CHEM498 Assignment (GK) 3

Prepare Data for Analysis

Sabine Plummer (40087050)

27/11/2020

Supporting Documentation

I chose to run KNN models and support vector machine (SVM) models, both of which are supervised classifiers. I actually tried changing the P value for a different training/test ratio of observations in the KNN model, but it seemed like the 70:30 line was as small as I could get for my test group and still include all 7 classes of wine quality. When I tried running it at 80:20, even changing setseed, I kept coming upon an error saying I didn't have the same number of levels for both my training and test sets. Hence, I ended up playing around with the tunelength (also in part because our class effort in finding a way to change the KNN distance measurement from Euclidean to anything else came to a communal wall), and I was curious as to its function. As for my SVM model, I ended up having to switch over to a cross-validation because the code was taking way too long to run. I ran a linear kernel first, then tried my hand at a non-linear kernel to see what would happen. Overall, what took the longest was actually running the script on my poor old laptop. Seeing as I'm most likely going to run machine learning scripts on my final project, I'm already anticipating more long waiting times and possibly lower validation credibility for the sake of brevity.

Model = KNN

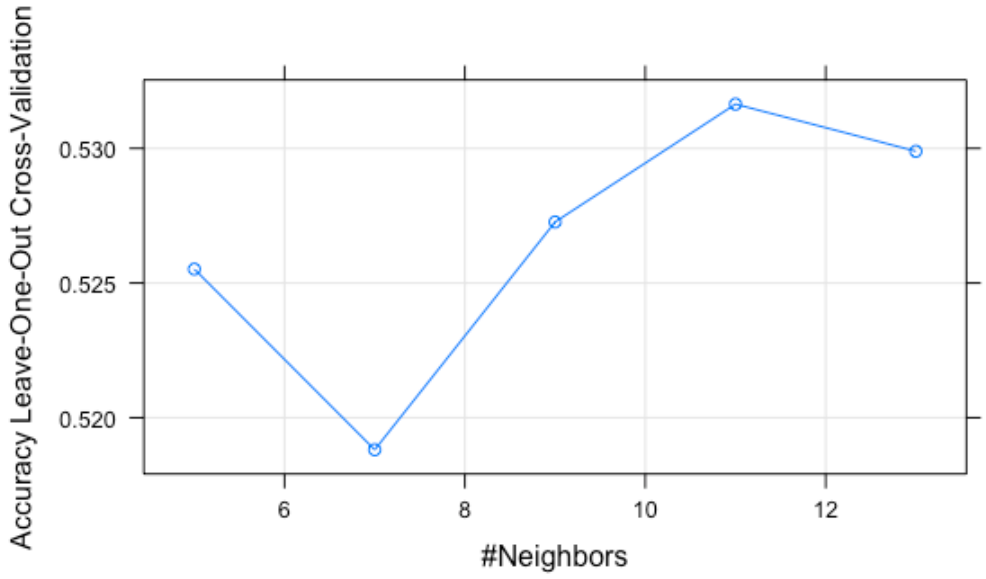
Parameters:

P = 0.7

Validation = LOOCV (full cross-validation)

Tunelength = 5

This model saw its most accurate results at a value of k=11. Most accurate, being a bit above 53%. Though my expert coding skills are little more precise than a coin flip, the issue seems to mostly be with the accuracy of "fringe" quality wines. Though sensitivity goes up to close above 60% for wines like class 6, it's virtually non-existent for wine of quality 3 or 9. In fact, outside of the 5-6-7 range, the is little if any sensitivity. I have an inkling this is due to the general lack of these quality wines in the training and test set, meaning they have few replicates of which to test. The model thus is able to test much more accurately these mid-range quality wines of which there were more of.



```
Overall Statistics

               Accuracy : 0.5459
                 95% CI : (0.5201, 0.5716)
    No Information Rate : 0.4486
    P-Value [Acc > NIR] : 4.68e-14

                  Kappa : 0.2958

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7  Class: 8
Sensitivity          0.000000 0.055556   0.5701   0.6707  0.39706 0.0222222
Specificity          1.000000 0.998587   0.8279   0.5765  0.89808 0.9845506
Pos Pred Value            NaN 0.600000   0.5822   0.5631  0.46957 0.0434783
Neg Pred Value       0.997958 0.965164   0.8207   0.6827  0.86764 0.9695712
Prevalence           0.002042 0.036760   0.2961   0.4486  0.18516 0.0306331
Detection Rate       0.000000 0.002042   0.1688   0.3009  0.07352 0.0006807
Detection Prevalence 0.000000 0.003404   0.2900   0.5344  0.15657 0.0156569
Balanced Accuracy    0.500000 0.527071   0.6990   0.6236  0.64757 0.5033864
                     Class: 9
Sensitivity          0.0000000
Specificity          1.0000000
Pos Pred Value             NaN
Neg Pred Value       0.9993193
Prevalence           0.0006807
Detection Rate       0.0000000
Detection Prevalence 0.0000000
Balanced Accuracy    0.5000000
> 
```

