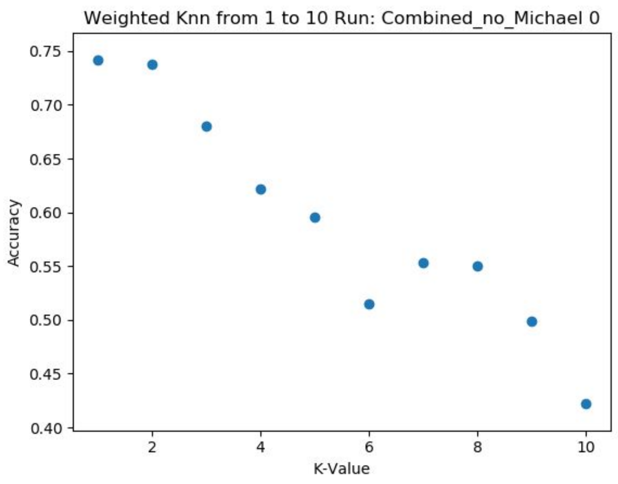
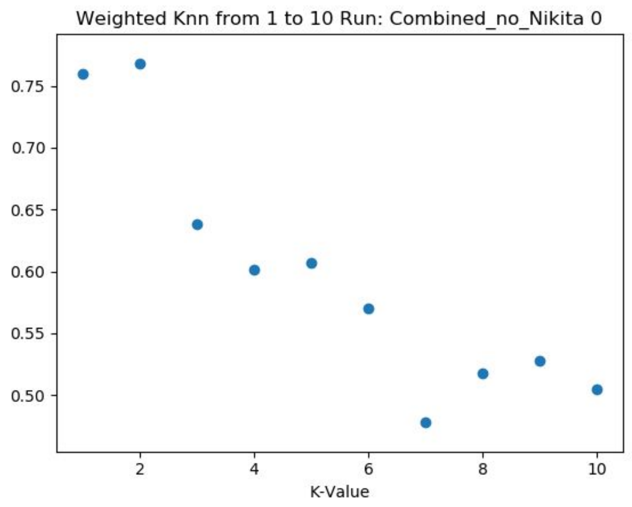
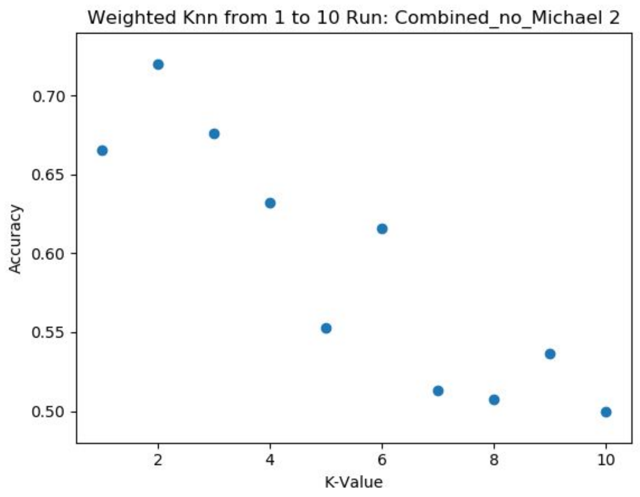
Project Part 1
3/21/19
Team TRM
TRUNG, NIKITA, ROSEMOND & MICHAEL

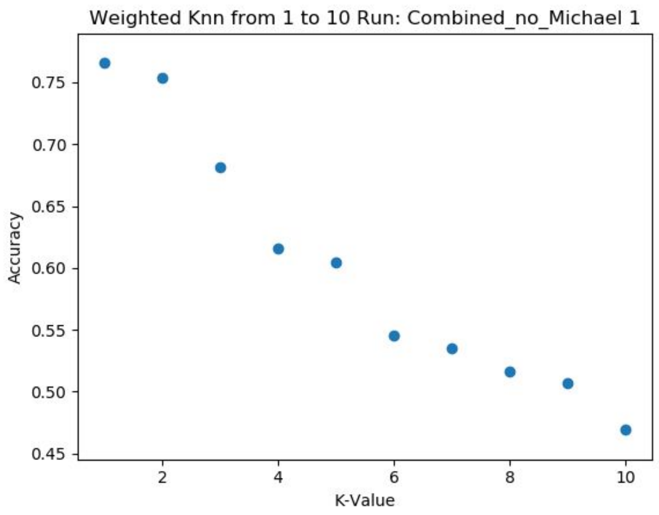
1.) How good is your performance when you train (and/or validate) using all but one teammember's data and test on the remaining teammember's data? Repeat this for each teammember being the test case multiple times (at least 3x each). Back up your response with graphs and/or tables of results.

All of our pictures were about the same, and we didn't have much noise in our backgrounds like other image datasets. As a result of this, when we took out individuals images, we found our scores to stay the same.



