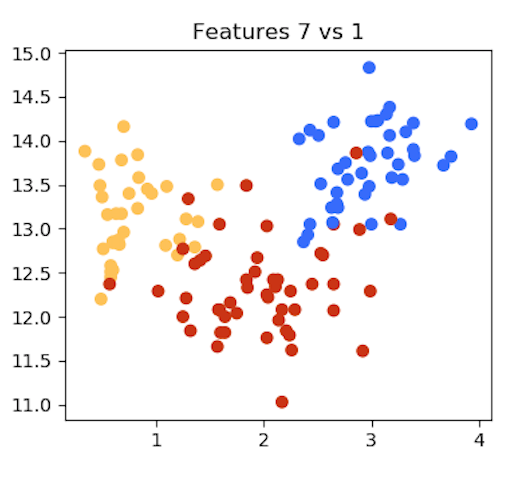
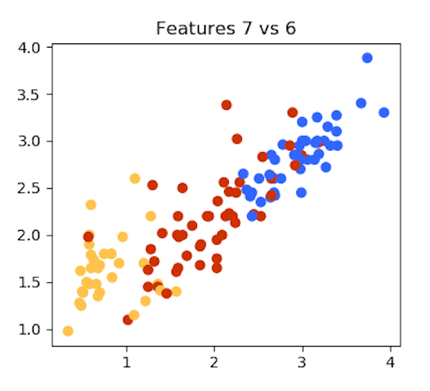
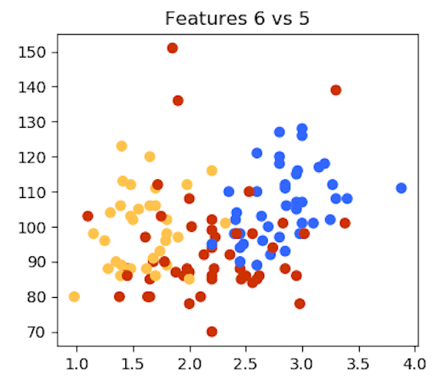
**SPS CW2 Report**

Feature Selection

When selecting the pair of features that we would use to train our classifier, there were a few features we were looking for.

Firstly we looked for feature pairs that gave distinct clusters of each group. Primarily this was focused on ensuring classes 1 (Blue) and 3 (Yellow) were linearly separable, as looking at the scatter plots it was quite apparent that class 2 (Red) didn’t have much linear separation from the other two classes regardless of which feature pairs were selected.

Secondly, as can be seen in the plot of our selected features, there is no strong correlation between them, while clusters are still present. This means that one feature cannot be simply predicted using the other, resulting in our classifier having more information to work with.

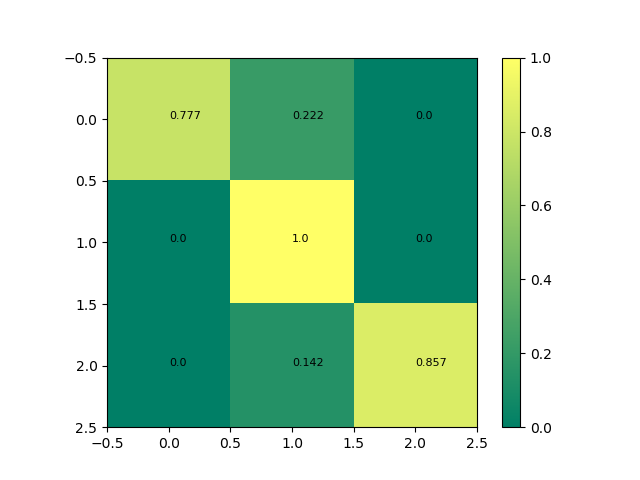
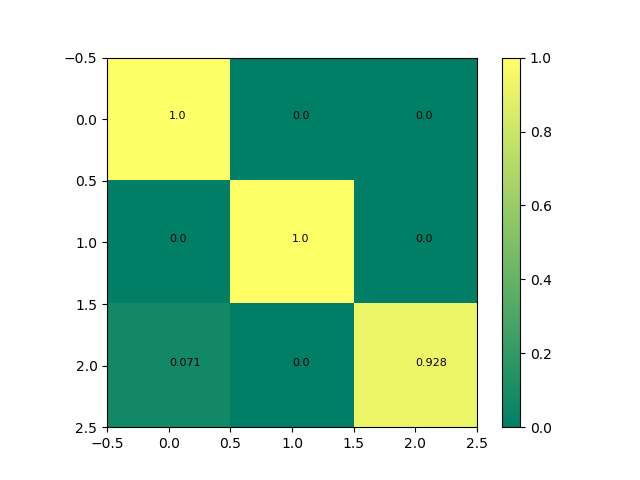


