SPS report

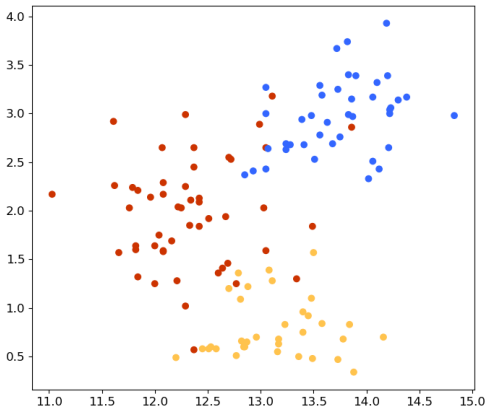
Theano Xirouchaki (tx17079) and Jonathan Hall (jh17979)

May 1, 2019

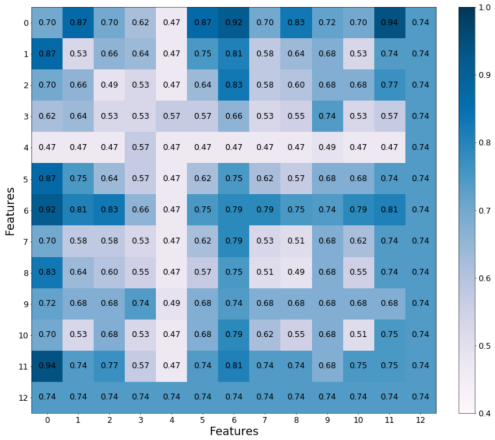
1 Feature selection

There were a number of factors to consider for feature selection. Firstly, we looked at the accuracy of each feature combination if it were to be used for a centroid classifier. This, although not perfect, gave us a rough idea of what features would perform well in a KNN classifier. Moreover, we calculated the mutual information of each feature with the labels, and noticed that a fair number of the features giving high accuracy with centroids also had high mutual information with the labels. Finally, we looked at the scatter plots of some promising combinations and decided on the first and seventh features, as they have high Centroid accuracy, fairly high mutual information with the labels, and they seem to give a pretty well split scatter plot.

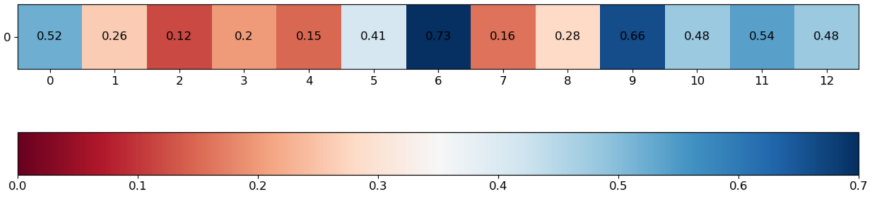
(a) Scatter plot of first and seventh features



(b) Accuracy of centroids with different feature combinations



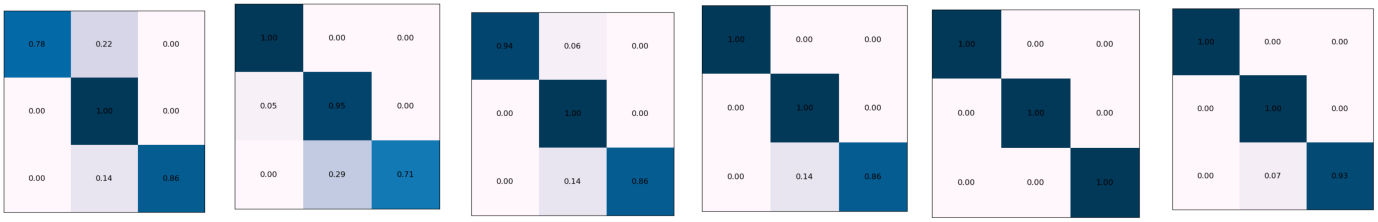
(c) Mutual information of features against labels



