CSE514 – Fall 2022 Programming Assignment 2

Sterling Lech – 505660

Introduction:

For this programming project, I was provided a table of data with an assortment of all letters of the English alphabet with various integer features that accompany each letter. These integer features can be used to identify the classification of an unknown letter given a classes integer data, but not the class itself. Using this dataset for classification is a perfect introductory project to binary classification.

I used multiple classification models were used for solving three different binary classification problems to test whether one classifier is more accurate at predicting the correct class than the other. After running the classification models on the datasets, I applied a sequential feature selection package to the datasets to reduce the number of features from 16 features to 4 features. Dimension reduction will reduce complexity, shorten computation time, and in most cases improve model accuracy due to the removal of redundant and/or misleading features. The sequential feature selector package that I used adds(forward selection) or removes(backward selection) features to form a feature subset in a greedy fashion. At each stage, the estimator chooses the best feature to add or remove based on the cross-validation score of an estimator.

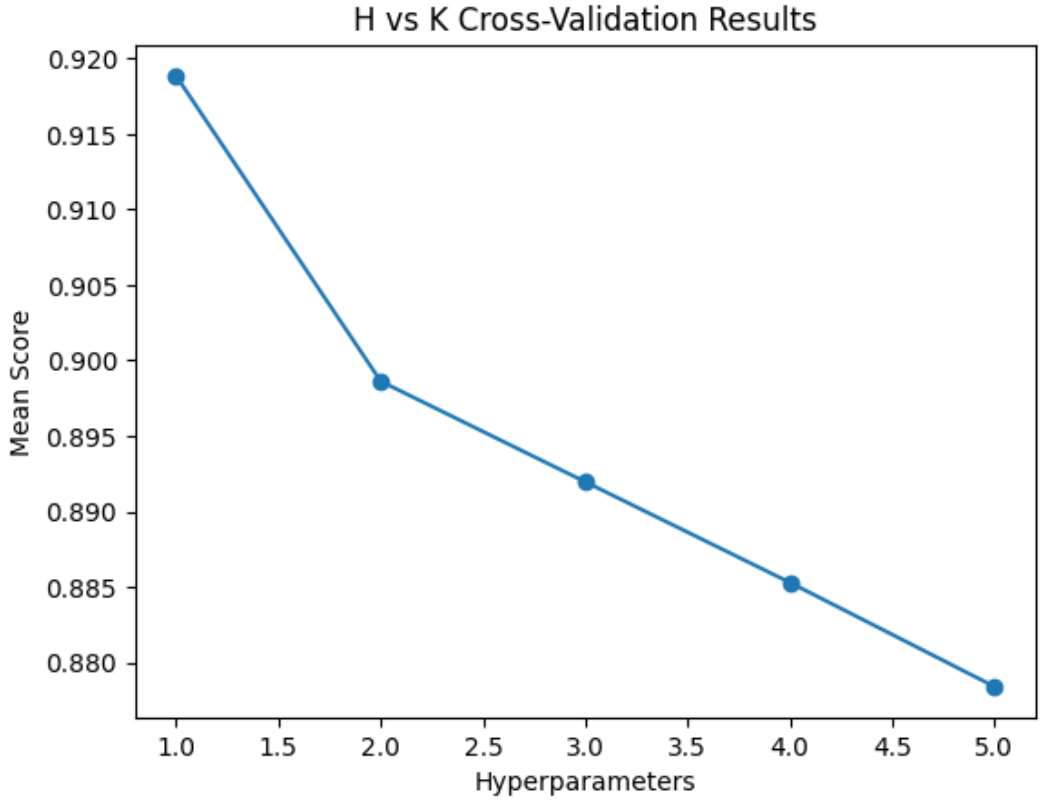
The three different binary classification models I solved for this programming project were H vs K, M vs Y and S vs L. I chose S vs L as the third set of letters to classify because they are the initials to my first and last name. Out of the three, I believed that M vs Y would be the most difficult to accurately classify.

