Data Mining: Major Coursework

Ryan Watkins

March 18, 2016

Contents

1 Task 1 Data Exploration and Clustering

Using the wine dataset, we explore clustering and PCA technique. PCA reduces the dimensionality of the data and organizes the data into principle components of original dimension size. The first PCA component has the highest variance, the second PCA component has the second highest variance and so on. It's usage is largely exploratory but can also be used for predictive modelling. It's usage can be thought of to show the structure of the variance within the data [5]. Figure 1 shows PCA on the wine dataset unnormalised.

For clustering, K-means clustering is utilized. K-means clustering technique organizes points into groups using euclidian distance from centroids and a preset amount of centroids, the K. Initial setup of the centroids is a research topic that has been widely explored [10], [1]. These centroids are somehow determined and then iteratively for each observation the nearest centroid is found by computing the distance between the observation and centroid for every centroid. After the observations are assigned a centroid, the centroids are recomputed by taking the average of the observations that now fall within the centroid. When there is no change in an iteration, convergence is reached.

1.1 Task1.1: Unnormalised PCA and K-means Clustering

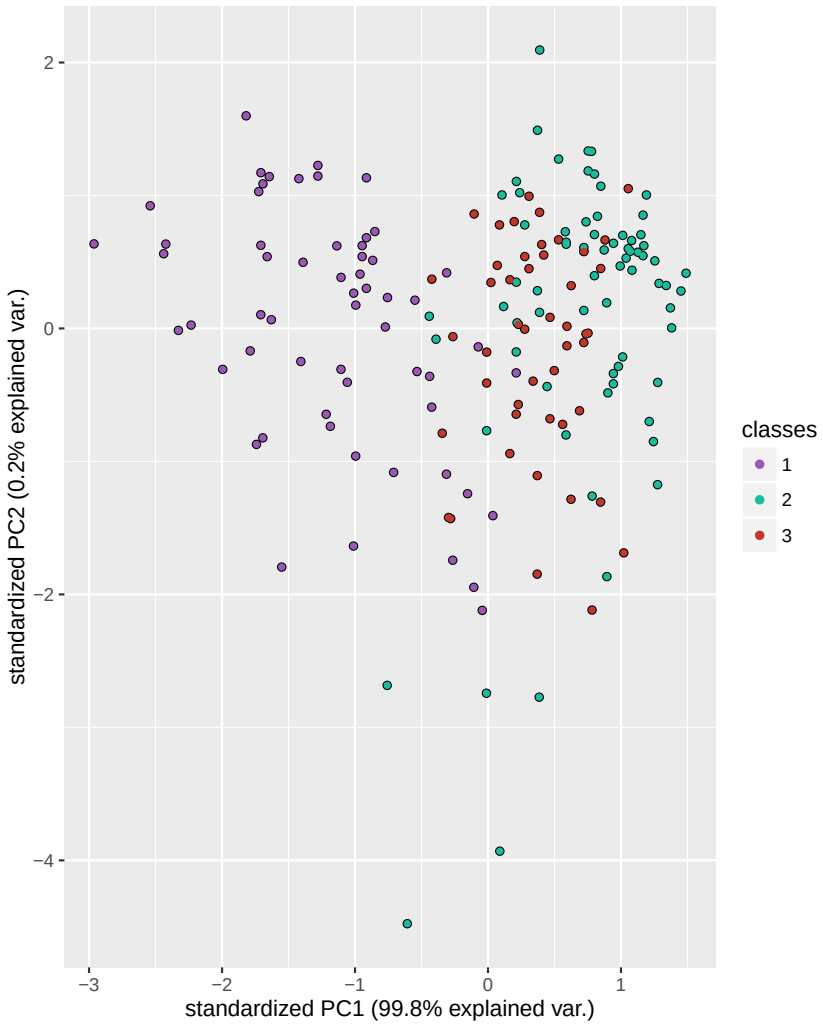

Figure 1: *Plot 1.* PCA on Wine dataset unnormalised

Performing clustering within *R* can be done with the *kmeans* function call from the library *clustering*. Thus, clustering is performed with:

```
rm <- kmeans(wined[,2:14], 3)
autoplot(cd_k_unnorm, data=scale(wined[,2:14]))
```

