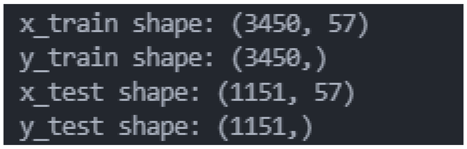
Stone Barrett

CSE 590: Intro to Machine Learning

Homework 2

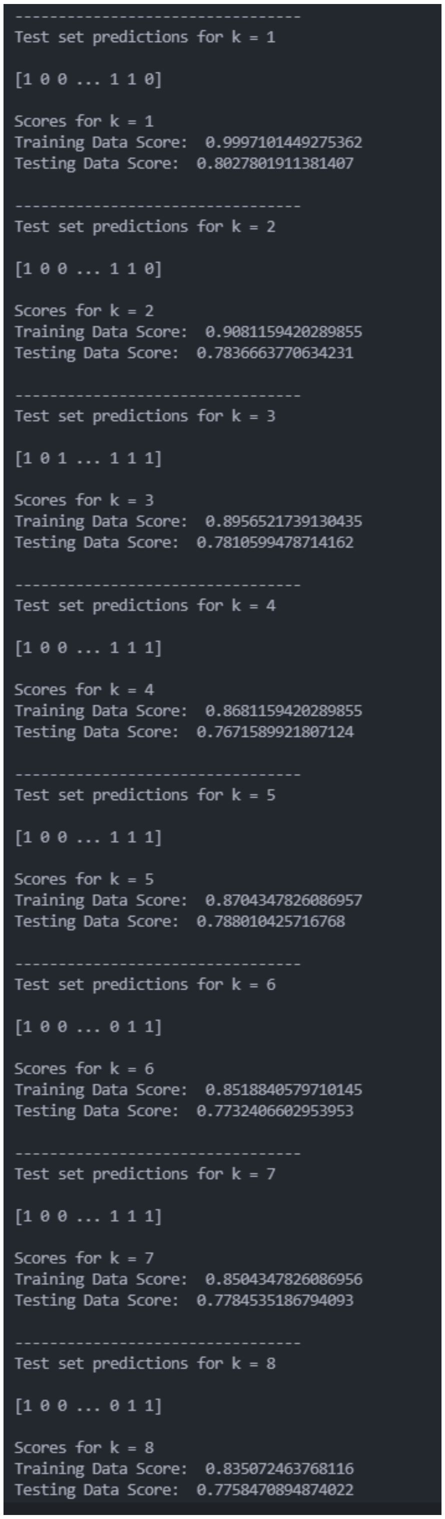
The goal of this assignment is to apply k-fold cross-validation with k = 5 to find optimal parameters for different classifier algorithms, then choose which algorithm is best suited to use to work with the provided data based on accuracy and computational time required. The algorithms in question are a K-nearest neighbors (KNN) binary classifier, a logistic regression classifier, and a linear support vector machines (SVM) classifier. Cases of underfitting and overfitting will also be identified while using these algorithms. The data provided is information regarding spam emails and the objective is to classify an email as spam or not spam by considering many other attributes about the email.

Figures



```
x_train shape: (3450, 57)
y_train shape: (3450,)
x_test shape: (1151, 57)
y_test shape: (1151,)
```

Figure 2.1 – Verifying data is read correctly by viewing the shape of it in a Pandas dataframe



```
--------------------------------
Test set predictions for k = 1

[1 0 0 ... 1 1 0]

Scores for k = 1
Training Data Score:  0.9997101449275362
Testing Data Score:  0.8027801911381407

--------------------------------
Test set predictions for k = 2

[1 0 0 ... 1 1 0]

Scores for k = 2
Training Data Score:  0.9081159420289855
Testing Data Score:  0.7836663770634231

--------------------------------
Test set predictions for k = 3

[1 0 1 ... 1 1 1]

Scores for k = 3
Training Data Score:  0.8956521739130435
Testing Data Score:  0.7810599478714162

--------------------------------
Test set predictions for k = 4

[1 0 0 ... 1 1 1]

Scores for k = 4
Training Data Score:  0.8681159420289855
Testing Data Score:  0.7671589921807124

--------------------------------
Test set predictions for k = 5

[1 0 0 ... 1 1 1]

Scores for k = 5
Training Data Score:  0.8704347826086957
Testing Data Score:  0.788010425716768

--------------------------------
Test set predictions for k = 6

[1 0 0 ... 0 1 1]

Scores for k = 6
Training Data Score:  0.8518840579710145
Testing Data Score:  0.7732406602953953

--------------------------------
Test set predictions for k = 7

[1 0 0 ... 1 1 1]

Scores for k = 7
Training Data Score:  0.8504347826086956
Testing Data Score:  0.7784535186794093

--------------------------------
Test set predictions for k = 8

[1 0 0 ... 0 1 1]

Scores for k = 8
Training Data Score:  0.835072463768116
Testing Data Score:  0.7758470894874022
```

