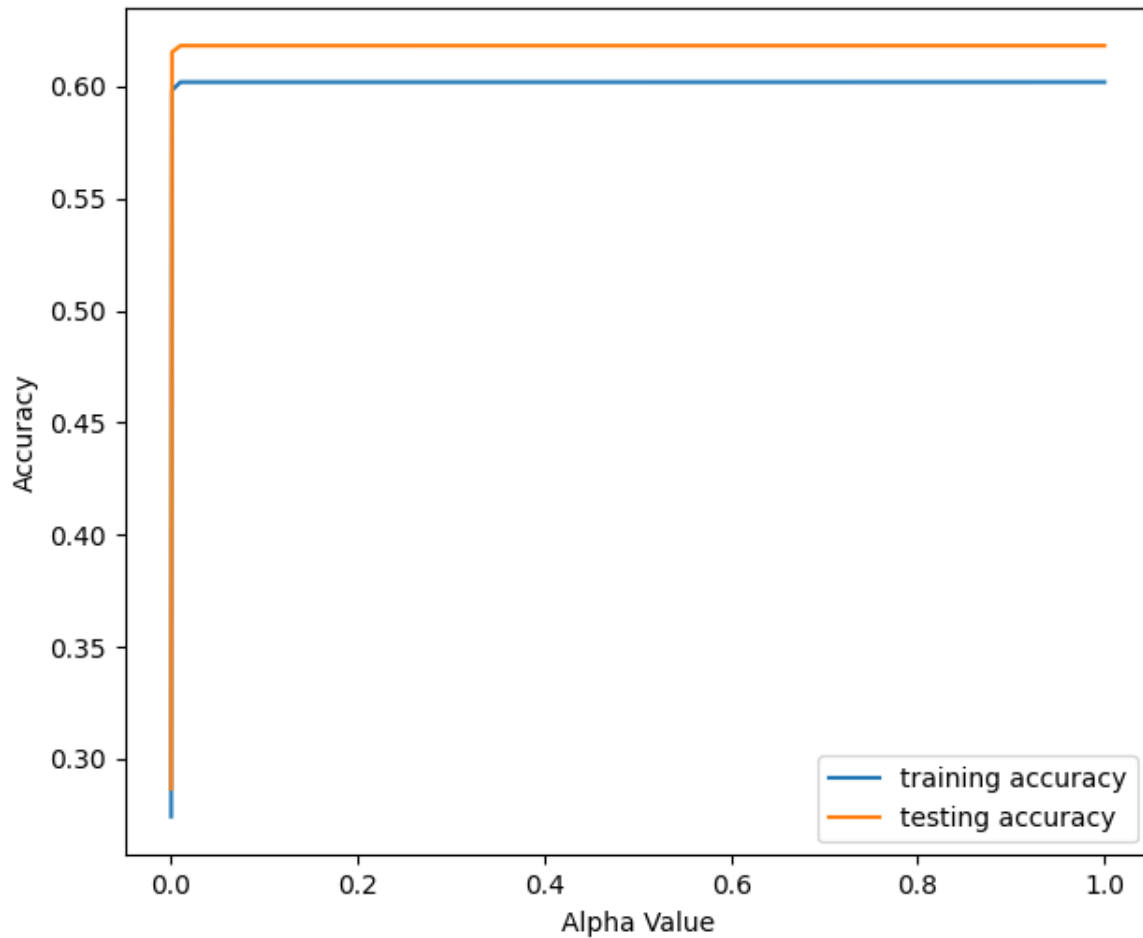
Problem 2a, 2b, 2c:

I do not believe it is possible to create such a dataset, specifically because of the caveat that the Decision Tree classifier pruning parameter `max_depth` must be set to 2. As we have observed in this course's homework and on this exam in Problem 3b, Decision Tree classification accuracy increases with the number of levels you allow it to go. Only having two levels would result in very low accuracy in most cases, so classification could not be perfect.

