Predicting COVID19 Cases

Rees LaBree and Zach Morrissey

The goal of this project is to predict new COVID case numbers per day for each county in Colorado. We will predict using several different models and features to tune and find the best hyperparameters, documenting progress along the way.

Our Github page is https://github.com/zmorrisseyj/CSCI4622.

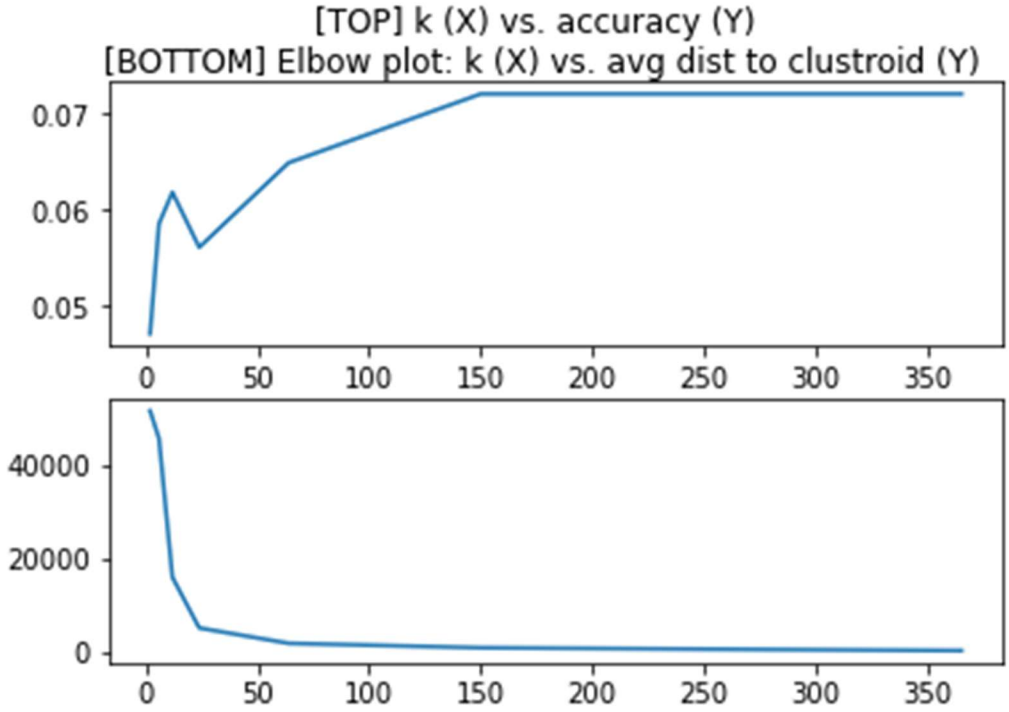
Data pulled from:

- Colorado Department of Public Health and Environment (CDPHE)
- Colorado Department of Revenue

K-Means

Our K-Means approach clusters data points on manhattan distance using all features. The features are Days Since March 17, County, Population, Total Sum Cases, New Cases per Day, Average Cases Per 7 Days, Average Daily Change in Cases Per 7 Days, Area of County, and Number of MIP Charges. Since the manhattan distance is used and all features are equally weighted, we think that days since march 17 and total sum cases are skewing the clusters since they chronologically increase. Further inquiry into performance would be to mess with feature dropout and random weighting to see what results in the highest accuracy. Runtimes for our 7 values of k with a single set of hyperparameters takes about 3 hours, so further tuning wasn't possible under our time constraints. These runtimes are to be expected because K means clustering is an NP-hard problem, so even for our relatively small data set we see long runtimes.

K means is also not a supervised learning method, so predicting a value using K means was unclear and we settled on an intuitive but unproven method. This method is described in the code, but essentially averages a random sample from the closest cluster of points.



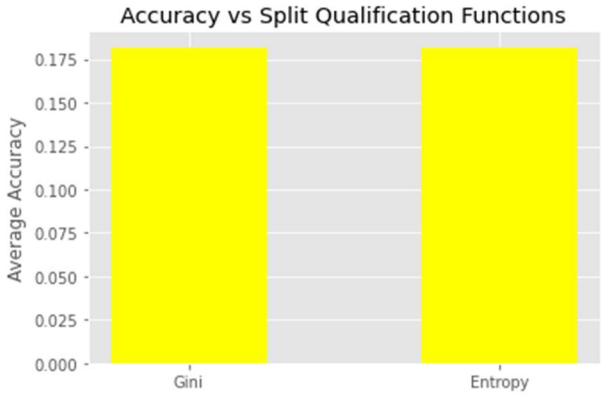
As we can see from our results, the accuracy had a significant positive correlation which is reassuring. Our elbow plot behaved as expected, further assuring the K means implementation is correct. The overall accuracy however is quite poor, ranging from 4% to 7% accuracy. This is obviously very poor, and a baseline that we would like to compare to is the random classifier. As mentioned in the details, we know that some features were bad. To get around this, we would want to run this with random feature sets to see which features behave poorly other than the ones we already suspect. Most likely we would run these random feature sets with k = 24 as suggested by our elbow plot. If we are still not satisfied after feature set tuning, we can then tune feature weights.