Figure 2: R code which performs K-Means clustering on unnormalised wine dataset

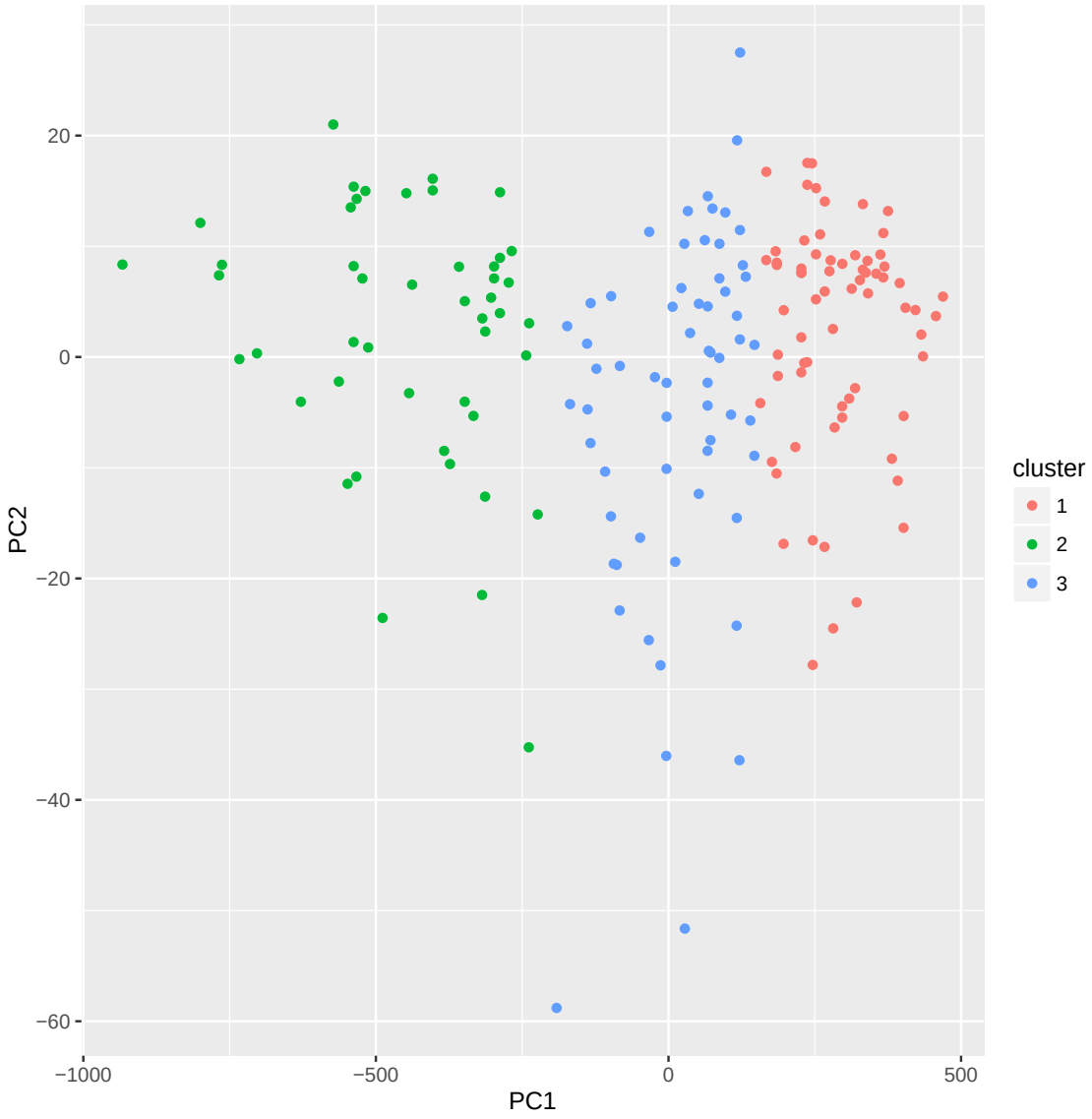

Figure 3: *Plot 2.* K-means clustering on Wine dataset without normalisation with cluster amount set to three

One can visualize each cluster and begin to look closely at the class distributions within the clusters. This can be used to explore the data more. Figures 4, 5, 6 demonstrates the class breakdown within each of the three clusters.