Figure 2.2 – Iterating through K-values to view test set predictions, training scores, and testing scores

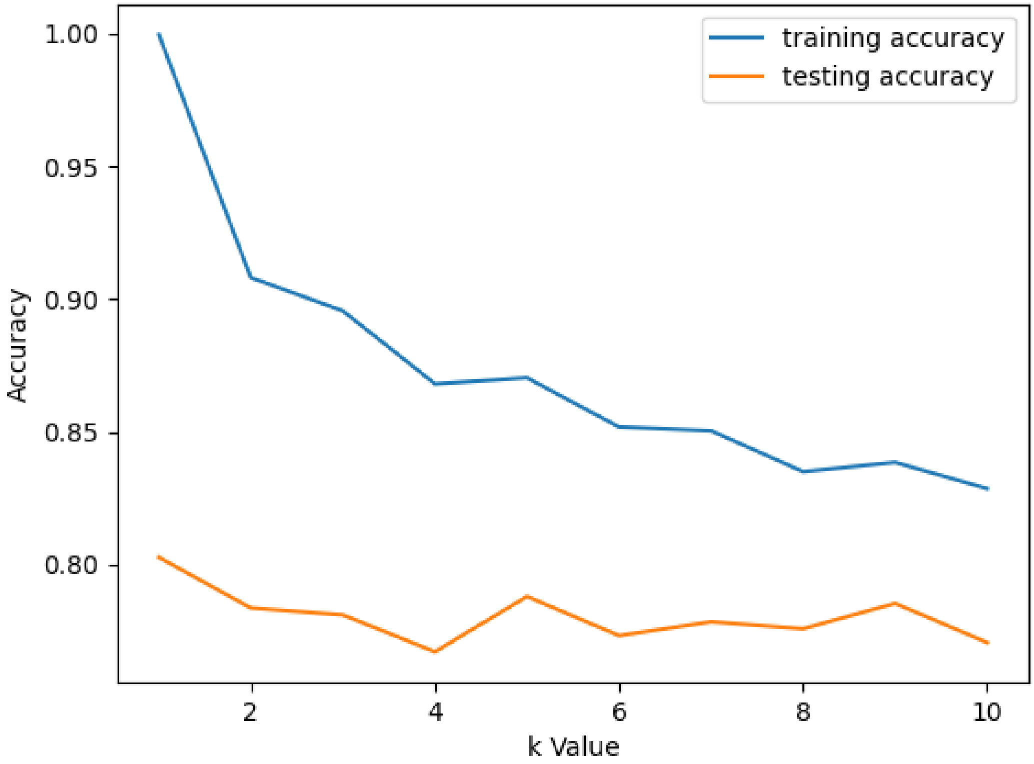
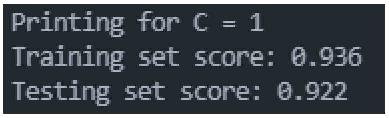
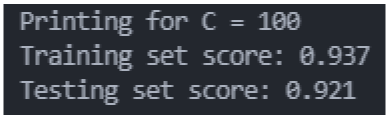


Figure 2.3 – Training and testing scores plotted alongside each other to choose optimal K



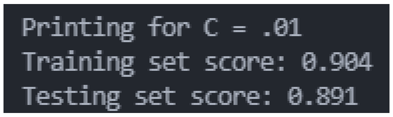
```
Printing for C = 1
Training set score: 0.936
Testing set score: 0.922
```

Figure 2.4 – Logistic Regression classifier scores for C = 1



```
Printing for C = 100
Training set score: 0.937
Testing set score: 0.921
```

Figure 2.5 – Logistic Regression classifier scores for C = 100



```
Printing for C = .01
Training set score: 0.904
Testing set score: 0.891
```