Shown to the side, features 7 and 6 were passed over in favour of features 7 and 1 due to both pairs having a similar degree of clustering and separated groups, but with 7 and 1 being less clearly correlated with each other. On the other hand pairs such as features 6 and 5 weren’t considered as, while they do also show a degree of clustering, the lack of any clear separation between classes would result in any classifier we created being inaccurate for identifying classes 1 and 3, and almost useless for identifying class 2.

KNN Classifier

Using the features we selected, our implementation of the KNN classifier has accuracies that range from 88% to 100% and appear to generally increase as *k* grows, with the exception of a small drop in accuracy when going from *k* = 5 to *k* = 6. This drop in accuracy could be for a variety of reasons, but is most likely due to an edge case from any of the classes that perhaps didn’t have a single modal neighbour class for *k* = 5 having its extra nearest neighbour be from the incorrect class.

The trend across *k* ∈ {1, 2, 3, 4, 5, 7} would seem to indicate that with the feature pair we have selected, that *k*s closer to 7 provide the best performance. This is probably due to the fact that while we have looked for features with distinct clusters, we could not avoid each class having a small number of outliers. This would result in small *k*s occasionally giving the wrong results for test values in cases where the data point happens to be closer to an outlier from a different class. Larger *k*s would avoid this as it would have a wider range of points to query, with the outlier being ignored in favour of the majority from the clusters class.

