EAS506

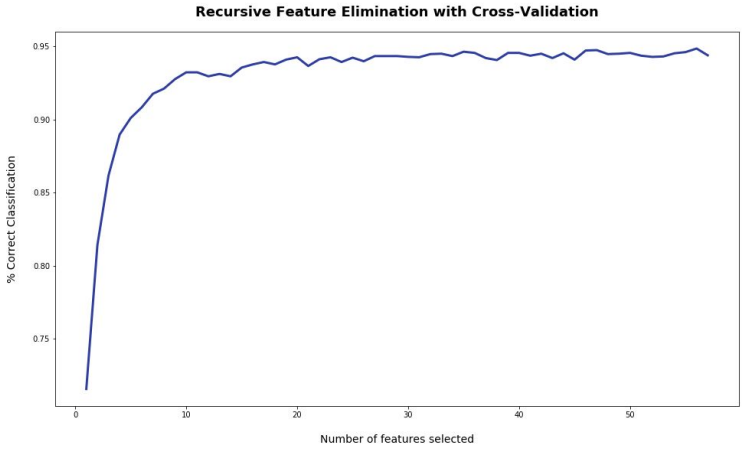
Written Report

HW5

Matthew Sah

1)

Using the RFECV function written inside sklearn, we can easily implement Cross-Validation with a simple classifier in which in this case i used Random Forest for it's more elasticity and ability to solve non linear models which in this case worked perfectly. After processing it returned an optimal feature selection of 50 predictors. However we can see the accuracy started to reach stability at around 10 feature and only gradually increased as predictors increased. During the process i've used both 10 and 50 for a single layer in the hidden layer, there were little to no difference in accuracy, therefore i deduced that the 0.01 % could be considered unnecessary.



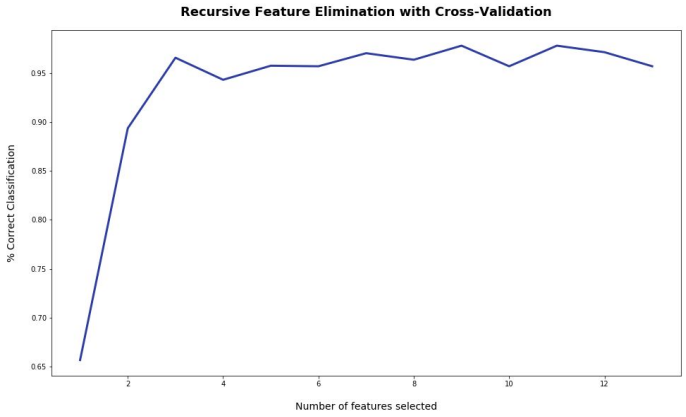
During the process i also tested with multiple layers in the hidden layer with the MLP library in sklearn which enables more accuracy control over the process of neural networks. during my tests since with a single layer of 50 nodes could already create a 90+% accuracy for this dataset, there were again little to no difference with multiple layers, however it's pretty frequent that with multiple layers it reduces the accuracy, i believe this to be because of the depth of the dataset.

Comparing neural networks to more traditional additive models like logistic regression showed little difference in this case, for logistic regression i received a 92% while with NN i received 94%. Again i believe if the dataset were to be non-linear it would be more easily to differentiate

the advantages between theses two datas. Another thing to note is that the neural network though not significant, still took longer than the time on logistic regression.

2)

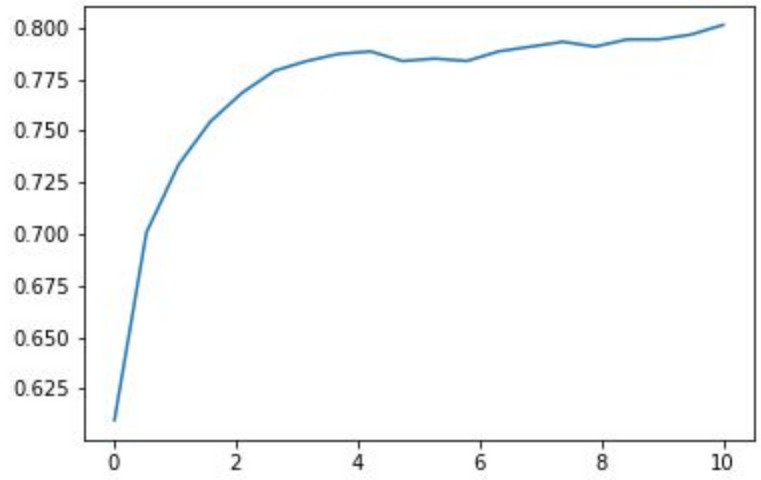
Going through different classification datasets (iris, wine ...etc), i settle with the wine data set considering it's diversity and number of predictors in comparison to easier datasets(iris). I created my univariate outlier by listing it as a value 10^14 times larger than its original value. Again with RFECV, i was able to find the number of useful predictors and use it as the nodes in the hidden layer which was in my case reached its peak accuracy with 9 variables but had very minor difference and increments with 3+ predictors.