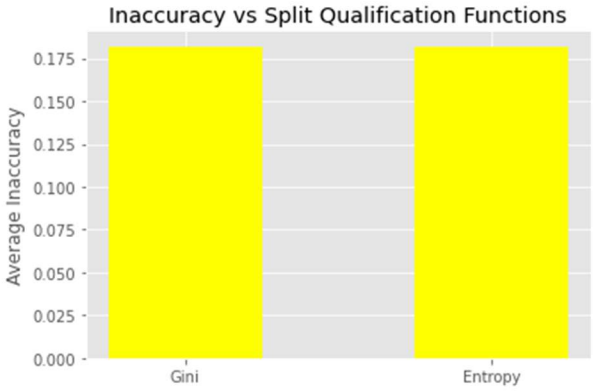
Random Forest and Adaboost

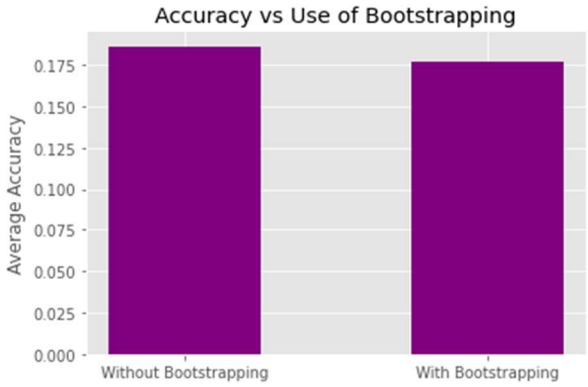
Since the predictive qualities of K-Means were poor, to say the least (although excellent at grouping like days), we decided to try and use Random Forest and Adaboost to get better predictive results. We went with the built-in methods for SciKit Learn to save a little time and skip right to hyperparameter tuning. We iterated through different values for the number of estimators, the qualification functions for splits, and whether we applied bootstrapping. The results are graphed on the next page.

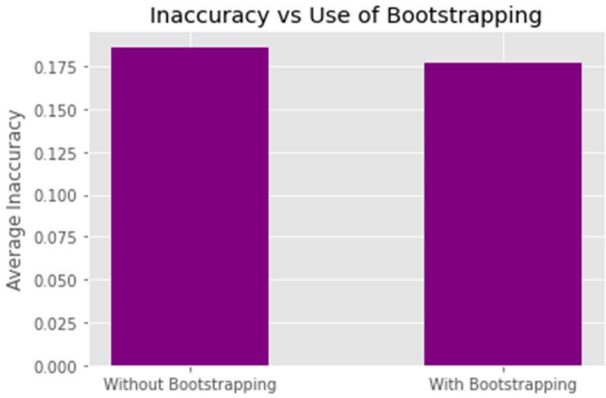
As you can see by the graphs, it appears that changes in the number of estimators used, the use of bootstrapping, or differences in split function accuracy equations made a remarkable change in accuracy, although there was a small average improvement in inaccuracy with the use of bootstrapping.

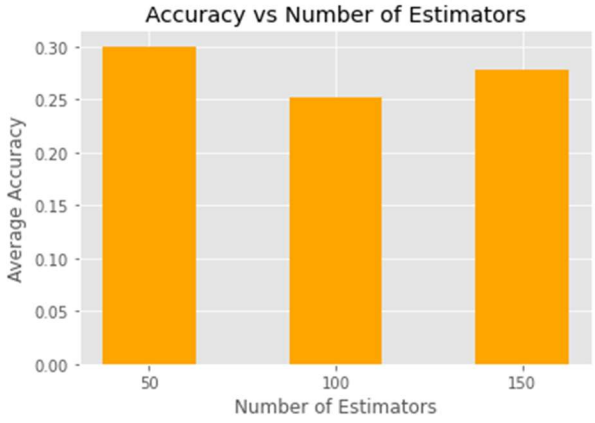
The best parameters in combination in Random Forest yielded an accuracy of 63.6%, and the an inaccuracy of 17.4%. We are defining an accurate guess as a prediction within +/-25% of the actual covid case numbers for a particular day, and an inaccurate guess as a prediction outside of +/-75% of the actual covid case numbers for a particular day.