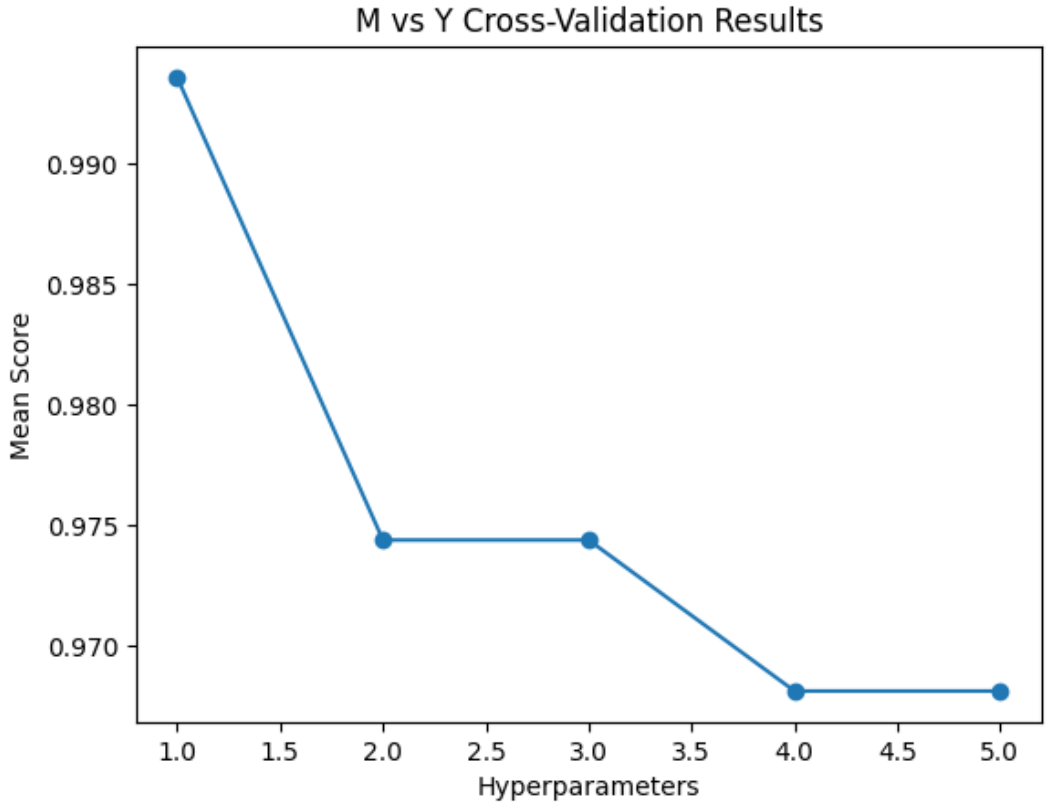
Results:

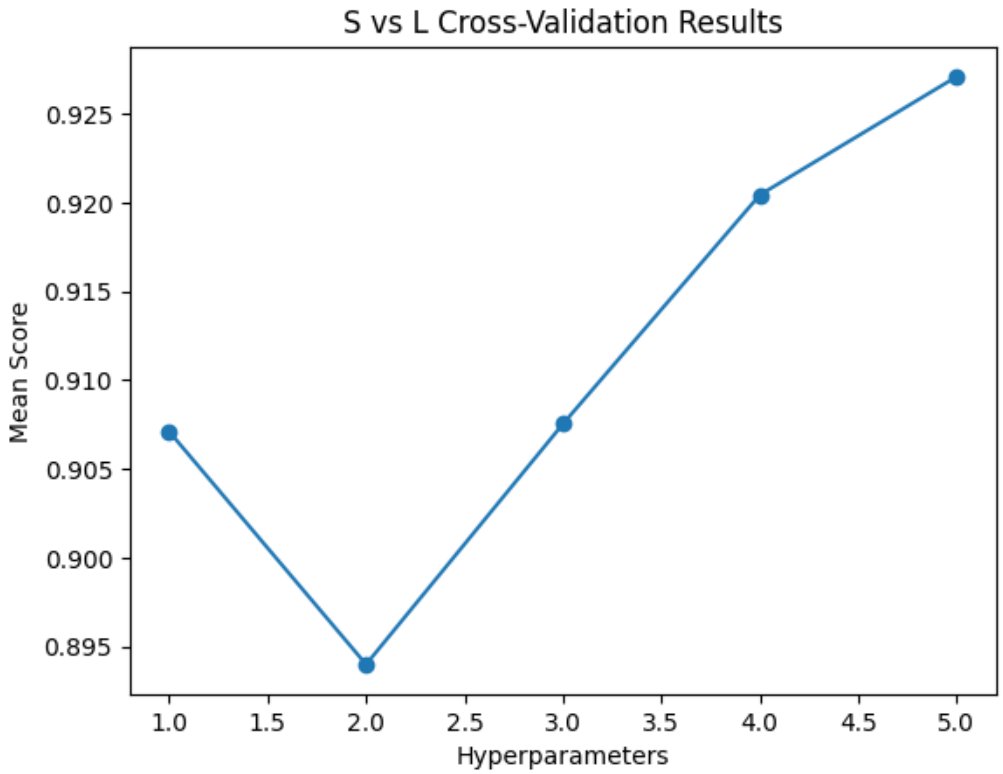
Support Vector Machine(SVM) Classifier:

The first classifier that I utilized was the Support Vector Machine. A SVM algorithm will seek to find a hyperplane in an N-dimensional space(N – number of features) that distinctly classifies the data samples. This classifier performs particularly well when there is a clear margin of separation between classes. It also performs well in high dimensional spaces. Which means that it may have the disadvantage when it comes to using dimension reduction to complement it. It also may not perform well if it is used for multi-class classification due to the overlap of target classes.

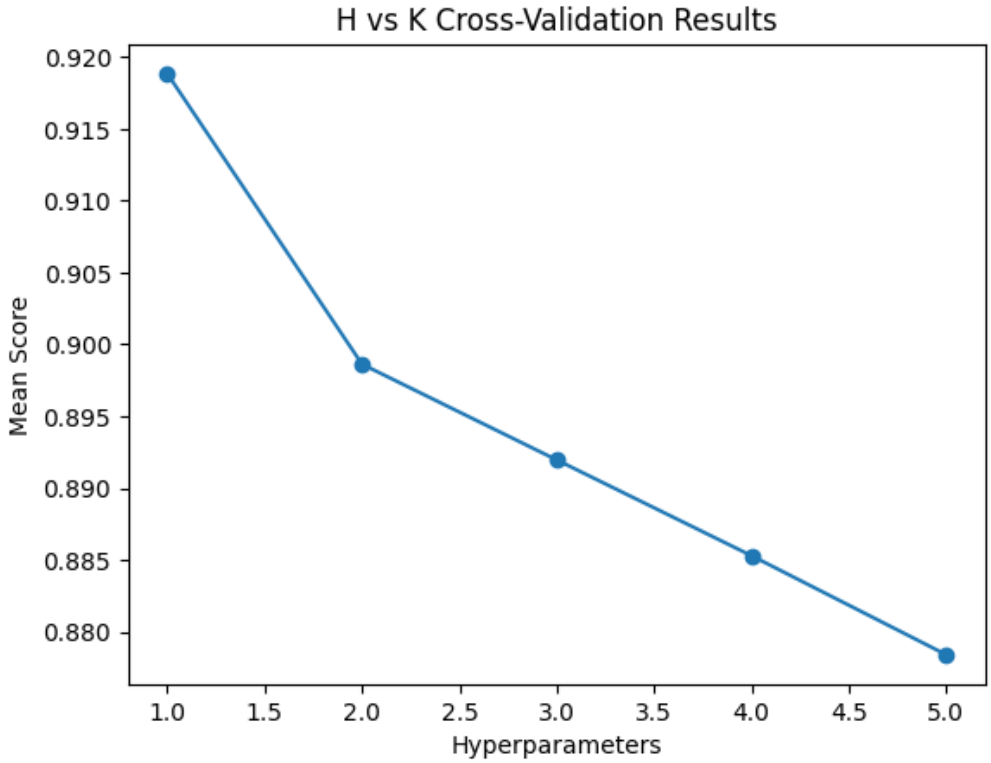
Cross-validation Results without Dimension Reduction:

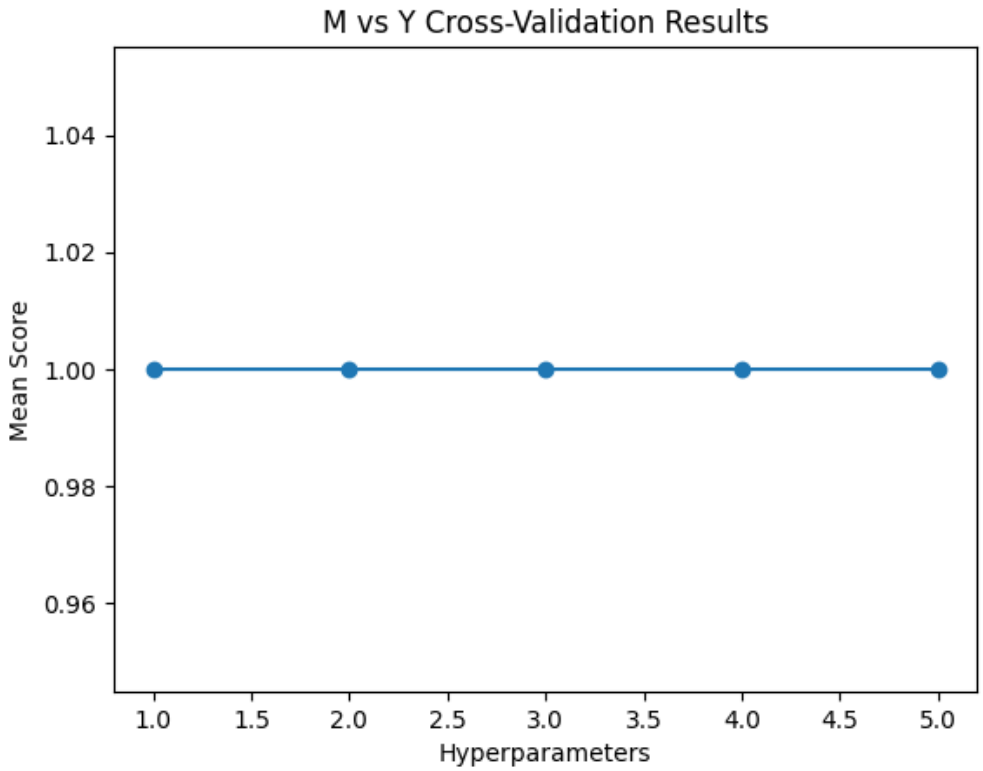


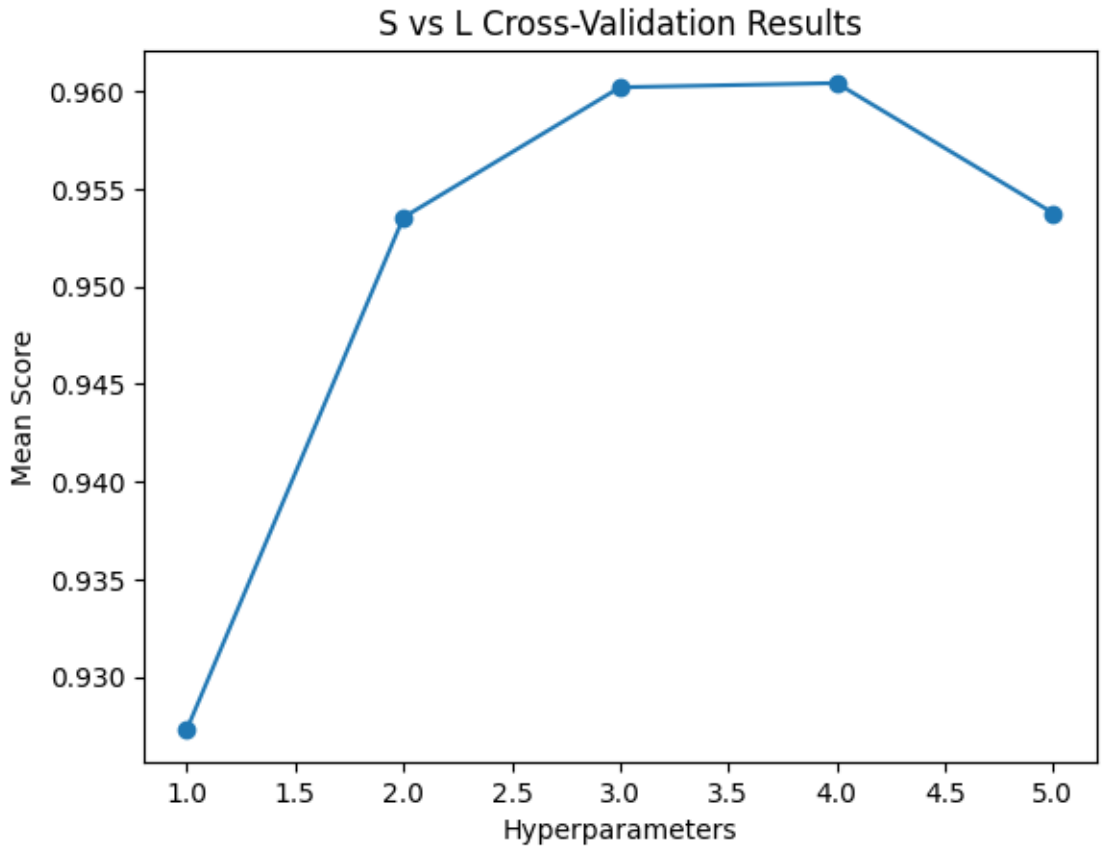




Cross-validation Results with Dimension Reduction:



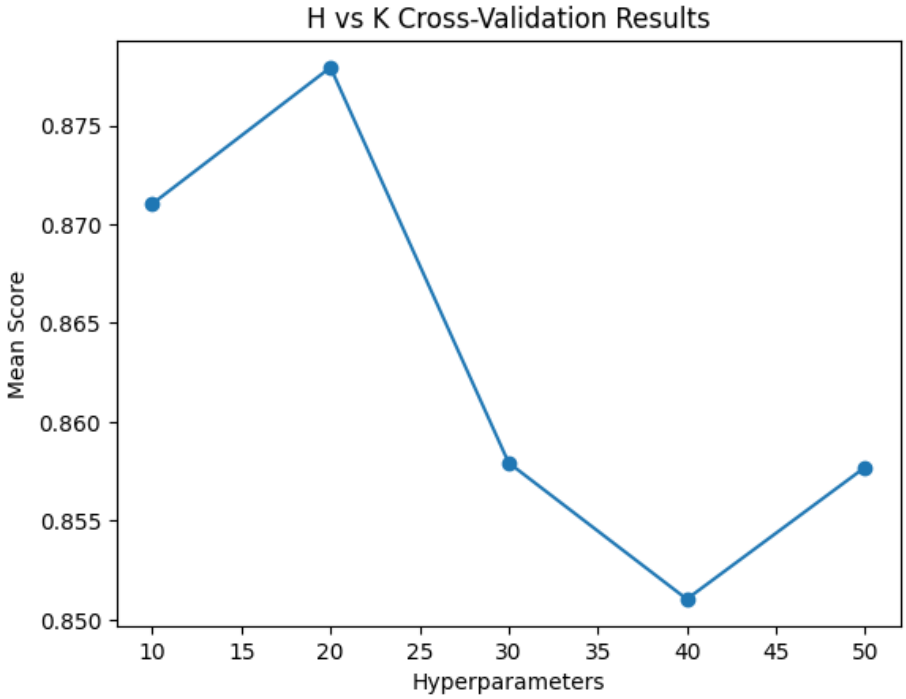

M vs Y Cross-Validation Results
Mean Score
1.04
1.02
1.00
0.98
0.96
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Hyperparameters