Figure 2.6 – Logistic Regression classifier scores for C = .01

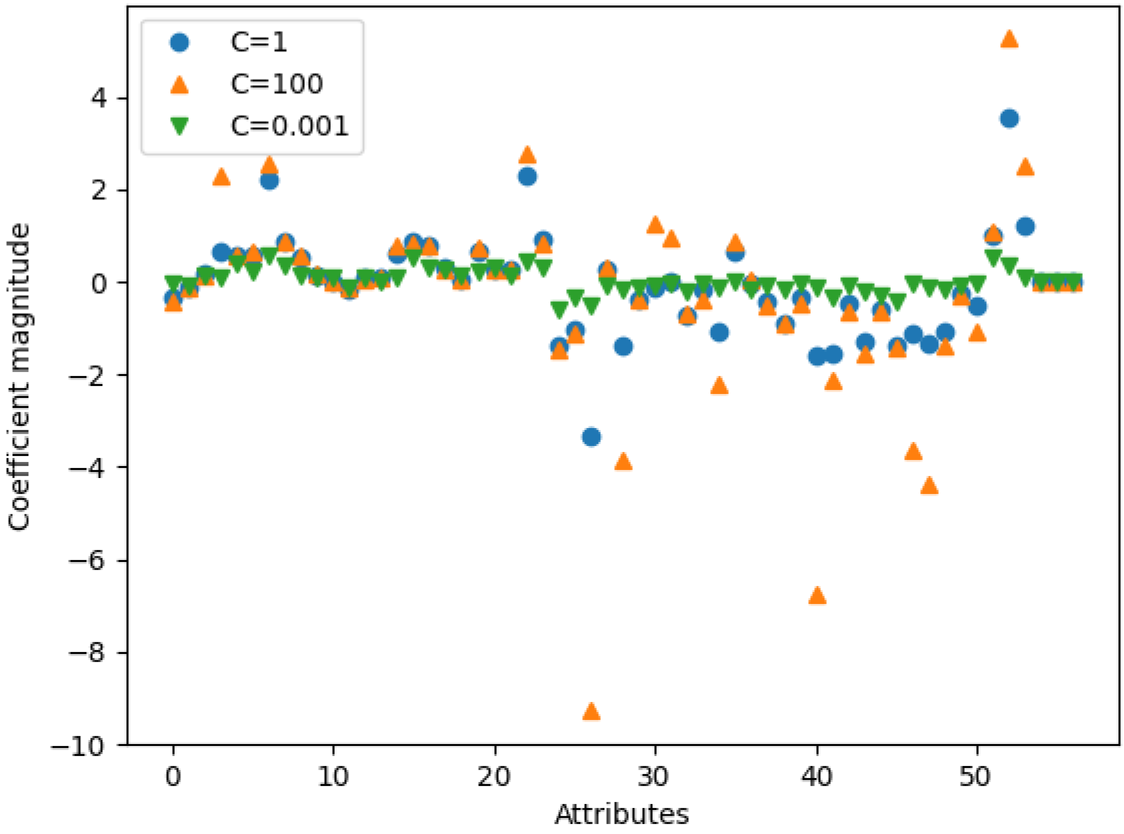
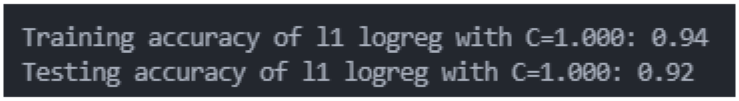
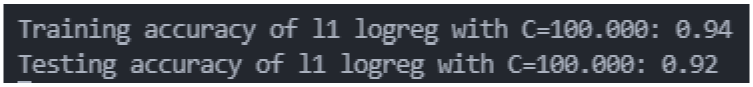


Figure 2.7 – Logistic Regression classifier feature magnitudes plotted



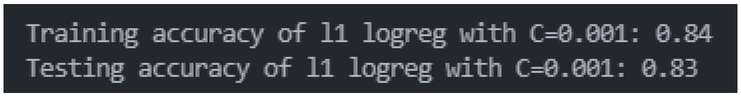
```
Training accuracy of l1 logreg with C=1.000: 0.94
Testing accuracy of l1 logreg with C=1.000: 0.92
```

Figure 2.8 – Logistic Regression classifier scores using L1 regularization for C = 1



```
Training accuracy of l1 logreg with C=100.000: 0.94
Testing accuracy of l1 logreg with C=100.000: 0.92
```

Figure 2.9 – Logistic Regression classifier scores using L1 regularization for C = 100



```
Training accuracy of l1 logreg with C=0.001: 0.84
Testing accuracy of l1 logreg with C=0.001: 0.83
```

Figure 2.10 – Logistic Regression classifier scores using L1 regularization for C = .01

