CW2 - Project Report

Asel Kitulagoda (ak17520) and Karthik Sridhar (ks17226)

April 2019

1 Feature Selection

We began our approach of selecting the best 2-Dimensional features by performing a feature versus feature scatter plot (*Figure 1*). We assigned a different colour class to each label to help us visualise the cluster and spread better. After

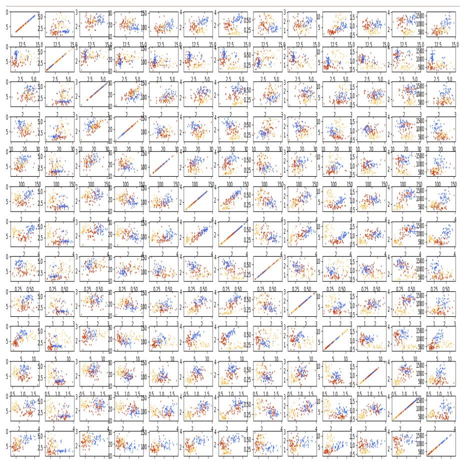


Figure 1: Feature vs Feature Scatter Plot

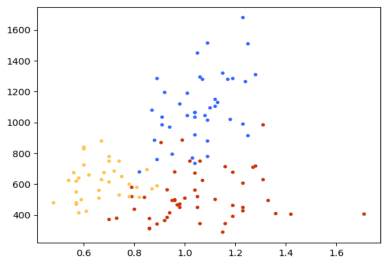


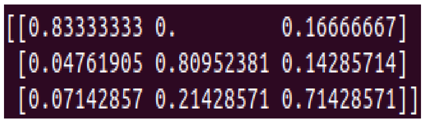
Figure 2: Scatter Plot of Feature 11 vs 13

thorough observation, we decided to go ahead with features **11** and **13** (*Figure 2*) as the data had a fair spread.

2 K-Nearest Neighbours

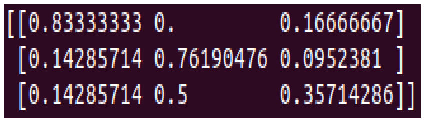
We were tasked with implementing a **KNN classifier** to classify the data-points and predict the class of the respective point. We used our selected features to reduce the training and test set. Our first step was to calculate the Euclidean Distance between each test and training instance from the respective sets. We then sorted the distances in increasing order and only selected the first **k** distances. We then used three counters (as there are three classes) to keep track of the number of neighbours from each class. Eventually, we predicted the class of the test instance by choosing the one with the highest counter.

2.1 Challenges



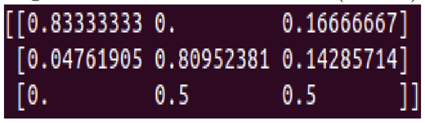
```
[[0.83333333 0.         0.16666667]
 [0.04761905 0.80952381 0.14285714]
 [0.07142857 0.21428571 0.71428571]]
```

Figure 3: Confusion Matrix (k = 1)



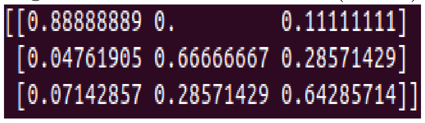
```
[[0.83333333 0.         0.16666667]
 [0.14285714 0.76190476 0.0952381 ]
 [0.14285714 0.5        0.35714286]]
```

Figure 4: Confusion Matrix (k = 2)



```
[[0.83333333 0.         0.16666667]
 [0.04761905 0.80952381 0.14285714]
 [0.         0.5        0.5       ]]
```

Figure 5: Confusion Matrix (k = 4)



```
[[0.88888889 0.         0.11111111]
 [0.04761905 0.66666667 0.28571429]
 [0.07142857 0.28571429 0.64285714]]
```

Figure 6: Confusion Matrix (k = 7)

Once we implemented our algorithm, we came across a possible situation of **ties**. When we ran our algorithm, we realised a test instance had the same number of neighbours for at least two classes for **k = 2** and **k = 4**. To try and avoid such cases, we tried to assign a *weight*, which is inversely proportional to the mean of distances of each class, and choose the one with the highest value. However, this reduced our accuracy which led to us choosing the class *randomly*. Another challenge we faced was choosing the right value of **k**. We ran our function for different values of k and generated the corresponding **confusion matrices** (*Figure 3*). Above is a comparison of confusion matrices for **k = 1, 2, 4 and 7**. We can see from the matrices that for k = 1 and k = 7, the classifier makes fairly accurate predictions, having lower false prediction rates, thereby returning an accuracy of **79.24 percent** and **73.58 percent** respectively. But in the case of a tie in k = 2 and k = 4, the confusion matrix clearly shows the higher rate of false predictions, thereby resulting a lower accuracy of **67.92 percent** and **73.58 percent** respectively.