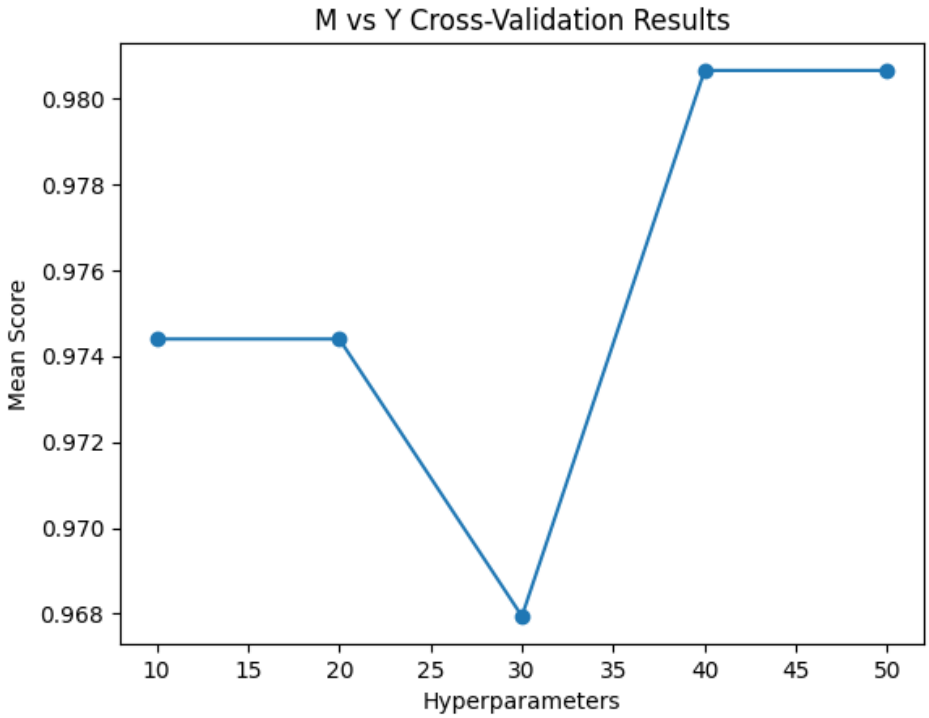

S vs L Cross-Validation Results
Mean Score
0.960
0.955
0.950
0.945
0.940
0.935
0.930
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Hyperparameters

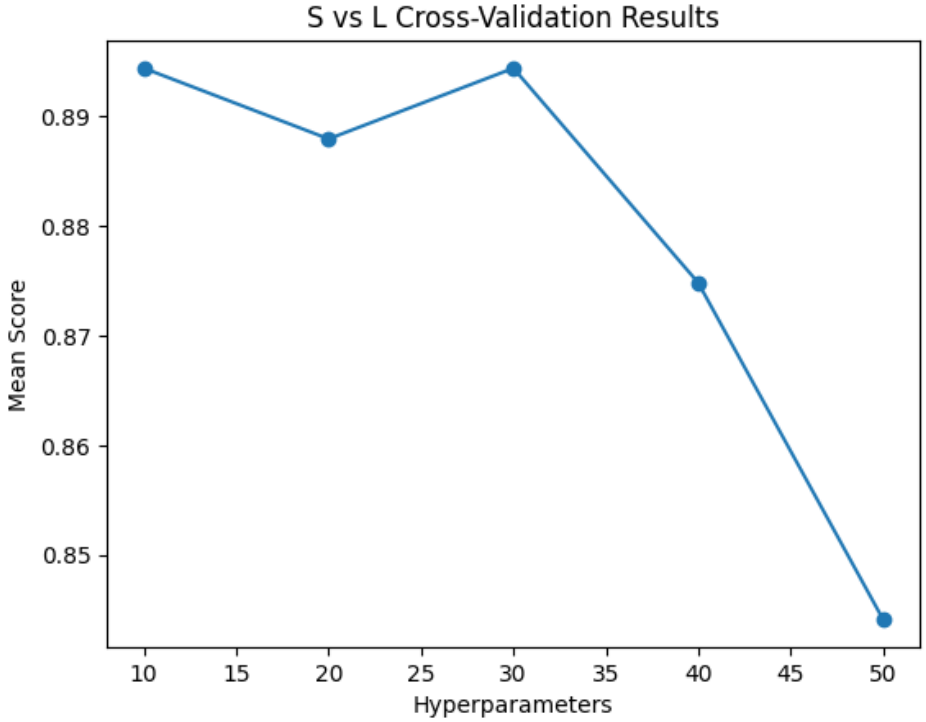
Decision Tree Classifier:

The decision tree classifier is a supervised learning method that is used for classification in this case. But it can be used for regression. The model predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree does not require normalized nor scaled data to infer these rules. It also handles missing values without issue. Some disadvantages of a decision tree classifier are that they usually require higher computation times and that small changes in the dataset it is using can often result in a large change in the structure of a decision tree causing instability