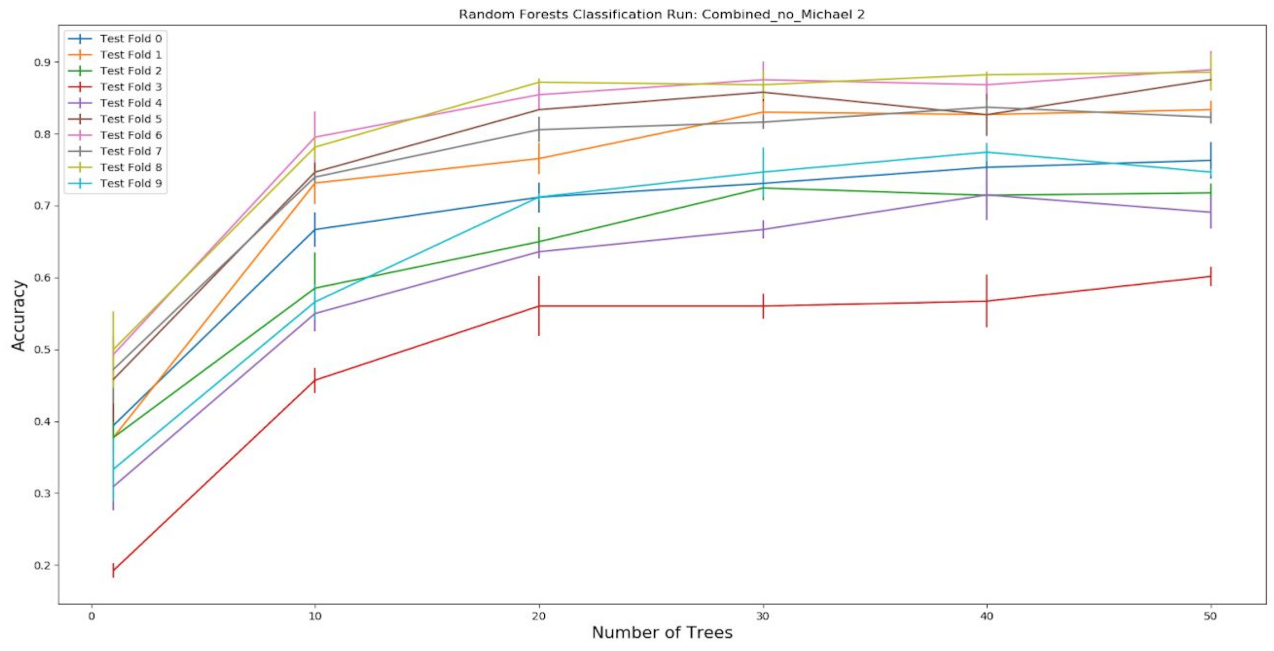


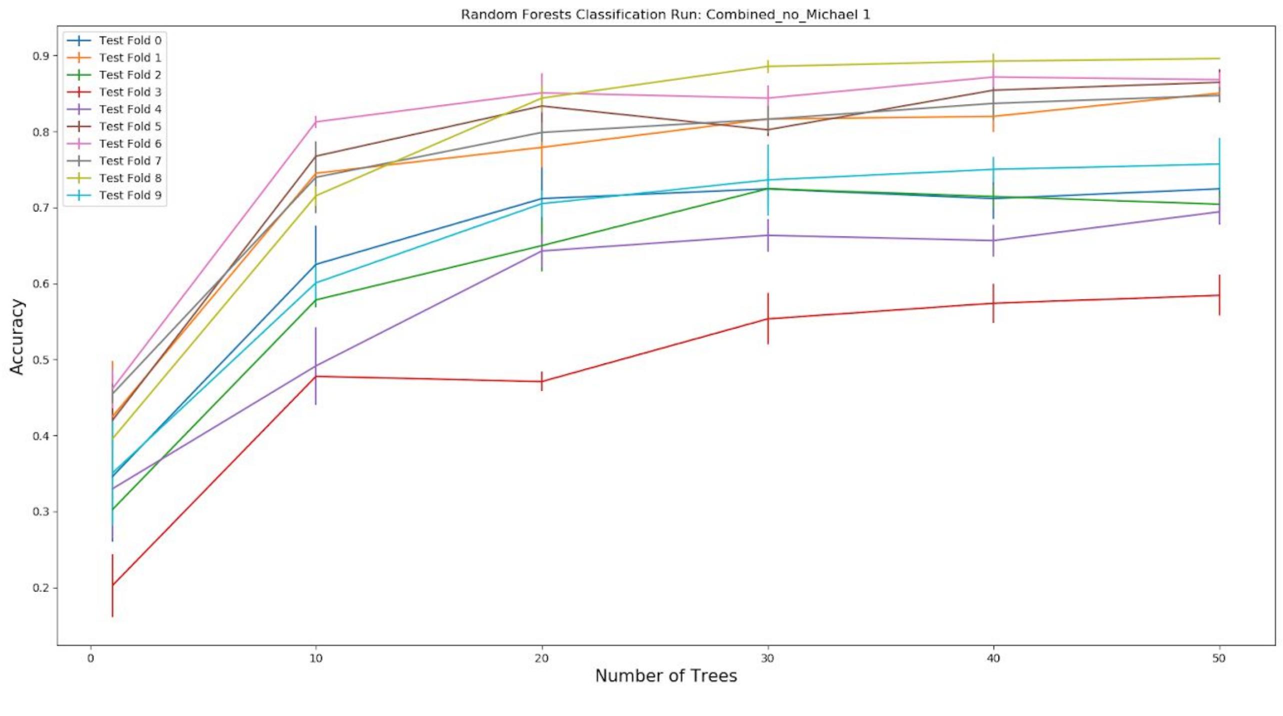


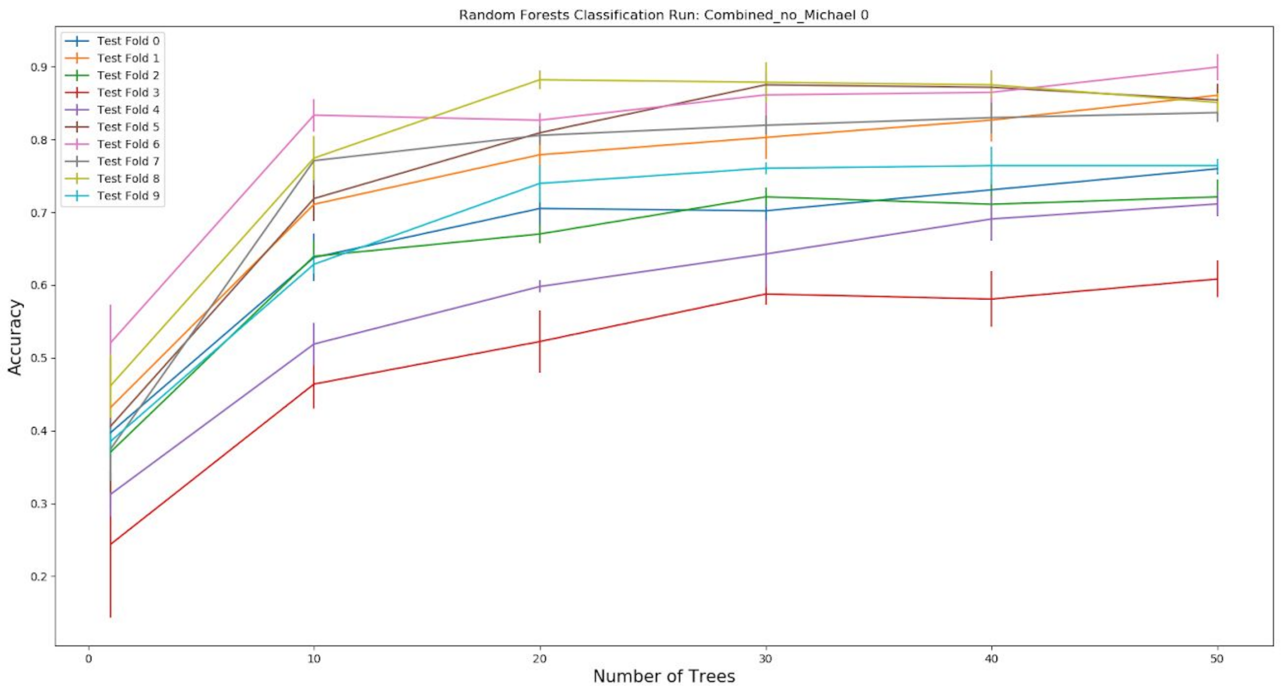


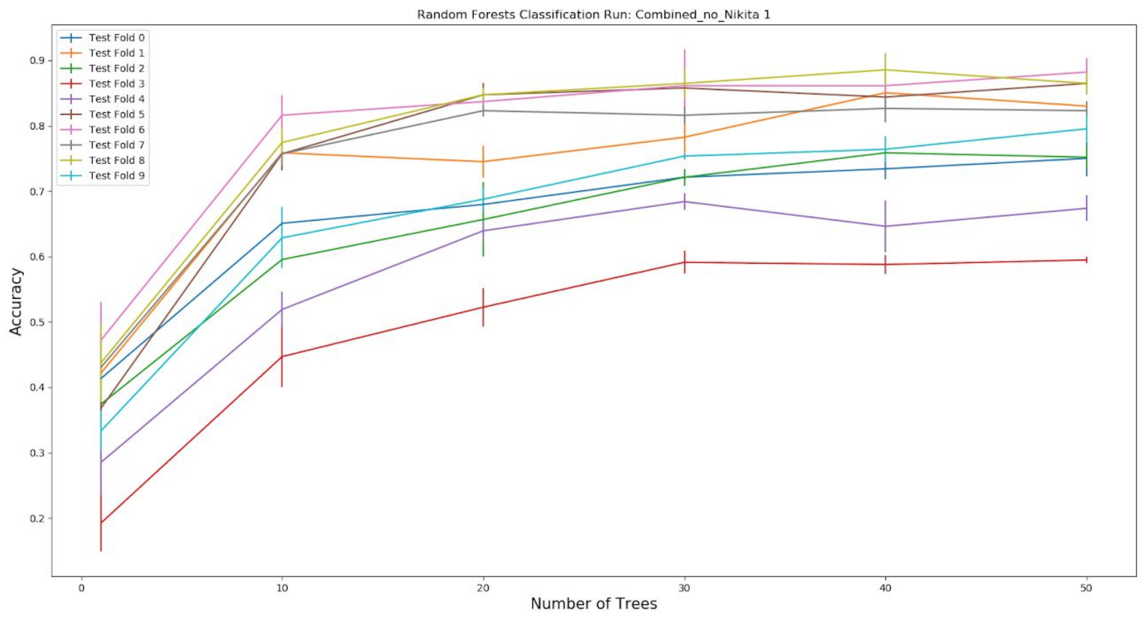
We trained our data using 960 total images, and we tried to vary our results by including other groups images. This was important in our preprocessing stages. In preprocessing, we found that some of our accuracy scores suffered. We believe that this was because these algorithms are trained to work on larger data and datasets. The original images have a lot more detail than our preprocessed images, so the resulting scores suffered. These algorithms also worked better on the images prior to preprocessing because they learned to understand patterns in the hand shapes and ignore color because we each had a different skin tone.