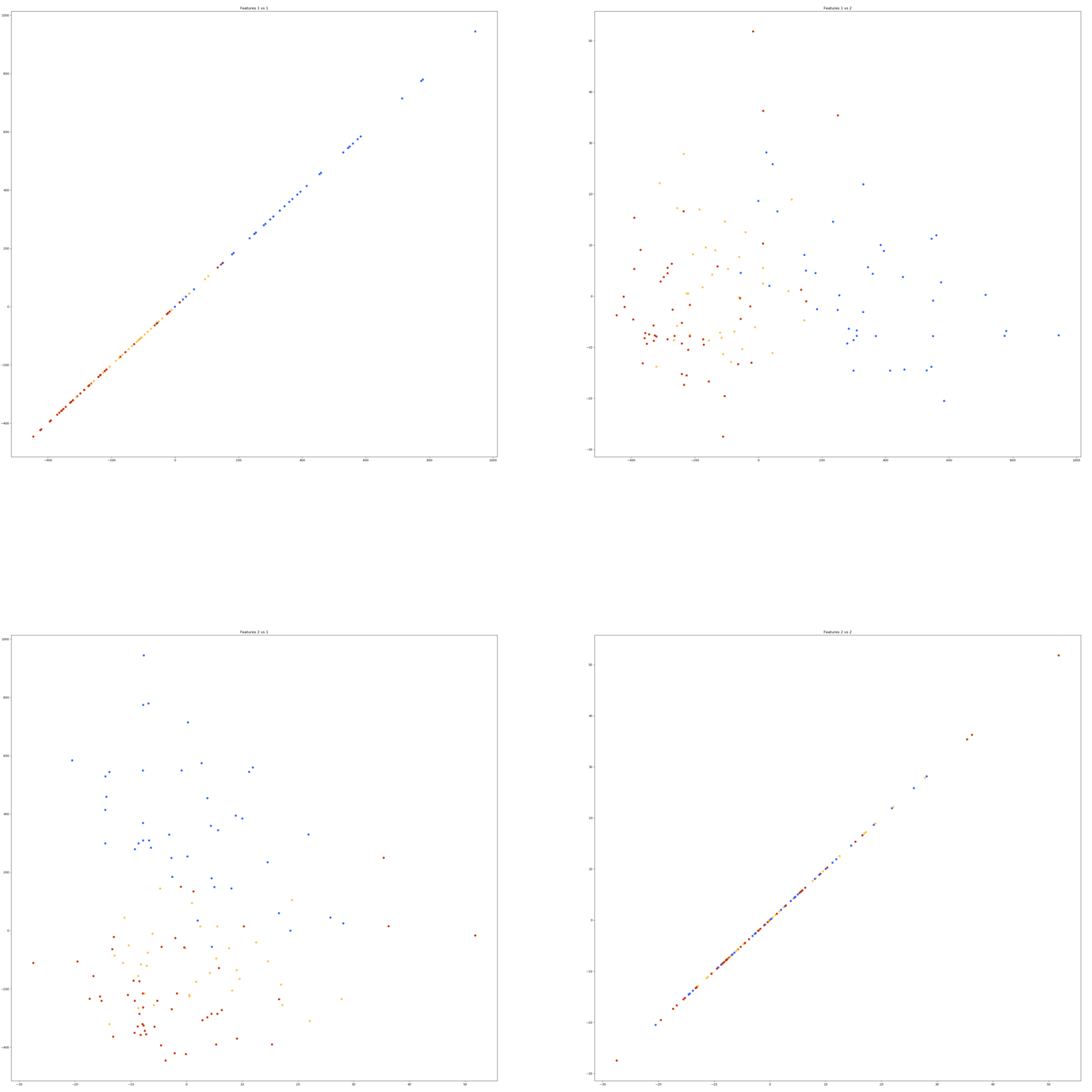


Another point of interest can be seen in how the confusion matrix develops as *k* grows.

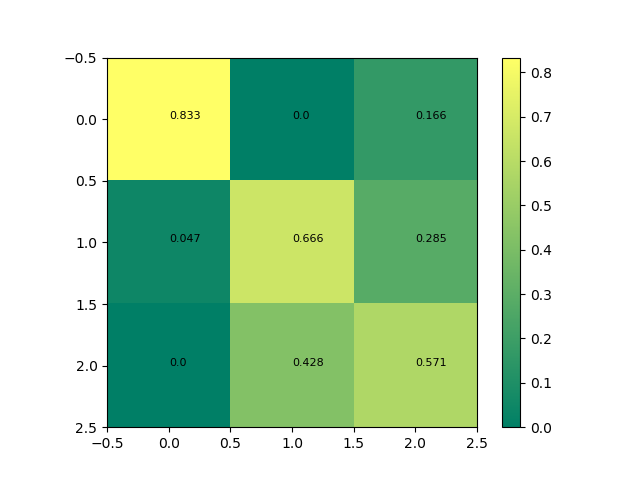
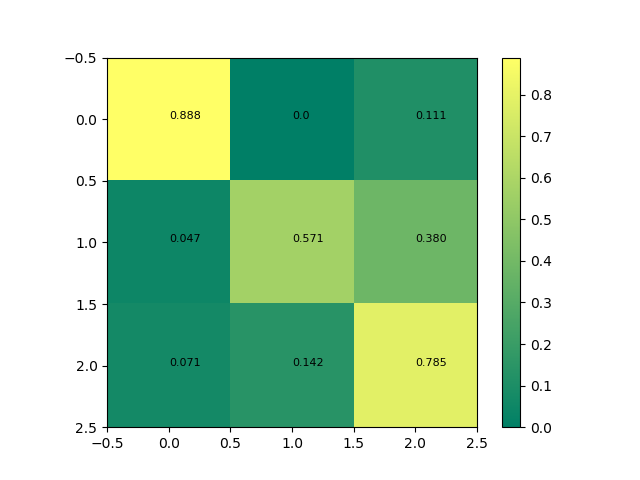
For *k* = 1, the only errors to occur are between class 2 and the other classes, which can be attributed to the feature pair we selected causing class 2 to have some spread into the class 1 and class 3 clusters. This changes when *k* = 6 as the wider range of nearest neighbours being checked allows for some test values to be mistake for class 1 when they belong to class 3. This also agrees with what was stated above, as the possible reason for why the accuracy of the classifier decreases from *k* = 5 to *k* = 6.