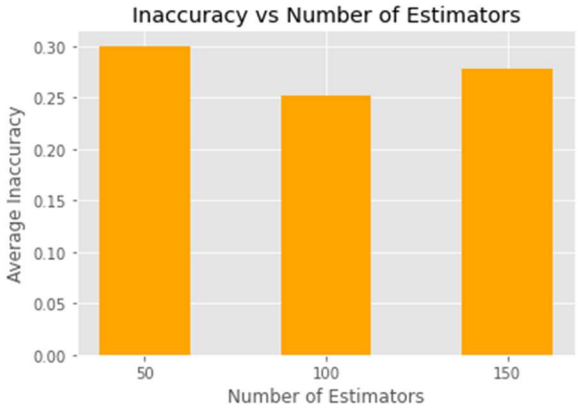

Accuracy vs Split Qualification Functions
Average Accuracy
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Gini
Entropy


Inaccuracy vs Split Qualification Functions
Average Inaccuracy
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Gini
Entropy

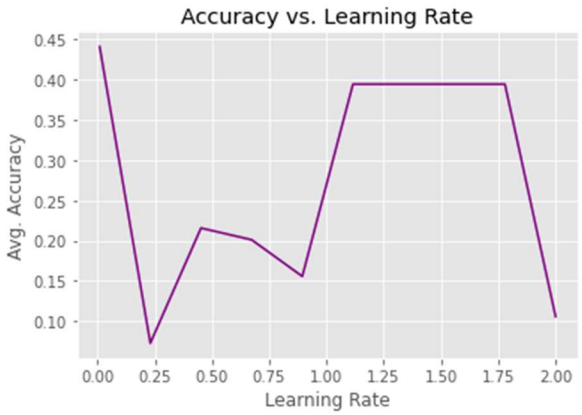

Accuracy vs Use of Bootstrapping
Average Accuracy
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Without Bootstrapping
With Bootstrapping

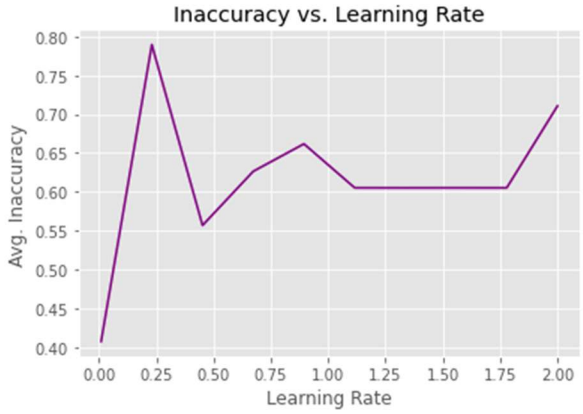

Inaccuracy vs Use of Bootstrapping
Average Inaccuracy
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Without Bootstrapping
With Bootstrapping

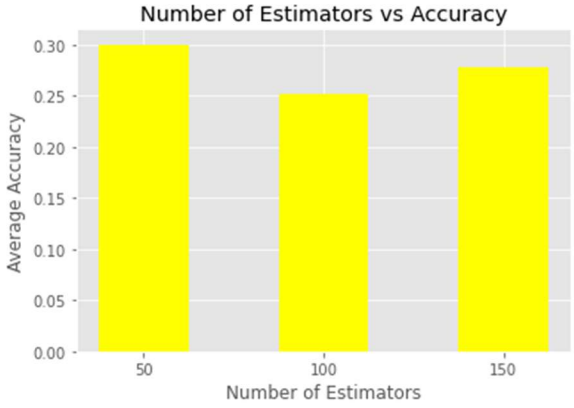

Accuracy vs Number of Estimators
Average Accuracy
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
Number of Estimators

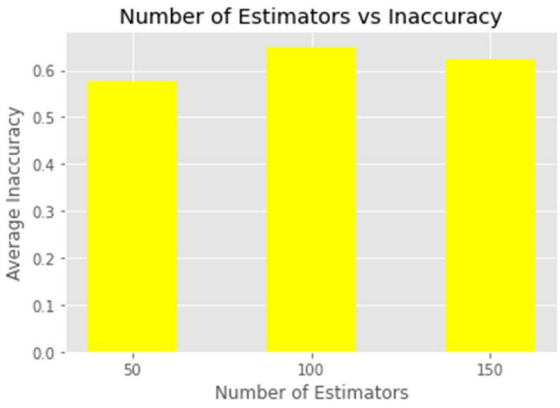

Inaccuracy vs Number of Estimators
Average Inaccuracy
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
Number of Estimators

Next we turned to Adaboost to see if we could get any better results than with Random Forest. Adaboost parameter tuning yielded similar results to Random Forest, and changes to the parameters didn't make a huge difference in the net outcome. The best combination of parameters in Adaboost yielded an 45.0% accuracy and 29.5% inaccuracy, significantly worse than Random Forest. Below are the results of tuning Learning Rate and the number of estimators.









Finally, using our best parameters from Random Forest we trained Random Forest on March 31st data and prior, and made predictions on the month of April. This lead to 58.3% accuracy and 26.9% inaccuracy.

Overall, we were able to meet similar accuracy and inaccuracy values between the training and qualification portion and then applying the best hyperparameters to April after training on March and prior. We consider this somewhat of a success. It is unfortunate not to be able to predict cases within 15% with a higher accuracy, however, the low inaccuracy rate of roughly 25% is a success. In our opinion, a high inaccuracy rate would be worse than a low accuracy rate from the standpoint that predicting a low number of cases when the actual number of cases is exceptionally high would have worse effects than predicting a low number of cases and actually getting a medium number of cases.