Model = KNN

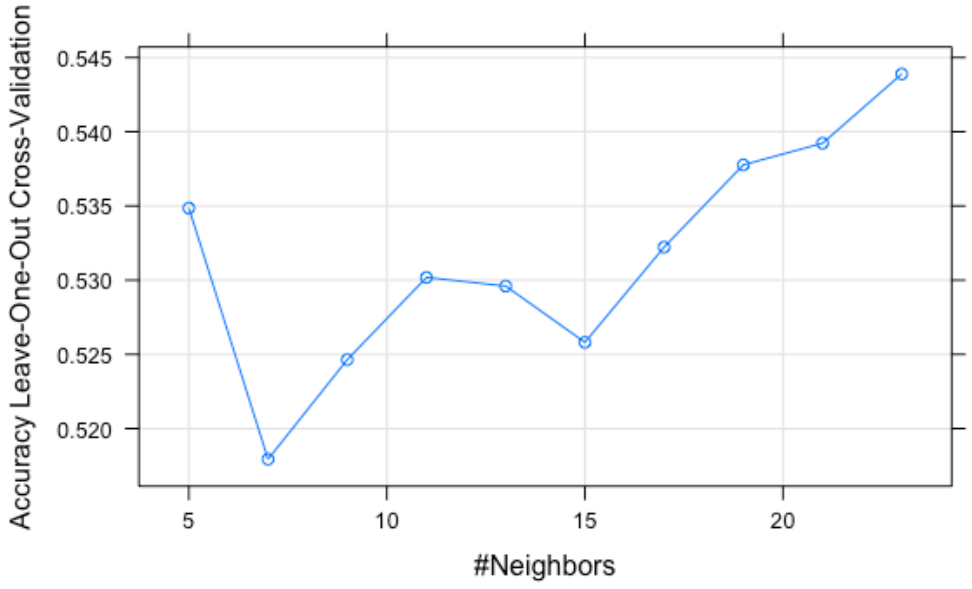
Parameters:

P = 0.7

Validation = LOOCV (full cross-validation)

Tunelength = 10

Here is the same model with just the tunelength changed from 5 to 10. A short google search yielded that the tunelength tells the code how many different default values for the main parameters (in this case, k). I noticed the number of nearest neighbours tried went up to almost 25 instead of 15 in the original model. This does run the risk of overfitting, especially since there seems to be a linearly increasing accuracy based on the increasing number of nearest neighbours. We do only get an increase of about 2% accuracy, which also supports this hypothesis. We also see the same issues with this model, fitting the middle three quality options (>70% class 5!) but tapering off in the edges.



```
Overall Statistics

               Accuracy : 0.5541
                 95% CI : (0.5283, 0.5797)
    No Information Rate : 0.4486
    P-Value [Acc > NIR] : 3.324e-16

                  Kappa : 0.2954

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000  0.00000   0.5655   0.7117  0.36397 0.000000
Specificity          1.000000  1.00000   0.8433   0.5346  0.90727 0.996489
Pos Pred Value            NaN      NaN   0.6029   0.5544  0.47143 0.000000
Neg Pred Value       0.997958  0.96324   0.8219   0.6950  0.86259 0.969262
Prevalence           0.002042  0.03676   0.2961   0.4486  0.18516 0.030633
Detection Rate       0.000000  0.00000   0.1675   0.3193  0.06739 0.000000
Detection Prevalence 0.000000  0.00000   0.2777   0.5759  0.14295 0.003404
Balanced Accuracy    0.500000  0.50000   0.7044   0.6231  0.63562 0.498244
                      Class: 9
Sensitivity          0.0000000
Specificity          1.0000000
Pos Pred Value             NaN
Neg Pred Value       0.9993193
Prevalence           0.0006807
Detection Rate       0.0000000
Detection Prevalence 0.0000000
Balanced Accuracy    0.5000000
> 
```