k = 6

k = 1

KNN PCA Classifier

Shown to the side is a scatter plot of the features from our training data after being orthogonally projected onto a 2-D space. As can be seen in this plot, classes 2 (Red), and 3 (Yellow) have main clusters, with a few outliers, while class 1 (Blue) doesn’t really have a main cluster of data at all. This is in contrast to our manually selected features which showed strong clustering for all classes. Another major difference between the PCA transformed data and the data obtained from our feature selection is the PCA data’s complete lack of separation between the data clusters for class 2 and 3.

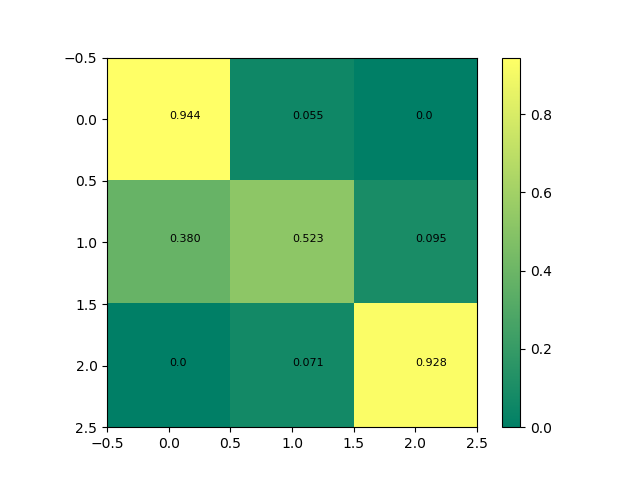


The effect of the data clusters for classes 2 and 3 having almost no separation from each other has had a large effect on the general accuracy of the KNN classifier when provided the transformed training data and test data, having accuracies between 69% and 82% for all *k* ∈ {1, 2, 3, 4, 5, 7} with no clear trend showing an increase or decrease of accuracy as *k* grows. However, by looking at the confusion matrices of our results, the reason behind this loss in accuracy is shown to be primarily due to the classifiers difficulty differentiating between data that belongs to class 2 and class 3. The scatter plot shows this is because of the clusters for classes 2 and 3 not being separated, which means any data from these classes could be surrounded by data from the other. This is why when compared to the results from our classifier using our manually selected feature pair, with decent separation of all three class clusters the accuracy of the classifier trained on the PCA transformed data is worse for all values of *k*. This problem could be avoided in future by implementing Fischer Discriminant Analysis which, instead of maximising component variance, attempts to maximise between class scatterness and preserve class information while still reducing the dimensions of features.