2 KNN

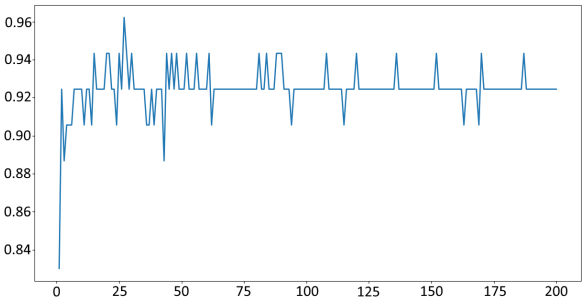
After implementing KNN, we compared our results to the sklearn KNeighborsClassifier results, to make sure our code was correct. We noticed that we got the same accuracy for all values of k except k=2. We made an educated guess that it might be our tie-breaking method being slightly different to sklearn's, so we looked at its documentation and changed from decreasing k by 1 to resolve ties, to using the first modal value. This resulted in getting the same accuracy as the python classifier. We tried standardising the data but the results were hardly different, so we left standardisation out for the basic KNN. From the confusion matrices we can see that the classifier struggles with the first class until k=4, hardly struggles with the second class at all, and struggles with the third class almost throughout.

(a) Confusion matrix for k=1 (b) Confusion matrix for k=2 (c) Confusion matrix for k=3 (d) Confusion matrix for k=4 (e) Confusion matrix for k=5,7 (f) Confusion matrix for k=6

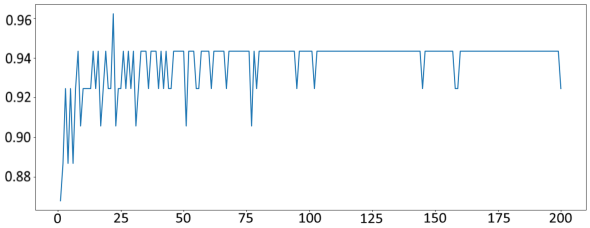


3 Alternative Classifier

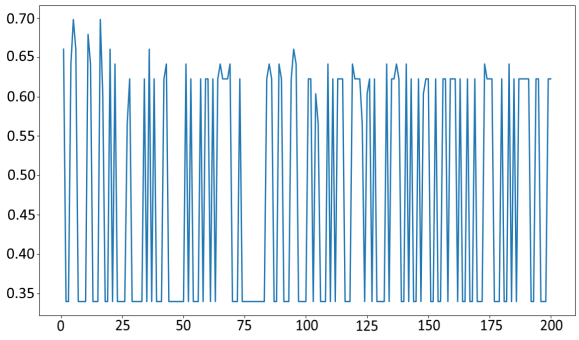
For the alternative classifier, we decided to avoid Naive Bayes as its advantages seem to lie mostly in providing results in real time, which is not something we need to consider in this case. Moreover, it works best when the features are independent, which is not the case for the features we have selected: their mutual information is 0.40236 and their Pearson coefficient is 0.31709, both of which are fairly high. Hence, we decided to implement decision trees, which led to an accuracy of 88.68%. Then, we got curious and decided to implement feature bagging and experiment with generating multiple decision trees and having them vote to make a prediction.
We further experimented with turning our classifier into a random forest by randomly selecting which feature to use at each split, but since we were using only two features, that didn't prove very successful. As such, the version we submitted only includes bagging. We found standardisation to have no effect on the decision tree, as expected, as long as we standardise the test data with the same mean and variance as the train data. Below you can see the variation in accuracy when using bagging vs random forest combined with entropy and gini when running 1 to 200 trees. The time increased linearly other than some minor (random) spikes due to CPU use. Our classifier does not quite reach 100% accuracy, like KNN does, and definitely takes longer to run, but we do see a benefit to it as it would react better to features with large scale difference. Our final version uses bagging, as random forest is too inconsistent, and gini, since it seems marginally better than entropy.