Model = SVM (linear kernel)

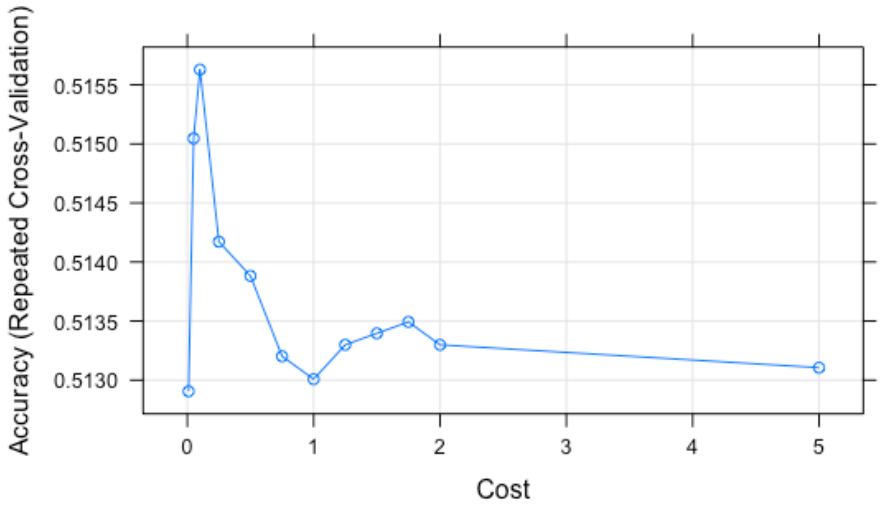
Parameters:

P = 0.7

Validation = repeatedcv (cross-validation)

Tunelength = 5

Support vector machine model, which took some heavy reading into (sources have been listed at the bottom of the R script) to begin to understand. Still, I believe I've come to grasp at least a significant portion of the theory! As such, I was excited to see what a non-linear kernel would give. This, below, is the linear svm model, run with a cross-validation (for the sake of my computer's health and my sanity) and a shorter tunelength to avoid over fitting. Particularly, I had to set a range of values for the C value and define a tunefrid parameter to yield proper tuning of my model instead of an assumed C=1 value. As is illustrated below, 1 was by far not the best weighing for this model. Even with tuning, however, I could only achieve accuracy barely above 50%. Moreover, the range of accurate classification was more narrow than the KNN, allowing only qualities 5 and 6 to be accurately (>80% for 6!) recognized. Yet, the kappa score is well below the already sad KNN kappa value.



```
Overall Statistics

               Accuracy : 0.5235
                 95% CI : (0.4976, 0.5493)
    No Information Rate : 0.4486
    P-Value [Acc > NIR] : 5.137e-09

                  Kappa : 0.1981

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000  0.00000   0.5333   0.8149   0.0000  0.00000
Specificity          1.000000  1.00000   0.8250   0.3593   1.0000  1.00000
Pos Pred Value            NaN      NaN   0.5617   0.5085      NaN      NaN
Neg Pred Value       0.997958  0.96324   0.8078   0.7046   0.8148  0.96937
Prevalence           0.002042  0.03676   0.2961   0.4486   0.1852  0.03063
Detection Rate       0.000000  0.00000   0.1579   0.3656   0.0000  0.00000
Detection Prevalence 0.000000  0.00000   0.2811   0.7189   0.0000  0.00000
Balanced Accuracy    0.500000  0.50000   0.6791   0.5871   0.5000  0.50000
                     Class: 9
Sensitivity          0.0000000
Specificity          1.0000000
Pos Pred Value             NaN
Neg Pred Value       0.9993193
Prevalence           0.0006807
Detection Rate       0.0000000
Detection Prevalence 0.0000000
Balanced Accuracy    0.5000000
>
```

Model = SVM (non-linear kernel) *Fair warning, this took forever to run, even with a cross-validation