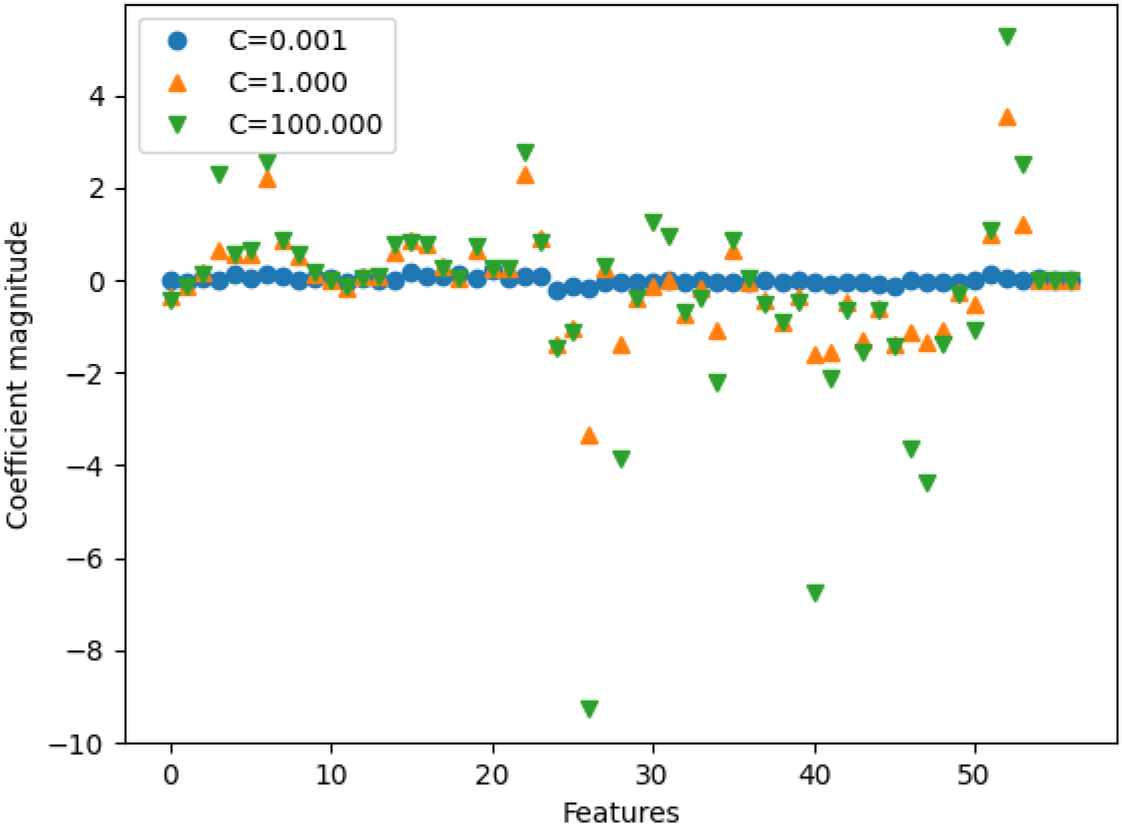
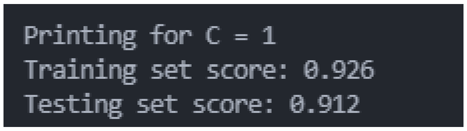
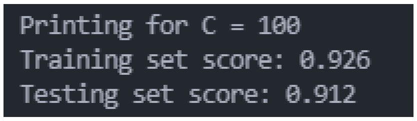


Figure 2.11 – Logistic Regression classifier using L1 regularization feature magnitudes plotted



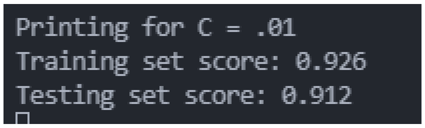
```
Printing for C = 1
Training set score: 0.926
Testing set score: 0.912
```

Figure 2.12 – Linear Support Vector Machines classifier scores for C = 1



```
Printing for C = 100
Training set score: 0.926
Testing set score: 0.912
```

Figure 2.13 – Linear Support Vector Machines classifier scores for C = 100



```
Printing for C = .01
Training set score: 0.926
Testing set score: 0.912
```