3 KNN - Three features

Taking the KNN Classifier a step further, we were tasked to use **three** features instead of the two selected. We started off by making a scatter plot of every feature against our two previously selected features. We then computed **covariance matrices** for the same. This helped us visualise and confirm our selection of feature 8 having the best spread and highest covariance among them. We reduced our train set to the features **8, 11** and **13**. Similarly, we ran the KNN algorithm developed above on the new reduced train set. Below is a comparison chart. The benefit of choosing three features is an increase in accuracy however,

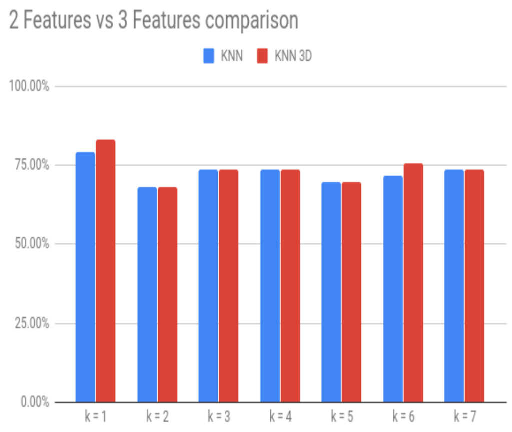


Figure 7: 2 Features v 3 Features

there is a greater risk of overfitting by using more features due to the *curse of dimensionality*. We hoped to address this problem by performing feature selection in pre-processing but the results show no conclusive evidence to choose three features over two.

4 Alternative Classifier - Naive Bayes Classifier

We found the likelihood of each test sample in the reduced test set and assigned the class with the highest probability to the respective sample. We then compared our results with the KNN classifiers.

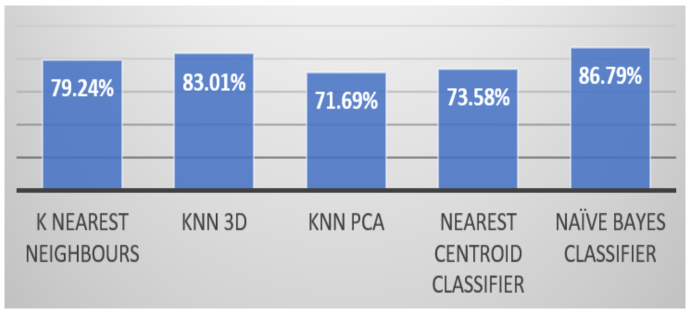


Figure 8: Comparing Accuracy of Naive Bayes with KNN (default k = 1)

4.1 Comparison with KNN

To understand the performance of our Naive Bayes Classifier better, we compared the final accuracy against the results of the above KNN classifiers for $\mathbf{k} = \mathbf{1}$. We can clearly see that the Naive Bayes Classifier gives better results. This is because KNN works on the tuning of **k** and choosing the optimal value for a new entry is not feasible for a real time prediction model. The KNN method is a **lazy** classifier and works without any pre-computations. The classification is done purely on the basis of nearest neighbours, but can significantly slow down with every new test entry since there is a need to compute distances. While Naive Bayes is an **eager** classifier and requires pre-computation of prior probabilities, it computes the respective likelihoods and classifies the samples on the basis of the highest probability. This allows the algorithm to work with smaller training data sets, and return faster and more accurate results.

5 KNN - Principal Component Analysis

We wanted to make sure our algorithm returns accurate results for automatic as well as manually selected features. We used Principal Component Analysis to do this. We first used Scipy's PCA to reduce the features and then transformed the train and test accordingly. With these newly transformed sets, we ran our KNN classifier and compared the results with our previous KNN classifiers for two and three **manually selected features**. As we can see from *Figure 9*, for

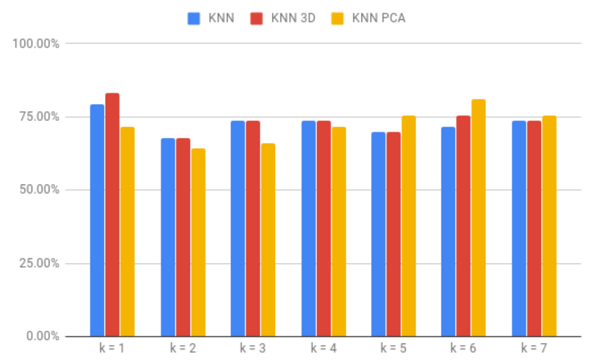


Figure 9: Comparisons of all KNN

lower values of **k**, there is a significant decline in accuracy for the PCA reduced features, but returns a higher accuracy for larger **k** values. This is because the PCA works by calculating the eigenvalues and eigenvectors for the training set and selects the two features with the highest variance thereby reducing uncertainty with feature selection. We also believe that using PCA to select features on our training set is not ideal due to the small number of features.