The outlier made a great impact for the test results of the modified data, only reaching 64% percent accuracy when the unmodified data could reach 94% easily. I later shrunk the difference of the outlier to a closer value, to 10^8, 10^3 respectively. The difference in accuracy was immediate although not significant, with 67% and 70% respectively. When tested with outliers only 10^1 away the accuracy could easily reach 80+% accuracy; I believe this is due to the outlier causing an possible exponential difference for the data.

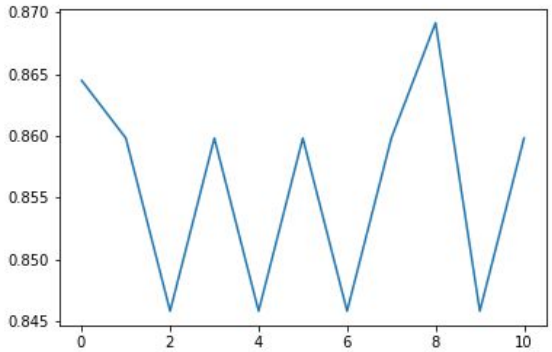
3)

For this dataset, i removed "Store7" since it's values were binary and i believed unlikely to affect the results in this case.

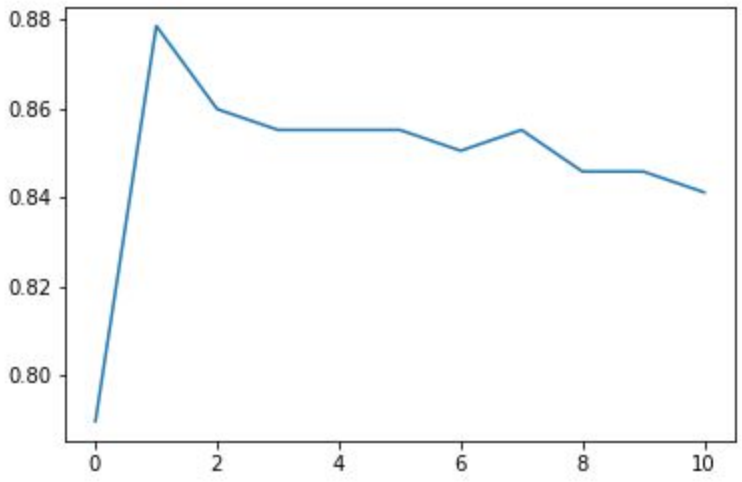


Running the datas through the SVC library under the sklearn.SVM libraries, we can easily change the cost coefficients(C = [0.1~10]). We can see that apart from when the cost was less than around 1, the accuracies varied very little and starting from when cost is equaled to 1.06 we get an accuracy of 73% with slight increments as cost increases. Max accuracy was reached with 80% at cost equals 10.

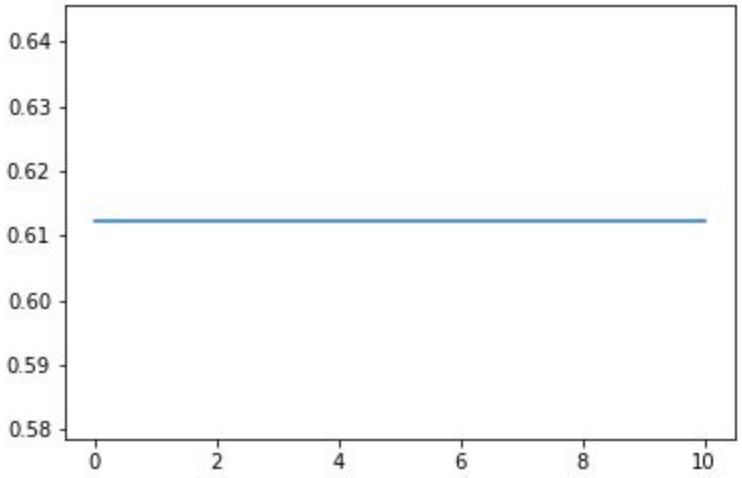
For (B) after changing to the polynomial kernel, we immediately notice one major difference, which was the time required to run through all cost variations. With the default settings, we can get the results in less than a second, however with the polynomial kernal in two degrees it takes more than 10 times the time required for the default settings, and also constantly causing memory and freezing issues. However the results are very consistent with an accuracy average of 85 throughout the cost variations.



I also tested this with the linear kernel inside SVC, it too significantly less time than the polynomial option but still took longer than the default settings(RBF). It displayed almost the same results as the default settings.



Finally i tested it with the sigmoid kernel with was the last but most interesting of the options (i was not confident enough to write my own callable kernel). The sigmoid option displayed the same accuracy throughout all the cost variations but were all low and insufficient to be considered useful.



Out of all the options the polynomial options took significantly longer and would take longer if the degree of polynomial went up. Despite the longer time required the polynomial option also displayed the highest and most stable accuracy out of all the options with an average of 86%.