K = 7

K = 3

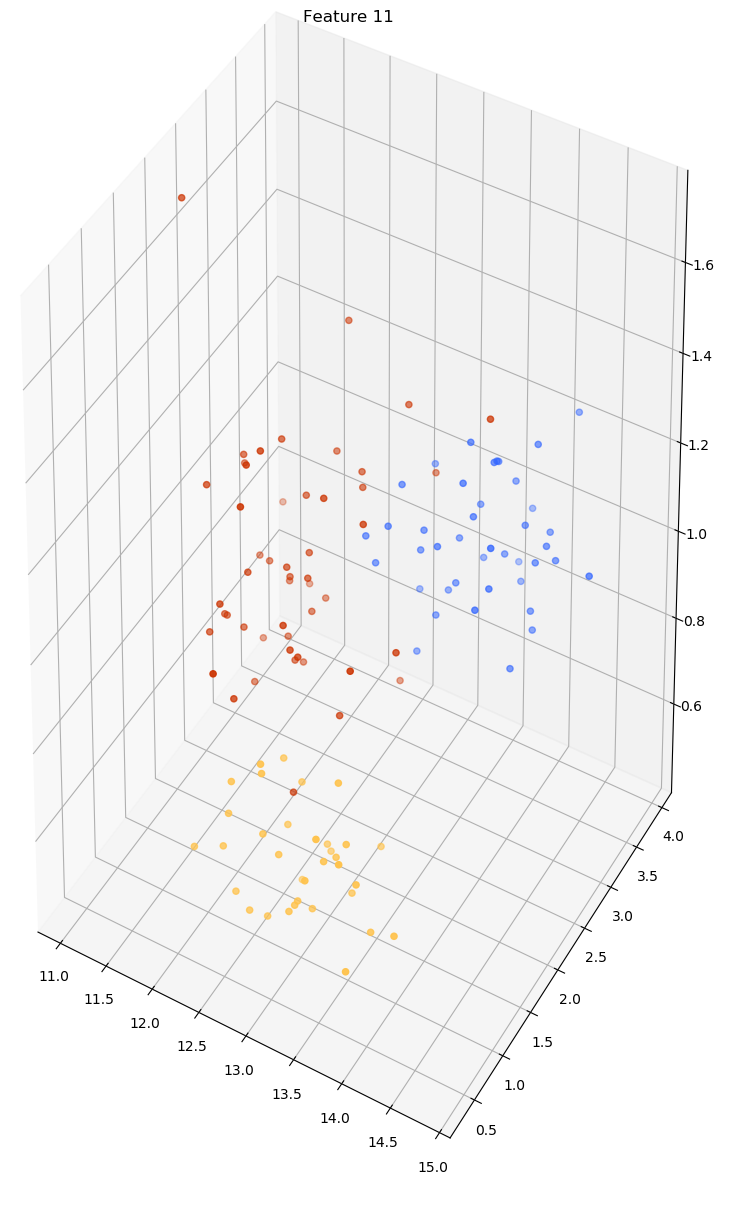
Naïve-Bayes Classifier

Our Naïve-Bayes Classifier had an accuracy of 77% when run on the test with our feature selection, which is less accurate than our KNN classifier for all tested *k*. This is likely due to our features 1 and 7 not being perfectly independent, as we assume for our Naïve-Bayes classifier. Though it is not strong, there is some correlation between the two features, which causes the Naïve-Bayes Classifier to perform worse than it would if the features were truly independent.

Naïve Bayes Confusion Matrix

We can tell from the confusion matrix that our Naïve-Bayes classifier is performing well on classes 1 and 3; better even than a low-*k* run of our KNN classifier. Its issues come from class 2, and are largely due to class 2 data points being mislabelled as class 1. This behaviour is consistent with the scatter plot shown earlier, which depicts some overlap between the two classes. The KNN classifier performs better here, likely because the class 2 data points which are inside the class 1 vector space are themselves clustered very closely. This makes them more likely to be correctly identified by the KNN classifier, as even though they are in the middle of many class 1 data points, their immediate neighbours are other class 2 data points. The same does not hold true for the Naïve-Bayes classifier, as the probability of these outlying data points being class 1 is far higher than their being class 2, due to the greater number of class 1 data points with similar feature values. Thus, the KNN classifier outperforms the Naïve-Bayes when it comes to class 2.

Additionally, the decision boundary of the Naïve Bayes Classifier is less flexible than that of the KNN, which could also contribute to its decreased accuracy. This means that the KNN classifier can better handle the class 2 data points which stray into the space predominantly occupied by other classes. However, it does also make it prone to overfitting, especially on smaller datasets, whereas the Naïve-Bayes Classifier largely avoids this issue.