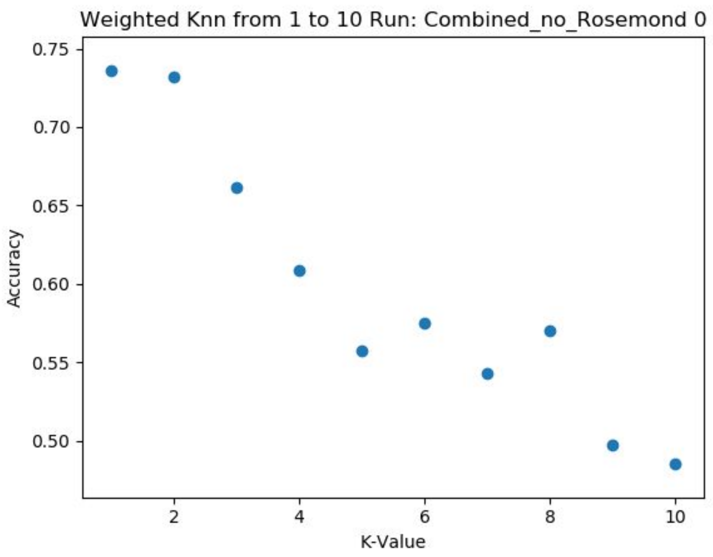
Random forests worked better for our dataset and features than the KNN. KNN score accuracy linearly decreased as we increased the amount of nearest neighbors we were looking at. KNN worked most effectively with 1 or 2 closest neighbors which is most likely not a good sign, and something we probably want to look into.

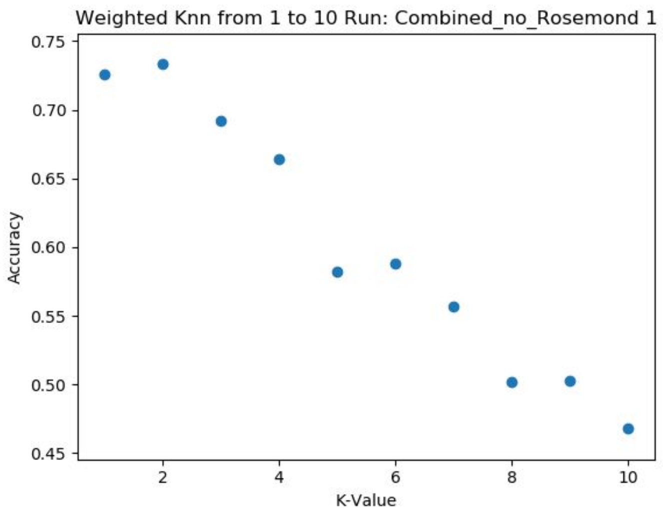

Random Forests Classification Run: Combined_no_Michael 2
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50

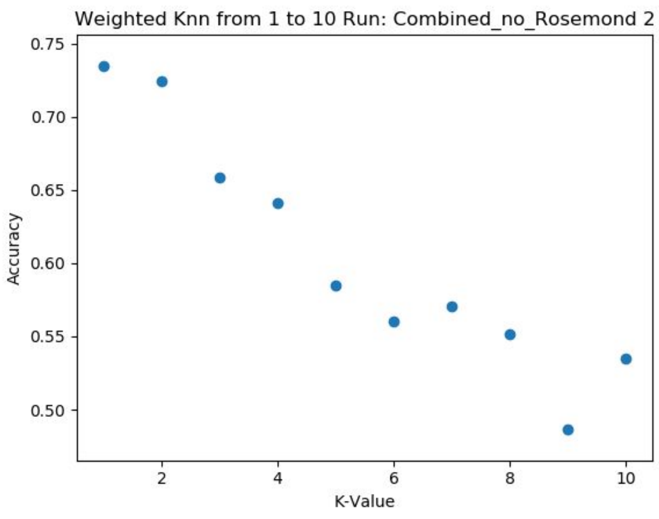

Random Forests Classification Run: Combined_no_Michael 1
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50

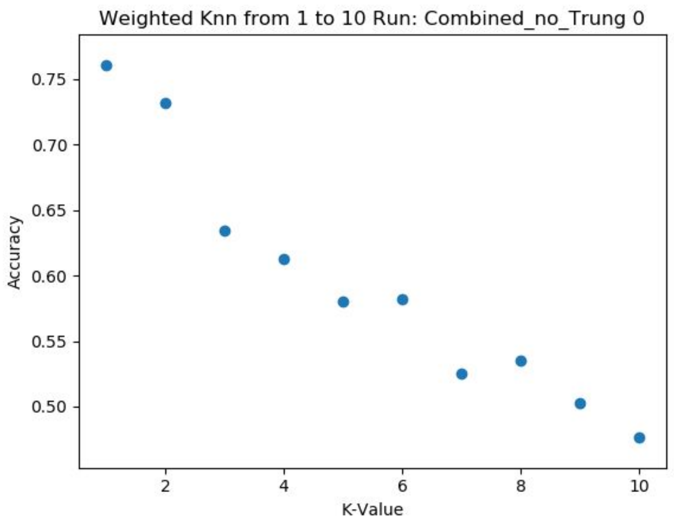

Random Forests Classification Run: Combined_no_Michael 0
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees

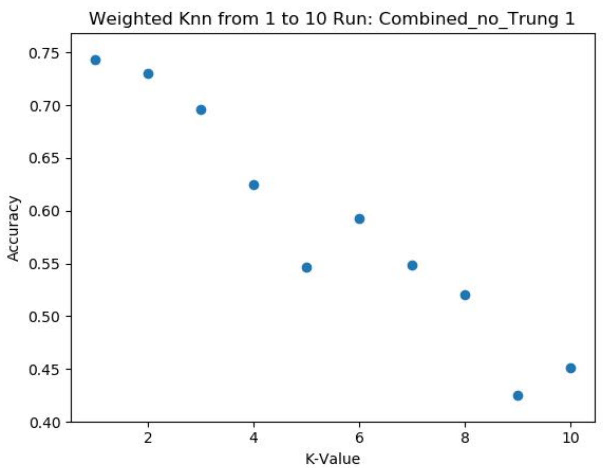

Random Forests Classification Run: Combined_no_Nikita 1
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees

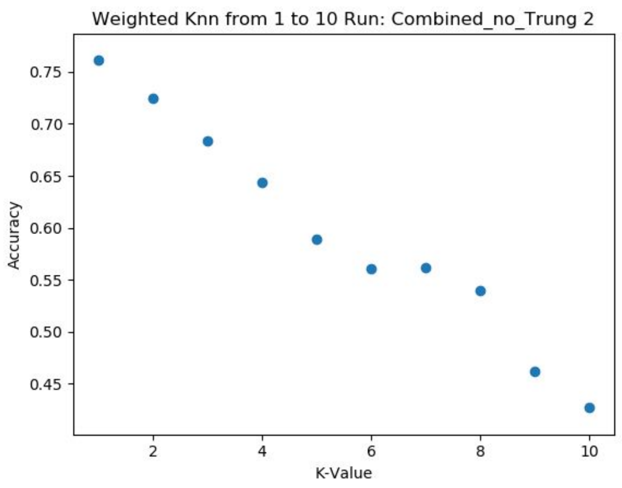

Weighted Knn from 1 to 10 Run: Combined_no_Rosemond 0
Accuracy
K-Value

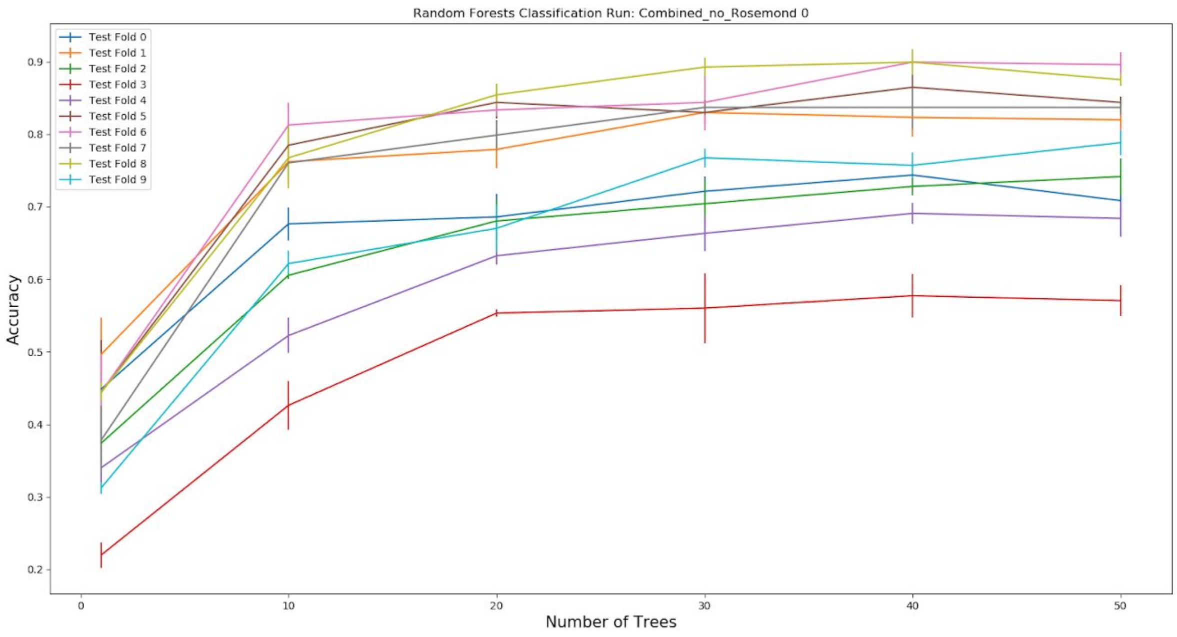

Weighted Knn from 1 to 10 Run: Combined_no_Rosemond 1
Accuracy
K-Value


Weighted Knn from 1 to 10 Run: Combined_no_Rosemond 2
Accuracy
K-Value

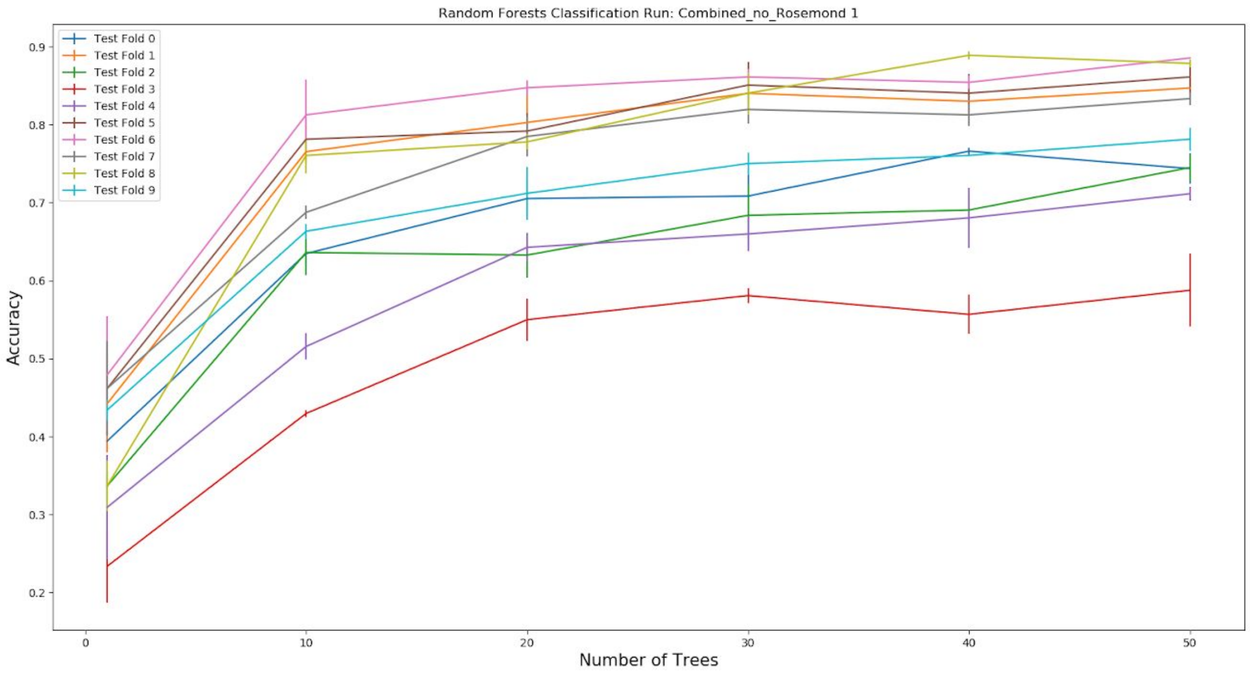

Weighted Knn from 1 to 10 Run: Combined_no_Trung 0