Problem 3a:

To analyze the effectiveness of the LASSO Regression model on this dataset, I decided to implement a coefficient magnitude chart and vary the alpha-parameter to find its optimal state.





```
Training set score: 0.27
Testing set score 0.29
Number of features used: 3

Training set score: 0.60
Testing set score 0.62
Number of features used: 8

Training set score: 0.60
Testing set score 0.62
Number of features used: 8
```

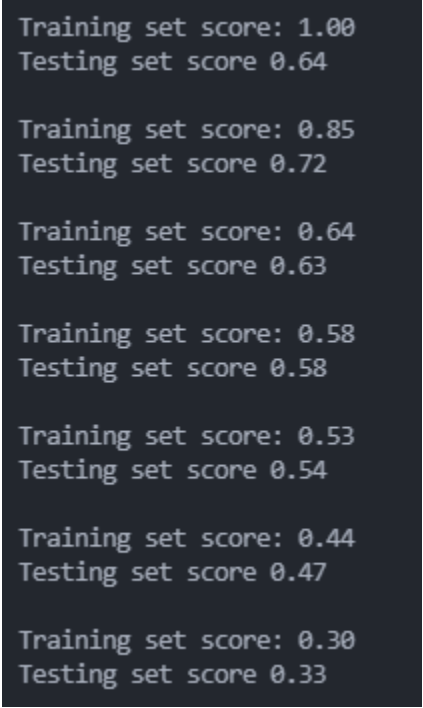
Figure E1.33

As can be seen in Figure E1.32, the training and testing accuracy plateau immediately and stay at the same place for all alpha-values. Since the two are close together, it does mean that the model isn't overfit. However, the two only hover around the 60% accuracy mark, meaning it

isn't a particularly good fit, either. The LASSO Regression model is a slight underfit for this dataset. The alpha-value seems to be equally optimal at any point between 0.1 and 1.

Problem 3b:

To test the effectiveness of the Decision Tree Regression model, I varied the max_depth pruning parameter and created a graph of where the training and testing accuracies grew to.



```
Training set score: 1.00  
Testing set score 0.64  
  
Training set score: 0.85  
Testing set score 0.72  
  
Training set score: 0.64  
Testing set score 0.63  
  
Training set score: 0.58  
Testing set score 0.58  
  
Training set score: 0.53  
Testing set score 0.54  
  
Training set score: 0.44  
Testing set score 0.47  
  
Training set score: 0.30  
Testing set score 0.33
```

max_depth	Training set score	Testing set score
1	1.00	0.64
2	0.85	0.72
3	0.64	0.63
4	0.58	0.58
5	0.53	0.54
6	0.44	0.47
7	0.30	0.33

Figure E1.34

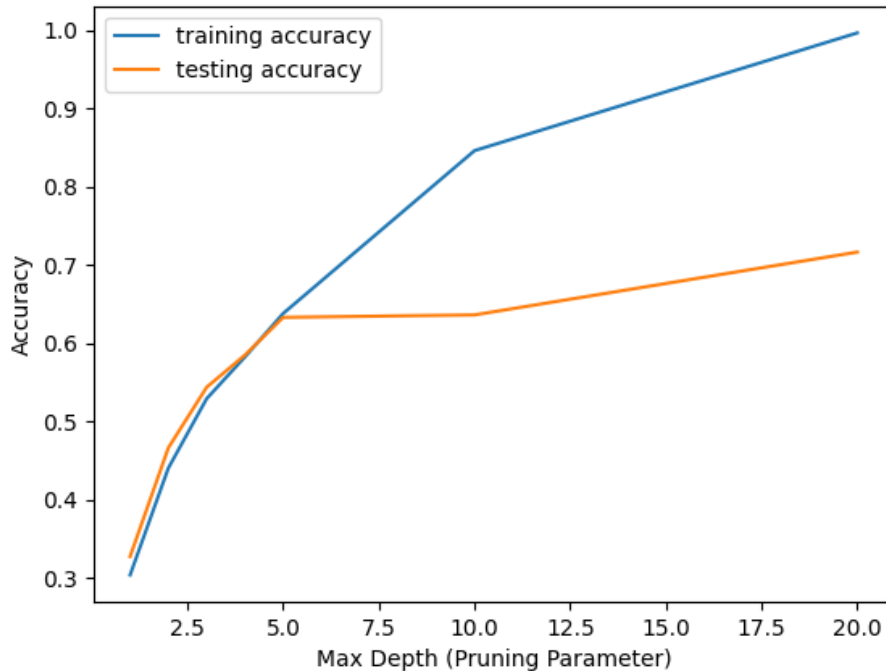


Figure E1.35

It is of note that the lines printed in Figure E1.34 represent a decreasing pruning parameter. Figure E1.35 is plotted in correct order, however. As we can see in Figure E1.35, the model starts out as severely underfit with both accuracies hovering just above 30%. As the max_depth value grows, so do the accuracy scores. Eventually, the training accuracy grows much faster than the testing accuracy, meaning that the model becomes overfit. So, to find the optimal max_depth value, we will back up to when the lines were near each other at the highest point. This would place the optimal value of the pruning parameter at 5.