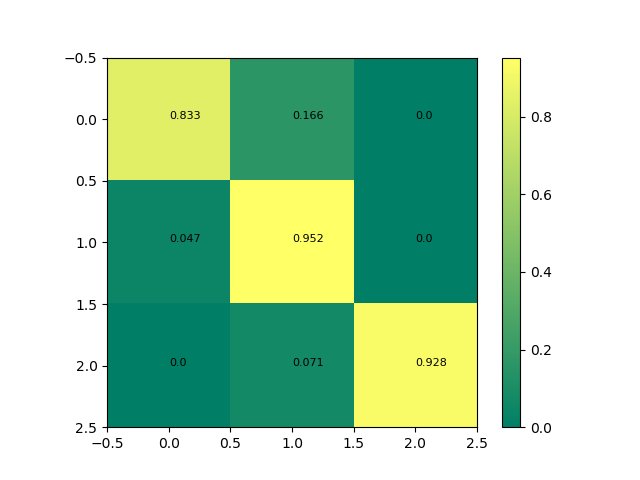


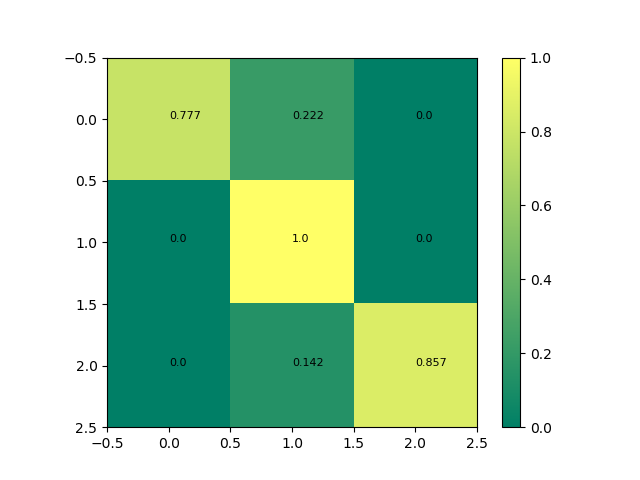
Three features KNN

We selected feature 11 for our third feature, after plotting each of the features against our already selected 1 and 7 on scatter plots. We were again looking for feature combinations that provided distinct clusters of data points within the same class. We particularly focused on choosing a third feature which, when combined with features 1 and 7, would provide a good separation between samples of class 1 and class 2, as the confusion matrices for our two feature KNN classifier showed that the main source of errors was class 1 samples being misidentified as class 2.

Feature 11 plotted against 1 and 7

Additionally, feature 11 was a good selection due to the values it could take being significantly smaller than either of our other two features in the training set, feature 11 ranged from ≈ 0.4 – 1.6, compared to 0.5 – 4.0 for feature 1 and 11 – 15 for feature 7. This is ideal, as feature 11 has the most overlap of the three features, and thus should not affect classification as much as the other two, better suited features. However, this effect could be replicated by implementing distance weighting based on feature value, so it was more a convenience than a necessity when picking a third feature.



Comparing the confusion matrices of our 3-D and 2-D KNN, we can see that adding feature 11 successfully increased the number of true positives when it came to class 1 and class 3. It did however lead to a slight decrease in accuracy with regards to class 2. This is because the new feature combination introduces a significant outlier for class 2, which is surrounded by class 1 samples, as can be seen on the above scatter plot. However, the increase in accuracy for classes 1 and 3 lead to an overall higher accuracy for our 3-D KNN classifier. Additionally, this outlier quickly ceases to be a problem as *k* increases, since all of the samples immediately surrounding it are of class 1.

2-D KNN, k=1

3-D KNN, k=1

Overall, our 3-D classifier provides greater accuracy than our 2-D classifier for all *k* ∈ {1, 2, 3, 4, 5, 7}. This is due to the extra feature making it easier to determine the correct class in cases where the sample would be near the decision boundary in 2-D space.