Accuracy
K-Value


Weighted Knn from 1 to 10 Run: Combined_no_Trung 1
Accuracy
K-Value

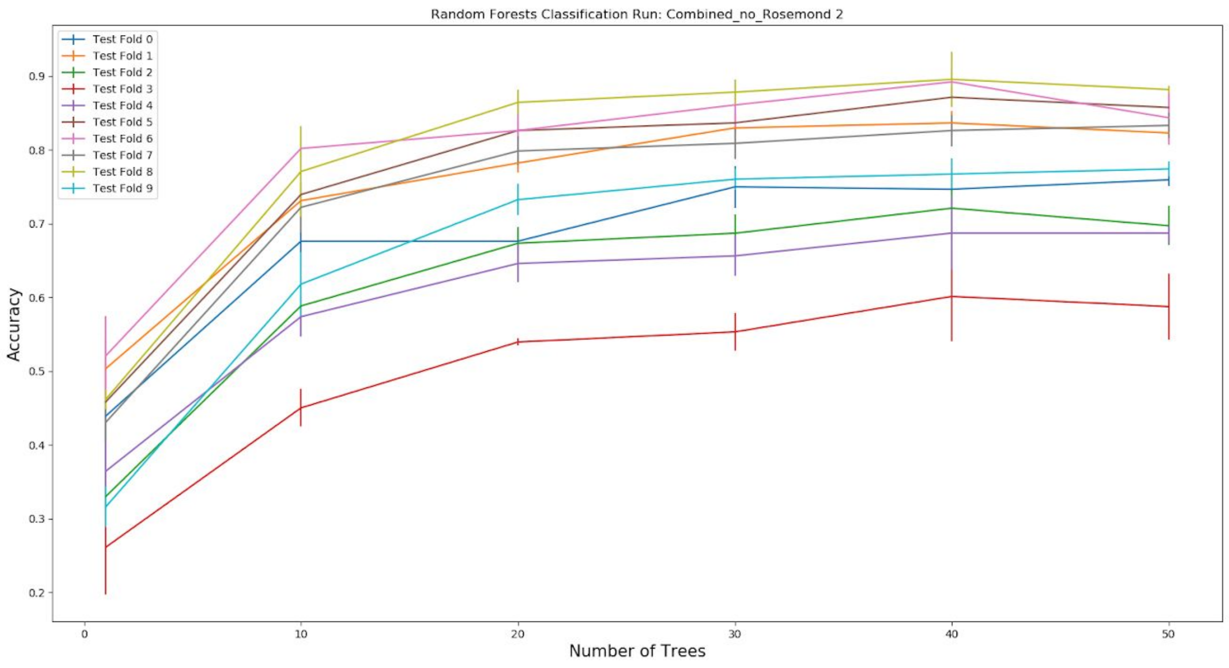

Weighted Knn from 1 to 10 Run: Combined_no_Trung 2
Accuracy
K-Value

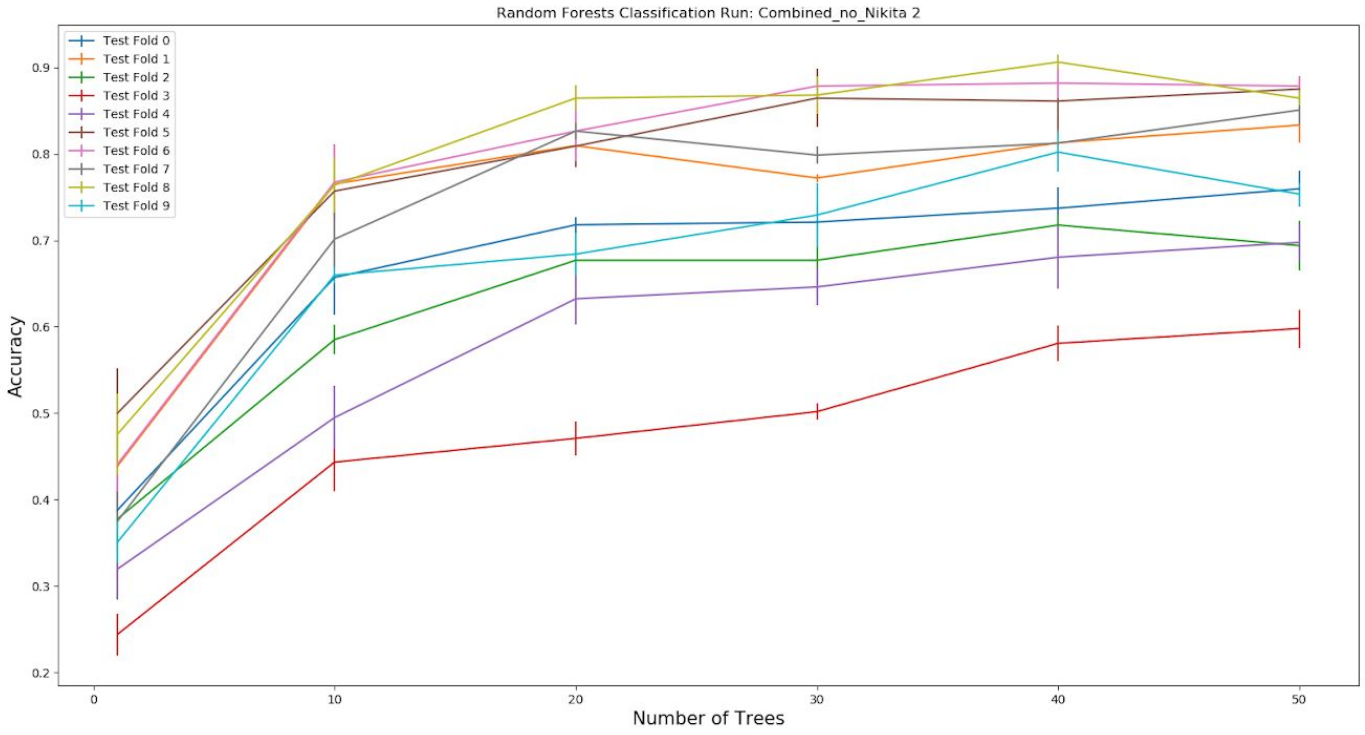


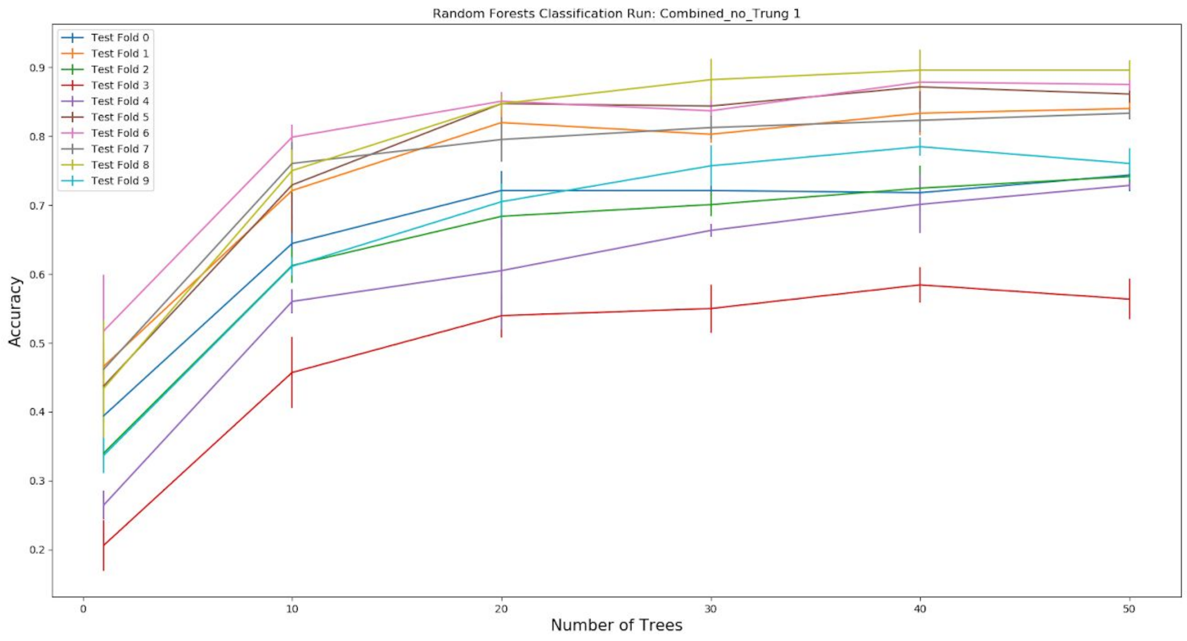
Random Forests Classification Run: Combined_no_Rosemond 0
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees

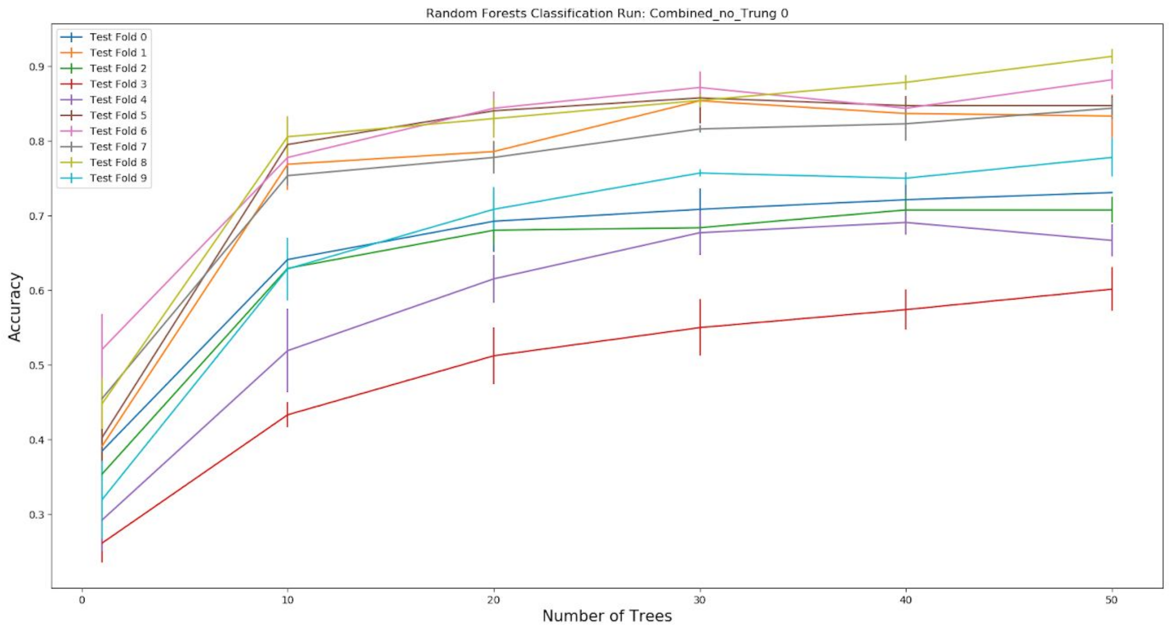


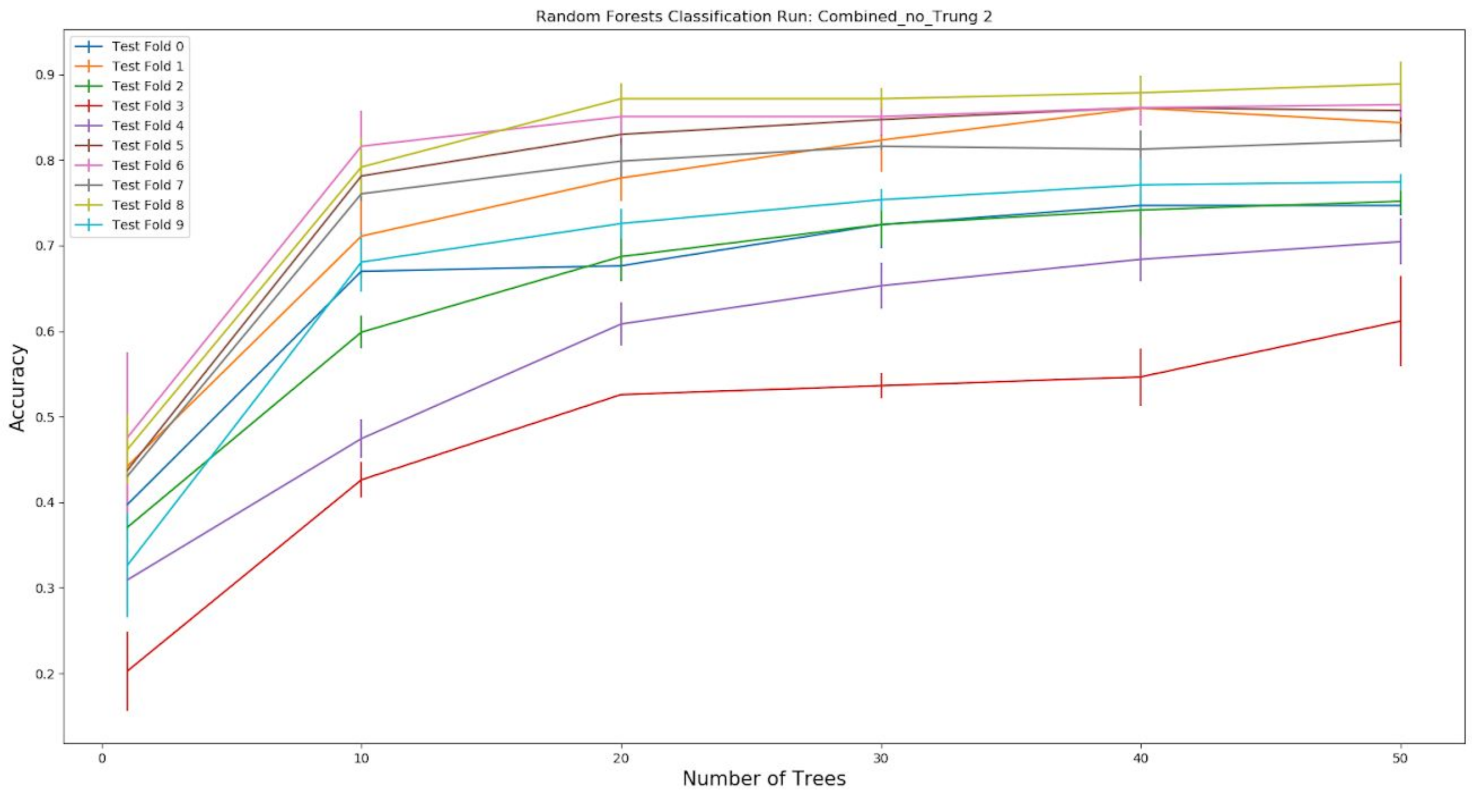
Random Forests Classification Run: Combined_no_Rosemond 1
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees


Random Forests Classification Run: Combined_no_Rosemond 2
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees


Random Forests Classification Run: Combined_no_Nikita 2
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees


Random Forests Classification Run: Combined_no_Trung 1
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50

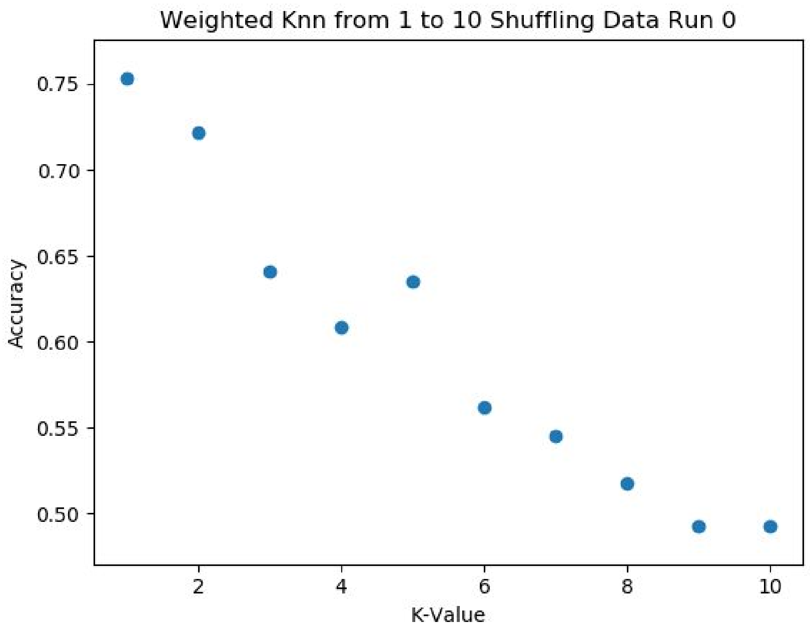

Random Forests Classification Run: Combined_no_Trung 0
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50