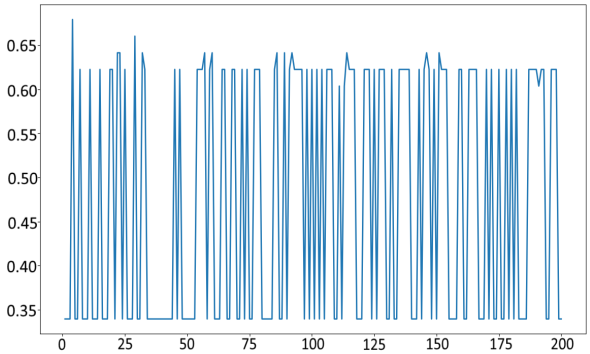
(a) Accuracy for bagging using entropy



(b) Accuracy with bagging using gini



(c) Accuracy with random forest using entropy



(d) Accuracy with random forest using gini

Figure 3: Accuracy against tree number

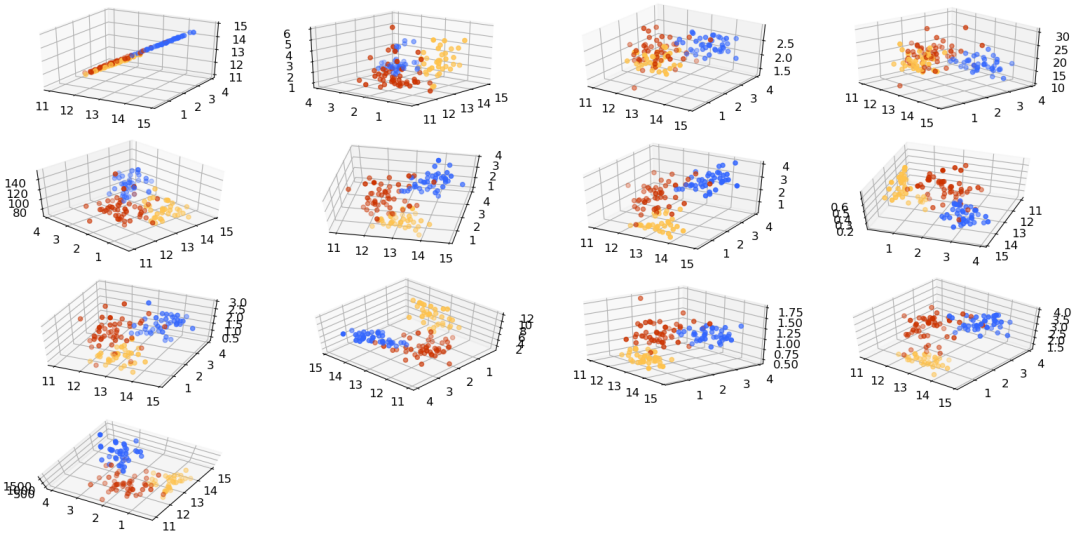
After doing some research on ways to further improve our system, we found out about fuzzy decision trees [1, 2] which should work better for continuous attributes than traditional decision trees, but we concluded that it was outside the

scope of the coursework to implement something so advanced.

4 KNN – 3 features

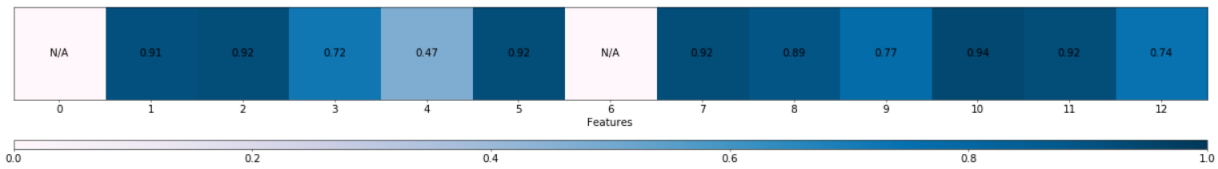
To select a third feature, we generated 3D scatter plots of the the two chosen features and each other candidate feature.

Figure 4: First and seventh features plotted against all other candidate features



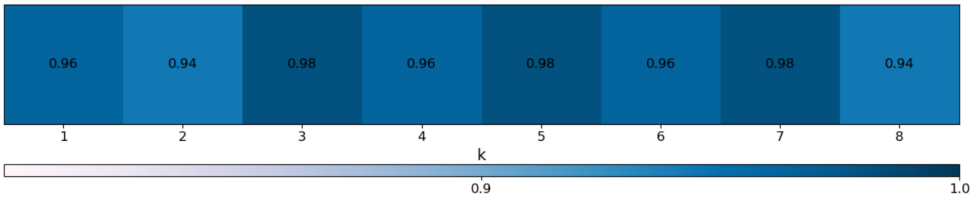
This allowed us to evaluate the features based on their separation. Judging from a three dimensional inspection of the scatter plot, we deemed the tenth feature to have the greatest separation as shown here. It was also shown to have above average mutual information with the labels. (Fig 1c.) We also looked at the centroid accuracy this feature would result in. (Fig 5.)