Problem 3c:

Given the outcomes observed and the results displayed throughout the previous two sections, it is safe to say that a Decision Tree Regression model is a better fit for predicting the California Housing dataset than the LASSO Regression model. At each model's optimal parameters, the testing accuracy of the Decision Tree is slightly higher than the testing accuracy of the LASSO model. It would also seem that the Decision Tree has higher potential for testing accuracy scores, as the LASSO model seemingly will never grow beyond where it started for this dataset.

Problem 4a:

To test the Random Forest classifier, I decided to vary the number of estimators parameter while running 4-fold cross-validation and find which value resulted in the best fit for the data.

I started with $n = 1$ and got the following results:

```
Cross-validation scores: [0.71513353 0.72700297 0.76854599 0.72321429]
Average cross-validation score: 0.733474194573972
Accuracy on training set: 0.924
Accuracy on testing set: 0.780
```

Next, I tried $n = 10$ and got:

```
Cross-validation scores: [0.9495549 0.9347181 0.92878338 0.95238095]
Average cross-validation score: 0.9413593330507277
Accuracy on training set: 1.000
Accuracy on testing set: 0.940
```

$n = 100$ results:

```
Cross-validation scores: [0.97922849 0.97626113 0.95548961 0.98511905]
Average cross-validation score: 0.9740245690264235
Accuracy on training set: 1.000
Accuracy on testing set: 0.976
```

$n = 1000$:

```
Cross-validation scores: [0.98219585 0.97329377 0.95548961 0.98214286]
Average cross-validation score: 0.973280521407376
Accuracy on training set: 1.000
Accuracy on testing set: 0.978
```

And finally, $n = 10000$:

```
Cross-validation scores: [0.97922849 0.97329377 0.95845697 0.97916667]
Average cross-validation score: 0.9725364737883283
Accuracy on training set: 1.000
Accuracy on testing set: 0.980
```

Generally speaking, the number of estimators being higher is a good thing in the Random Forest classification. However, we can see that from $n = 1000$ to $n = 10000$ the average cross-

validation score actually decreased slightly. The testing set score did improve, though it seems at this point an increase by a factor of 10 on the number of estimators only means a .02 gain on the testing score. All the way back to $n = 1$, we can see that this classifier is technically overfitting to this dataset and as the number of estimators increases, the magnitude of overfitting decreases. By $n = 100$, the testing set accuracy is barely lower than the training set's 100% score. Given the consideration of the minute difference between $n = 1000$ and $n = 10000$ as well as the fact that higher estimator counts drive up computational resource cost significantly, I would say the optimal n -value in this case is 1000.

Problem 4b:

I have already answered some of this question in Problem 4a's section above, though it is important to mention that the generalization capability is quite high for this model. Even though the training set accuracy is a perfect 100% (which usually means a severe overfit and that the model has "memorized" the training data instead of learning from it) in this case, the testing set accuracy gets up to a 98%, which means the model generalized very well.

Problem 4c:

I had actually already implemented a feature importance visualization before seeing the part about a grayscale heat map. I will include both in this section!