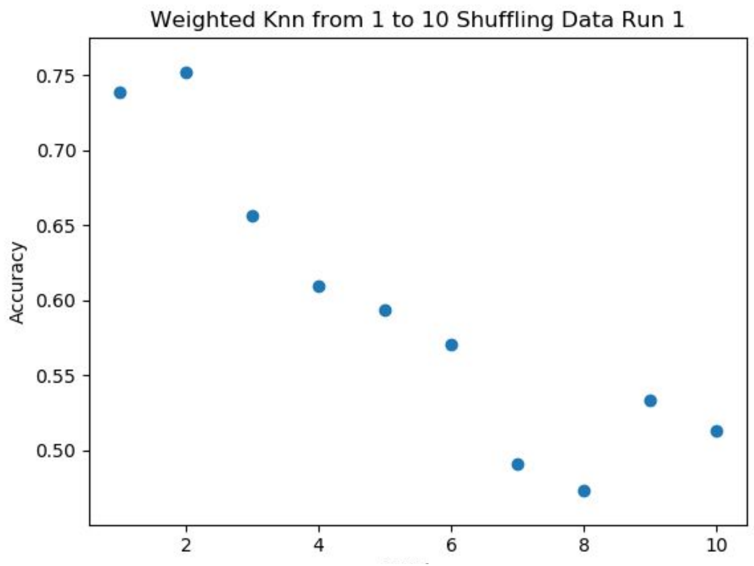

Random Forests Classification Run: Combined_no_Trung 2
Test Fold 0
Test Fold 1
Test Fold 2
Test Fold 3
Test Fold 4
Test Fold 5
Test Fold 6
Test Fold 7
Test Fold 8
Test Fold 9
Accuracy
Number of Trees
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50

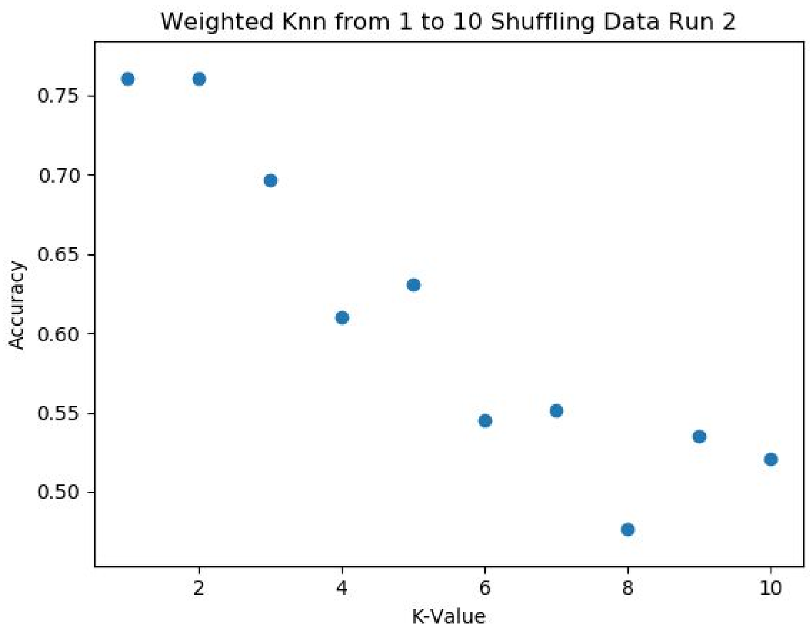
2.) How good is your performance when you shuffle all teammembers' data and use 70% for training (and/or validation) and the remaining 30% for test? Repeat this multiple times (at least 3x). Back up your response with graphs and/or tables of results.

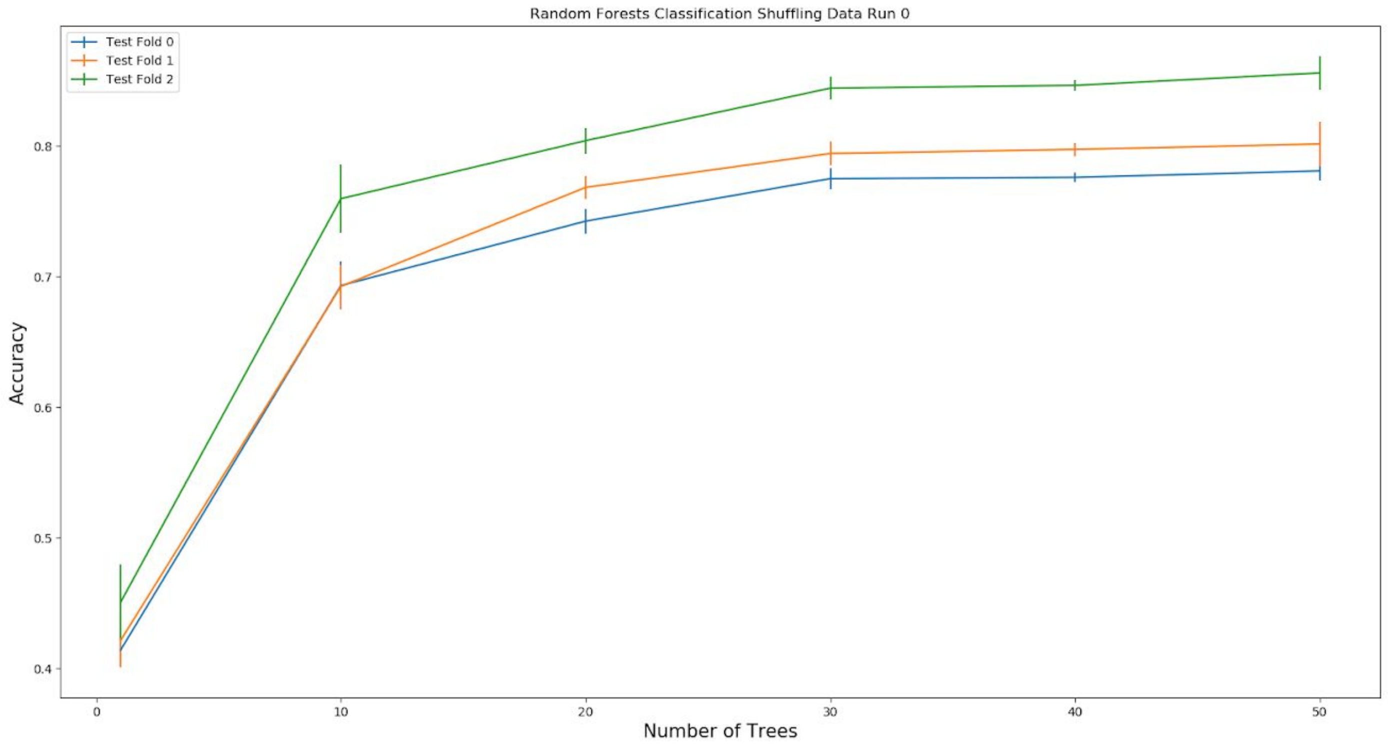
We see that with our KNN we get an accuracy of 70-75%, and with Random Forests roughly 80%. For our K-Nearest Neighbors as we increase K our accuracy drastically decreases. The best K value for us was 1 or 2. The optimal number of trees in our random forest was about 30-40. When we shuffle all teammates data and do a 70-30 split we see that our accuracy stays about the same, and still linearly decreases with increases in K for KNN.