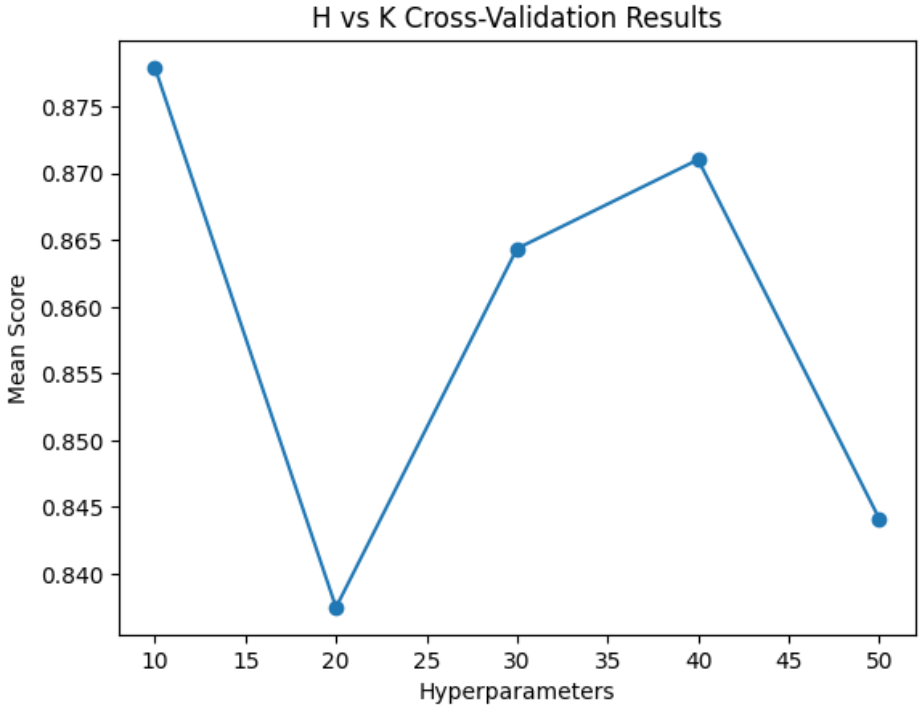
Cross-validation Results without Dimension Reduction:

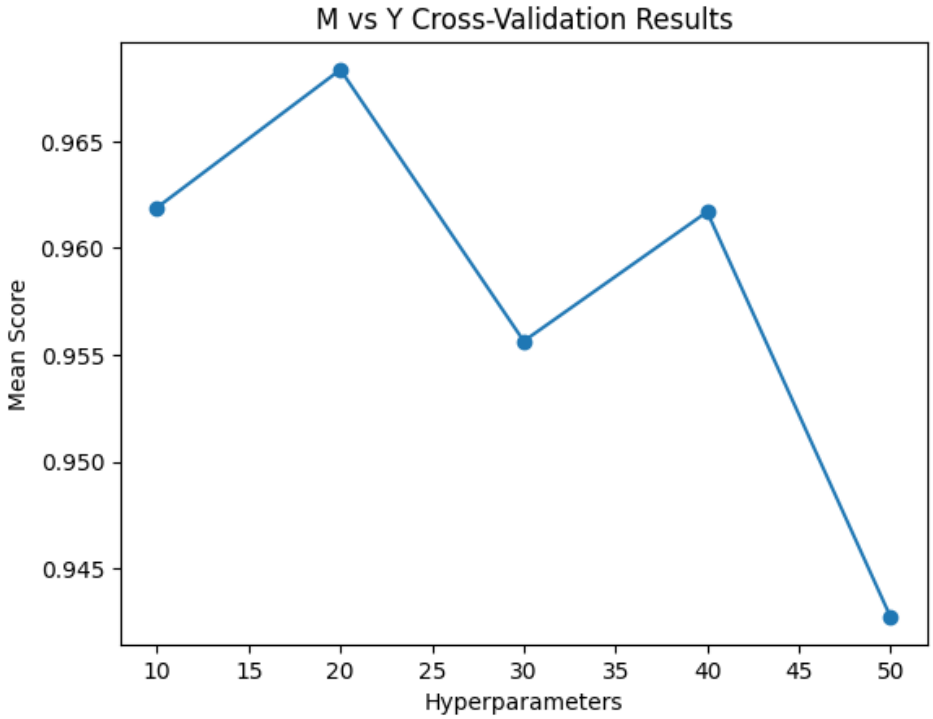


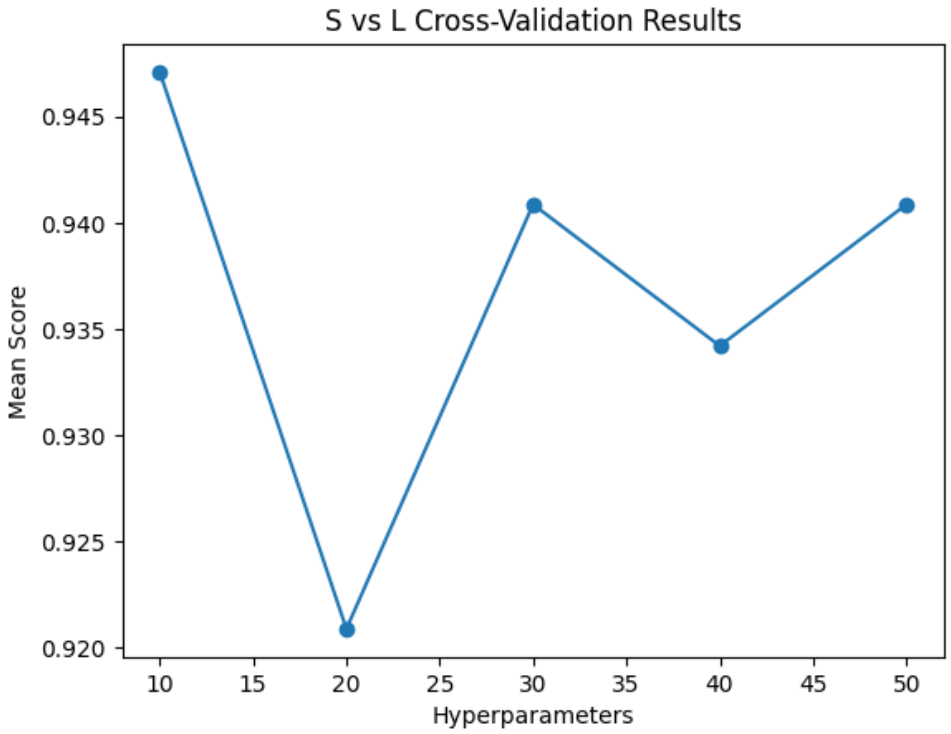

M vs Y Cross-Validation Results
Mean Score
0.980
0.978
0.976
0.974
0.972
0.970
0.968
10
15
20
25
30
35
40
45
50
Hyperparameters


S vs L Cross-Validation Results
Mean Score
0.89
0.88
0.87
0.86
0.85
10
15
20
25
30
35
40
45
50
Hyperparameters

Cross-validation Results with Dimension Reduction:







Elapsed Model Runtime in Seconds

	SVM	Decision Tree
H vs K	0.22120189666748047	0.04003620147705078
M vs Y	0.05604982376098633	0.03903555870056152
S vs L	0.10009098052978516	0.040036678314208984
H vs K Dimension Reduced	0.2232017517089843 8	0.039940834045410156
M vs Y Dimension Reduced	0.05705094337463379	0.045040130615234375
S vs L Dimension Reduced	0.09890627861022949	0.03903627395629883

Discussion:

The SVM models without dimension reduction perform accurately overall. But their performance seems to diminish as the hyper parameter is tuned to increasing increments. The runtime of the SVM models without dimension reduction take the longest to run according to the table above. While the decision tree classifier models without dimension reduction on the other hand run about twice as fast as the SVM models. The decision tree classifier models without

dimension reduction perform accurately, but the accuracy is sporadic as the hyper parameter in tuned to increasing values.

The performances of the SVM models with dimension reduction seem to perform better than the SVM models without dimension reduction. Especially in regards to M vs Y binary classification and S vs L binary classification. Although the accuracy of the models seem to improve with dimension reduction, there seems to be very little improvement in the runtime of the models according to the table above, however. The decision tree classifier models with dimension reduction on the other hand seem to improve the accuracy at certain hyperparameters, but diminish in accuracy at hyperparameters they previously performed well with prior to dimension reduction.

If I were to perform this experiment again, I would probably still use SVM for binary classification because it responded very well prior to dimension reduction and with near excellence after dimension reduction. I would also scrap the Decision Tree classifier to try another classifier out that could possibly yield better performance. I think that the KNN classifier may have been the better model for binary classification in retrospect.