Figure E1.46

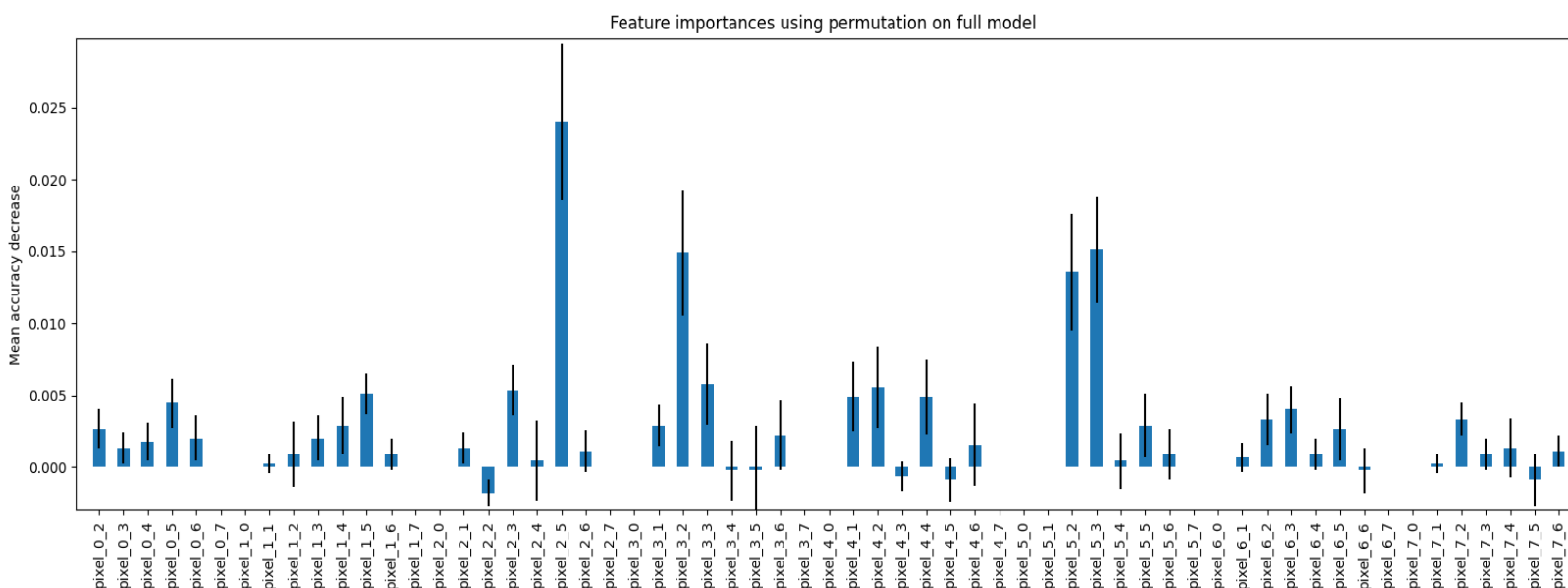
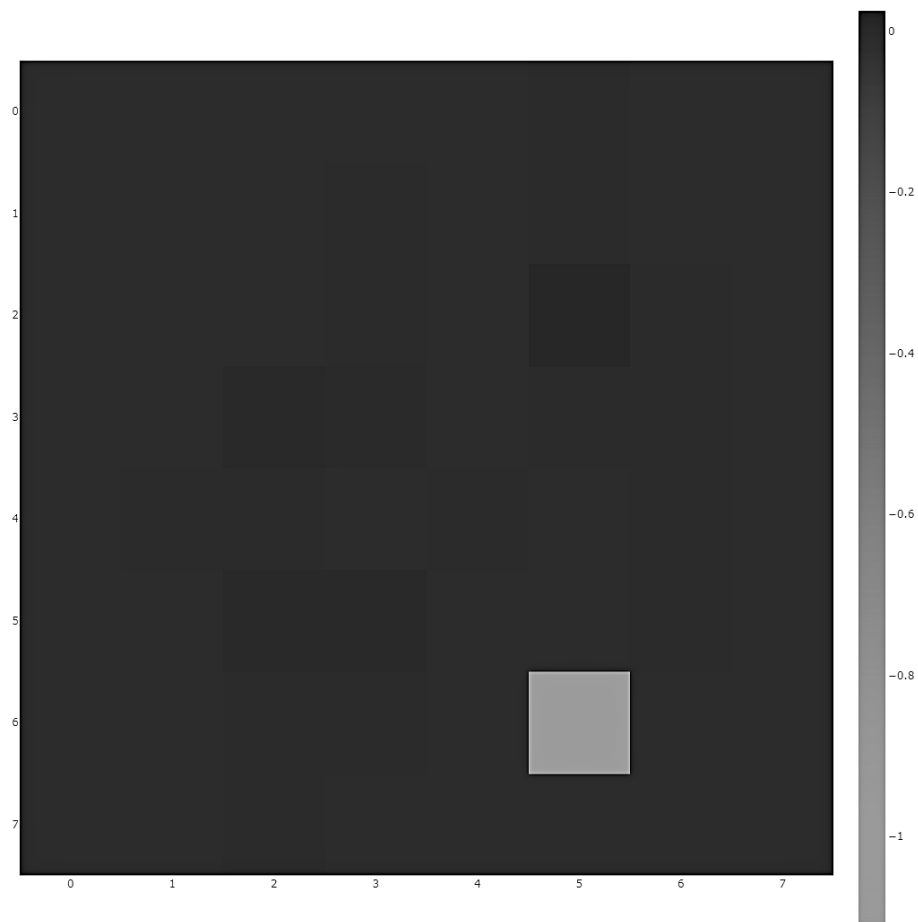


Figure E1.47



From Figure E1.46 and E1.47 , we can see that most features are very similarly quantified in terms of relevance to predicting testing set data. However, there are a few features that stand out significantly. These features are Pixel 25, Pixel 32, Pixel 52 and Pixel 53. One can conclude these four pixels out of 64 are the ones that set decisive differences between most handwritten digits.

Problem 4d:

There are no consistently misclassified samples.