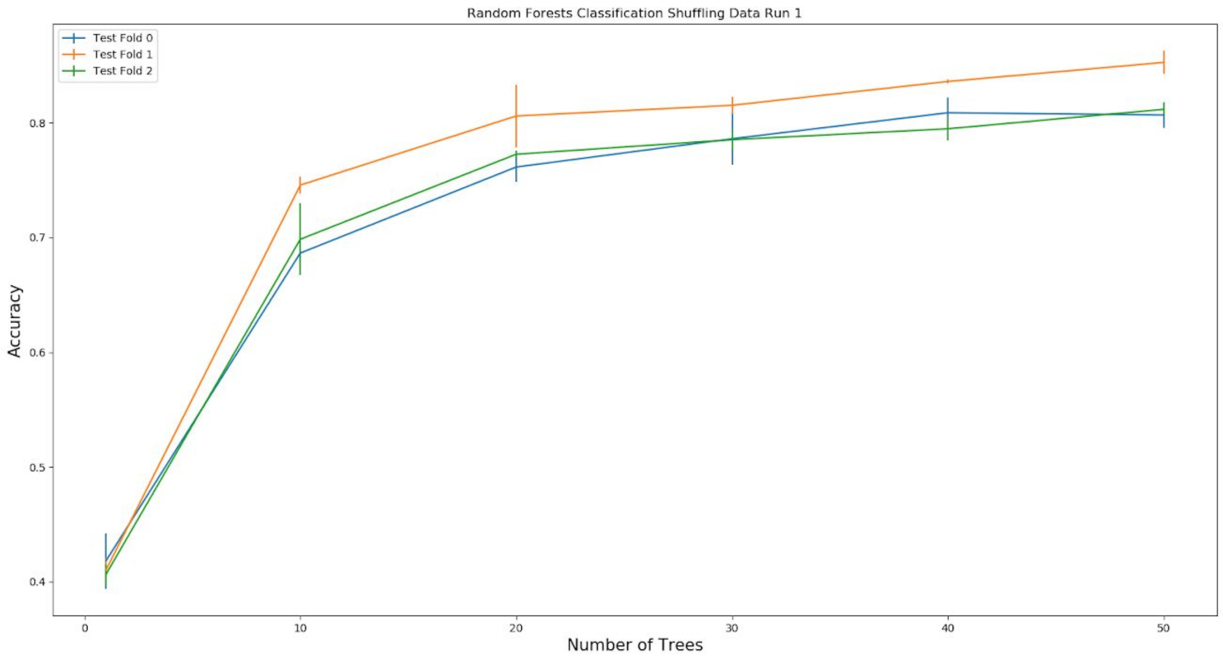
For the Random Forests, scores improved with lower number of trees, but still peaked around 90%. Below we have all of the results from the 3 runs.

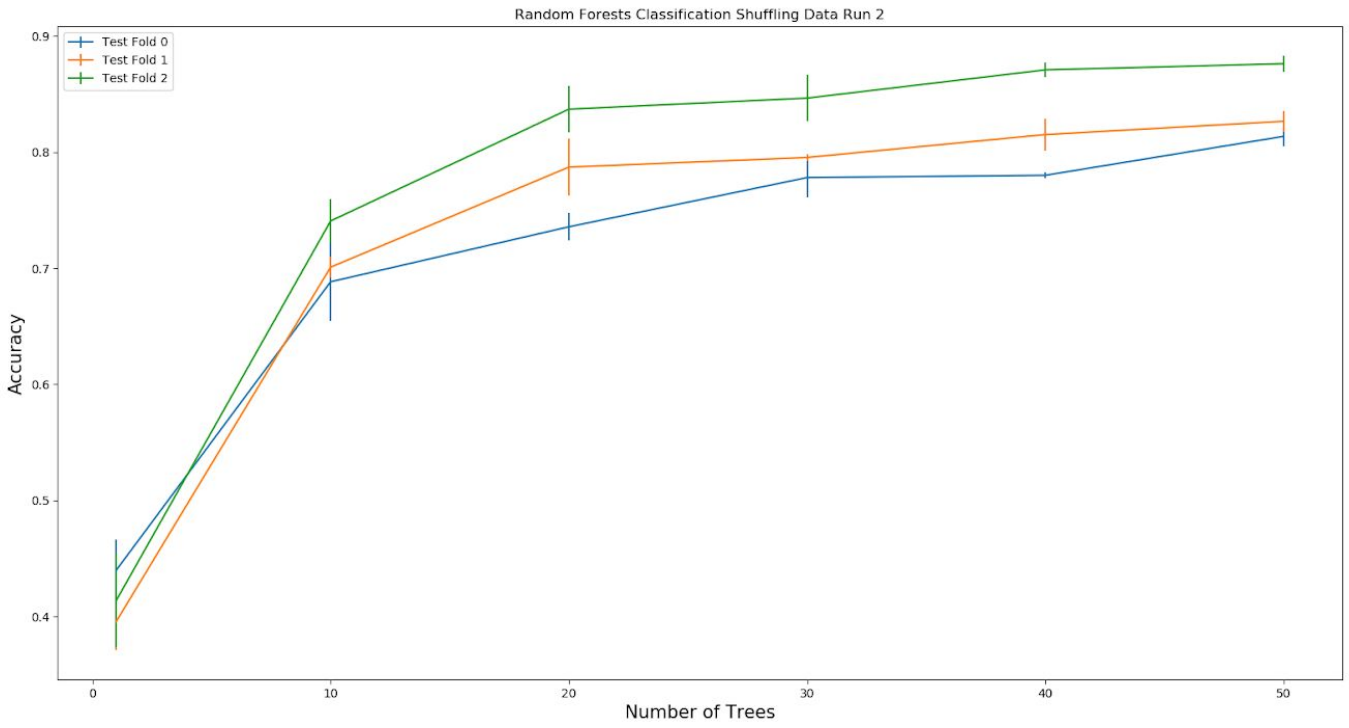





Weighted Knn from 1 to 10 Shuffling Data Run 2
Accuracy
K-Value
0.75
0.70
0.65
0.60
0.55
0.50
2
4
6
8
10


Random Forests Classification Shuffling Data Run 0
Test Fold 0
Test Fold 1
Test Fold 2
Accuracy
Number of Trees
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50


Random Forests Classification Shuffling Data Run 1
Test Fold 0
Test Fold 1
Test Fold 2
Accuracy
0.4
0.5
0.6
0.7
0.8
Number of Trees
0
10
20
30
40
50


Random Forests Classification Shuffling Data Run 2
Test Fold 0
Test Fold 1
Test Fold 2
Accuracy
0.4
0.5
0.6
0.7
0.8
0.9
Number of Trees
0
10
20
30
40
50

3.) Are there any issues with your preprocessing or feature extraction? What are they? Why are they issues? What is your plan for future development in project?

We've had several issues preprocessing our images. The first thing we noticed in our dataset vs other datasets is that our images are clear and have little noise. In other images people left in their shoulders, arms, and various other background noise. We decided to create some preprocessing algorithms which try to find the hand and crop around it, this worked effectively for hand symbols which were condensed, but for letters like 'L' or 'G' part of the finger strays away from the condensed mass so we lost a part of our fingers in preprocessing. It was difficult to find the correct libraries which had things we could use for this project. We haven't previously done any work with feature extraction or preprocessing, and most of the articles and research papers we found were already advanced enough that they didn't really need to explain what they were doing.

Another difficulty with preprocessing was finding a data structure that we could use that was compatible with the libraries that we found. We settled with a 3d matrix, but encountered some issues flattening it down for other libraries.

We looked at several feature extraction algorithms, mainly "SURF", "SIFT", "ORB", edge extraction, and others. Surf and Sift became patented so they were removed from the opencv library, but were replaced by a new and improved "ORB". We found several orb implementations, namely on opencv and scikit-learn image. We ran into an issue with the opencv version because there was a lack of documentation regarding how the images should be loaded into Orb. With the opencv version, we found that images had to be grayscale. The opencv version also required images to vary in contrast between pixels. We will continue looking at the opencv version of ORB as well as other feature extraction techniques that we haven't seen yet.

For the future we plan on exploring more preprocessing and feature extraction algorithms and how we could modify them to work for our specific project. There has already been a lot of research done on hand symbol recognition, but that uses video feed and background subtraction strategies to get the hand. We also will look into using Convolutional Neural Nets once we begin using PyTorch and learn more about them. It seems that CNN's are the modern way of dealing with problems like this, and since PyTorch has much more support than the libraries we were looking at, there might be some new possibilities for tackling this problem.