Figure 2.14 – Linear Support Vector Machines classifier scores for C = .01

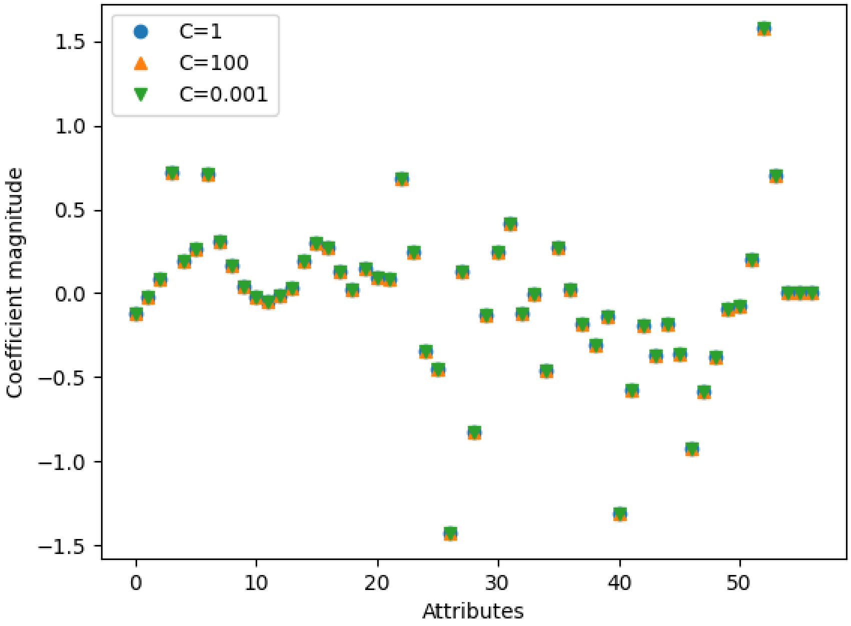


Figure 2.15 – Linear Support Vector Machines classifier feature magnitudes plotted

Problem 1a: KNN binary classifier

To begin, we revisit the K-Nearest Neighbors algorithm, which considers the nearest K datapoints to a given position to predict where one of similar attributes would land. In this scenario, the job is to determine a line of demarcation between spam and not spam and place testing data accordingly. Figure 2.2 shows how values of K are iterated through and scored based on training data fit and testing data fit. My program tested values of K from 1 to 10, as these are generally considered reasonable values to test for. Anything greater typically results in an underfitting scenario. In fact, in Figure 2.3 we can actually see underfitting taking place as early as K = 3. Based on the scores analyzed, the optimal value of K would happen to be 1 for this dataset. At K = 1, the training data accuracy is perfect, and the testing data accuracy is at its global maximum. This algorithm completed its run of this dataset very quickly.

Problem 1b: Logistic Regression classifier

Next, we will look at the Logistic Regression classifier. In this case, we vary the parameter C to find the optimal value much like K for the previous algorithm. Figures 2.4, 2.5, and 2.6 show the training scores and testing scores for C = 1, C = 100, and C = .01 respectively. Based on these scores, we can see that the optimal C could lie at either C = 1 or C = 100, given that the scores only differ by a thousandth of a percentage point in either direction. Figure 2.7 shows the feature magnitudes plotted for this model, meaning that we can see how important a specific attribute of the emails is to determining whether it is spam or not. We can see from this graph that for C = 100, there are a couple features that have significant weight on the classification. Figures 2.8, 2.9, and 2.10 show the training and testing scores for C = 1, C = 100, and C = .01 respectively when L1 regularization is applied. From these scores, we can determine that C = 1 and C = 100 are identical in accuracy. Figure 2.11 shows the feature weights for this run, which looks very similar to the aforementioned graph in Figure 2.7. This algorithm completed its run of this dataset very quickly.

Problem 1c: Linear Support Vector Machines classifier

Finally, we will use the Linear Support Vector Machines classifying algorithm on this dataset. Similarly to the Logistic Regression classifier, we will need to vary parameter C to find its optimal value here. Figures 2.12, 2.13, and 2.14 show the training and testing scores for C = 1, C = 100, and C = .01 respectively. These scores seem to indicate that all three of these tested C values are of equal viability. I imagine this is caused by something in the program running incorrectly, as the Linear Regression classifier suggested otherwise. This is not a guaranteed reason to assume error, however in this case I suspect as much. Figure 2.15 shows the feature magnitudes plotted, where we can see that apparently they were all identical for the three different C values as well. As a side note, there is a typo in Figure 2.15 suggesting that C = .001 when it is actually C = .01. This algorithm took a very long time to run its course on this dataset.

Conclusions

After testing these three models on the provided dataset, it would seem there is a clear answer on which I would select as a "best algorithm" for this dataset. KNN ran very quickly, but it did not yield very accurate results. (Medium overfitting descending into strong underfitting.) SVC provided very accurate results (no underfitting or overfitting) but took a very, very long time to run. Logistic Regression provided very accurate results (no underfitting or overfitting) AND only required a few seconds to run. Based on this experiments, it would seem the best algorithm for binary classification for the provided dataset would be Logistic Regression.