Figure 5: Centroid accuracy for different possible third features



After running our code with and without standardisation we noticed that, unlike the 2d KNN, standardisation had a significant positive effect on our accuracy. We understand that to be because as the number of dimensions increases, the system becomes more sensitive to scaling issues. We are standardising by fitting the model to the training data, and using that model to standardise both the training set and test set. Although this does involve the assumption that the test data is derived from the same normal distribution as the training data, we feel that is reasonable to assume.

Figure 6: KNN 3D accuracy for various values k, after standardisation



5 KNN PCA

For KNN PCA we were originally getting quite low accuracy until we introduced standardisation. This makes sense as the features are originally on much different scales, for example the thirteenth feature is in the hundreds and thousands, while the eighth feature has no values greater than 1. As such, the PCA algorithm would consider some features much more than others due to the variance being off because of scale. Standardisation solves that issue. We noticed that the results for PCA are more consistent regardless of k than for 2D KNN, but are a bit lower than 3D KNN. Moreover, the confusion matrices show PCA is only struggling with class 2, while 3D KNN only struggles with class 3.

Figure 7: KNN PCA accuracy for various values k, unstandardised

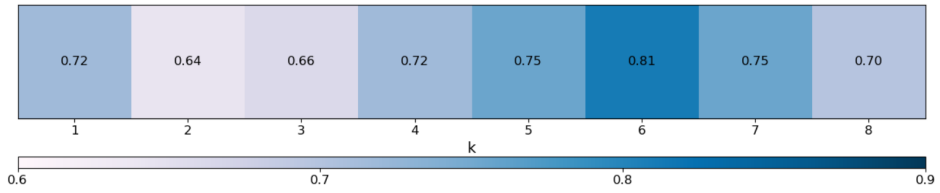
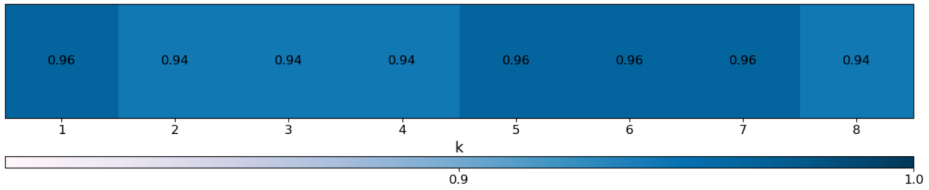
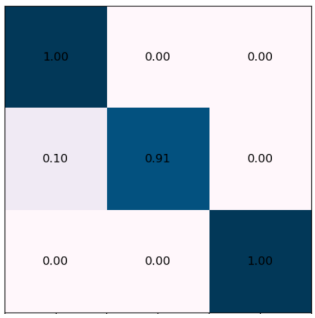


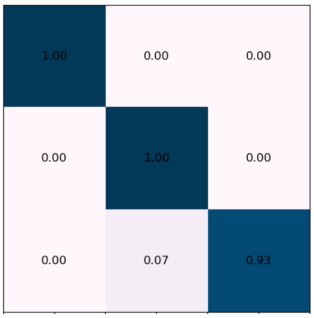
Figure 8: KNN PCA accuracy for various values k, standardised



(a) Confusion matrix for KNN PCA k=5



(b) Confusion matrix for 3D KNN with k=5



References

[1] Fuzzy Decision Trees, Catalin Pol, Url:https://docplayer.net/29278995-Fuzzy-decision-trees.html

[2] A complete fuzzy decision tree technique, Cristina Olaru, Louis Wehenkel, Fuzzy Sets and Systems, Volume 138, Issue 2, 2003, Pages 221-254.