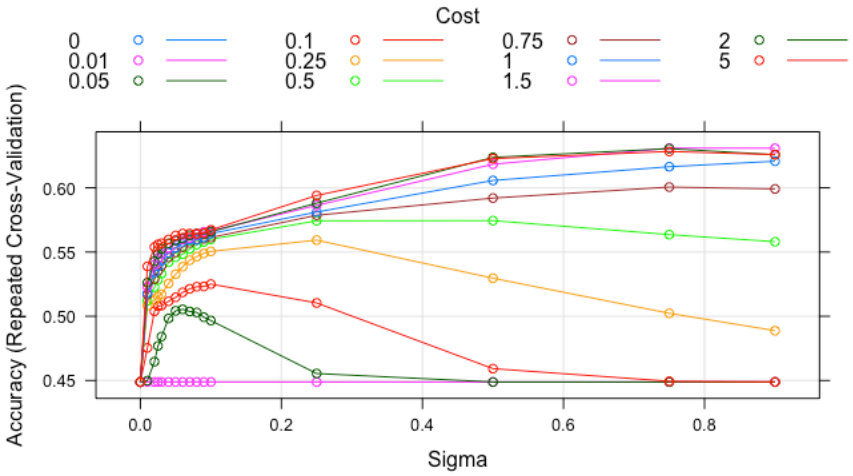
Parameters:

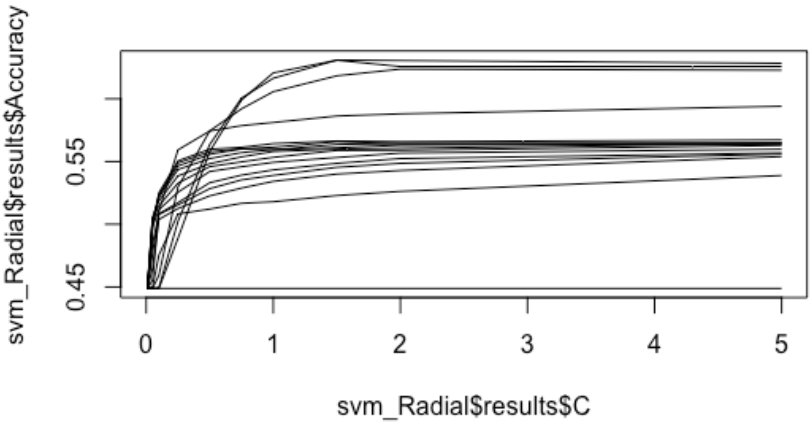
P = 0.7

Validation = repeatedcv (cross-validation)

Tunelength = 5

Taking a look at the non-linear fit for the wine data, the tuning ended up choosing a sigma of 0.75 and a c of 1.5. The accuracy graph seems to indicate some overfitting, especially in the sigma value as it increases past the 0.1 value. The C value would most likely have been somewhere between 1.5 and 5 depending on the final sigma value used anyway. There does seem to be a plateau in increasing accuracy around C=2 for the most accurate of model parameters. That being said, the obtained model has some overfitting, but does show a large increase in accuracy and range compared to the other methods so far. The accuracy is clearing 64%, the kappa value is the highest of all the models so far, and the classes showing sensitivity go from 4 to 8, the most of all models. As such, I believe this model, with perhaps a bit of further tuning to make sure it doesn't catch too much noise, would be the best model for this data (of those tested).






```
Overall Statistics

               Accuracy : 0.6413
                 95% CI : (0.6161, 0.6658)
    No Information Rate : 0.4486
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4305

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000 0.111111   0.5908   0.8194  0.47059 0.244444
Specificity          1.000000 0.996466   0.9023   0.5617  0.94820 0.997191
Pos Pred Value            NaN 0.545455   0.7179   0.6034  0.67368 0.733333
Neg Pred Value       0.997958 0.967078   0.8398   0.7927  0.88741 0.976616
Prevalence           0.002042 0.036760   0.2961   0.4486  0.18516 0.030633
Detection Rate       0.000000 0.004084   0.1749   0.3676  0.08713 0.007488
Detection Prevalence 0.000000 0.007488   0.2437   0.6093  0.12934 0.010211
Balanced Accuracy    0.500000 0.553789   0.7466   0.6906  0.70940 0.620818
                      Class: 9
Sensitivity          0.0000000
Specificity          1.0000000
Pos Pred Value             NaN
Neg Pred Value       0.9993193
Prevalence           0.0006807
Detection Rate       0.0000000
Detection Prevalence 0.0000000
Balanced Accuracy    0.5000000
> 
```