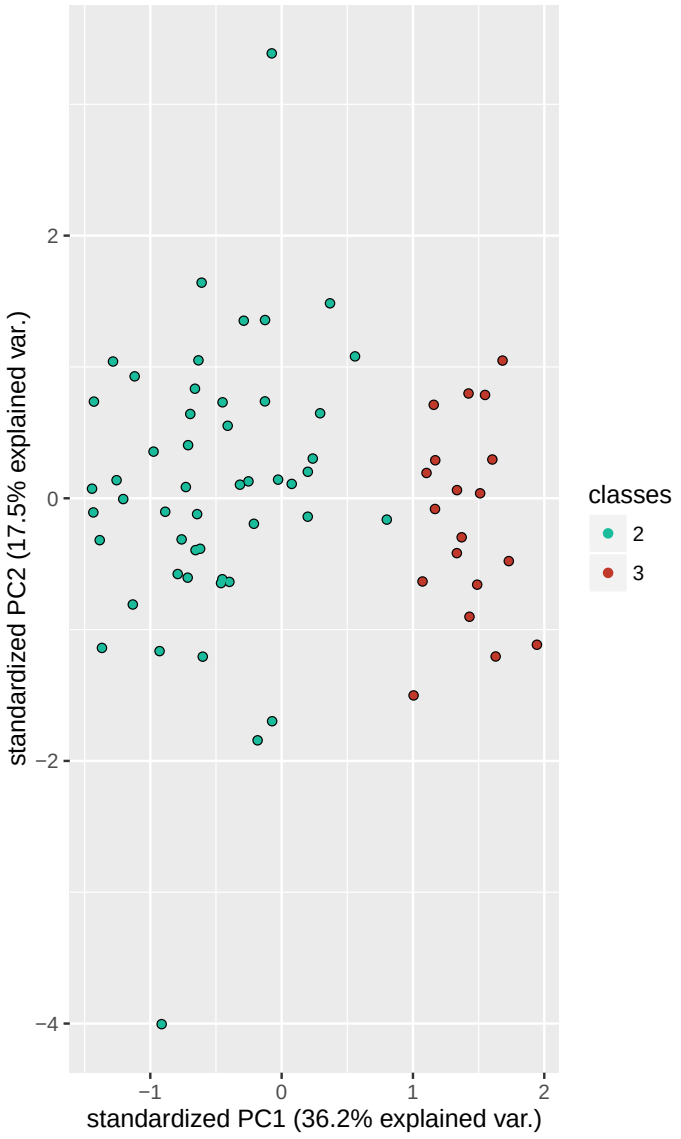

Figure 4: *Plot 3a.* 2D plot with class labelling for cluster one. The plot captures close to even proportion of class two and class three observations.

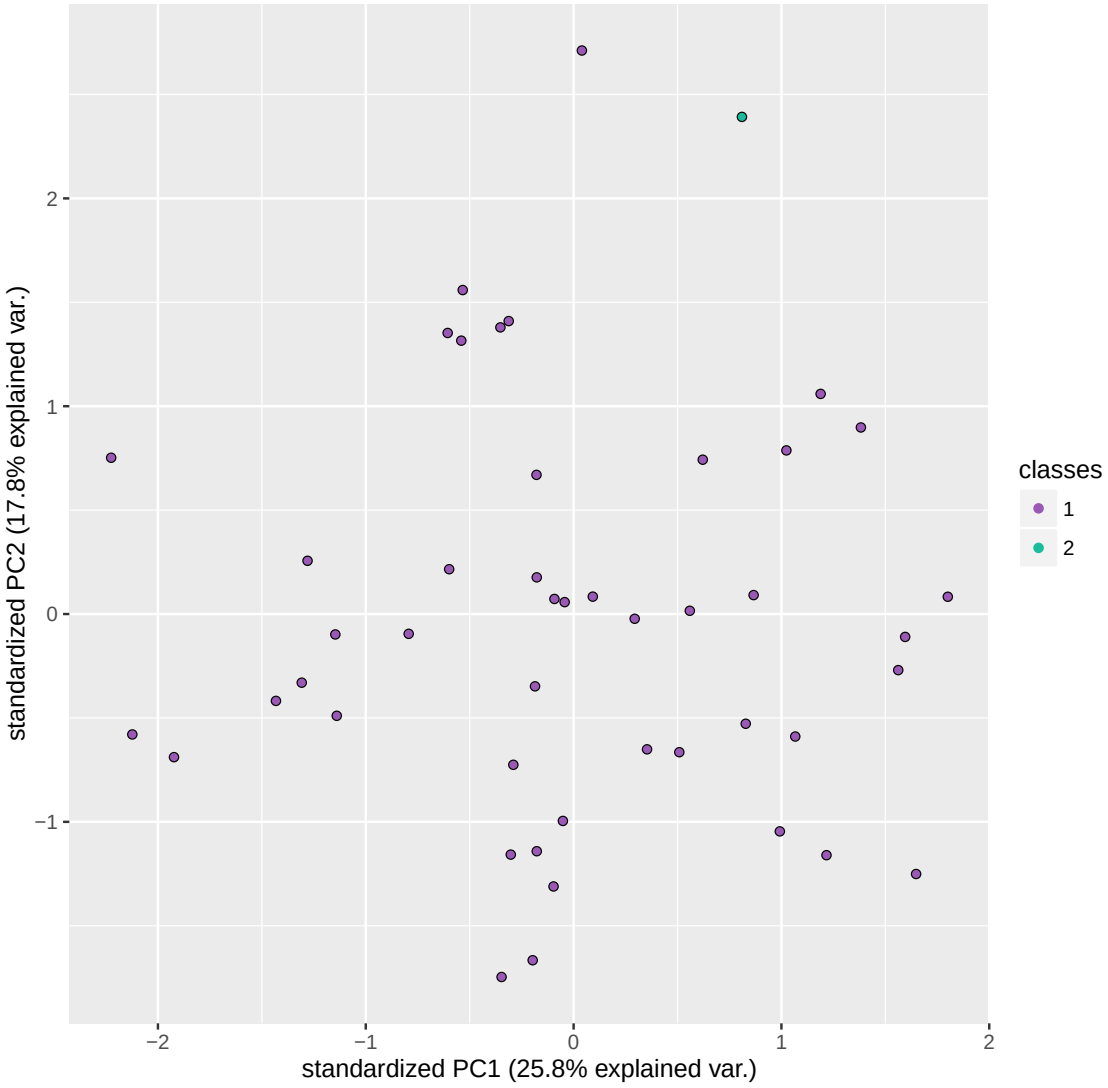

Figure 5: *Plot 3b.* 2D plot with class labelling for cluster two. Heavy observation of class one. One data point from class two.

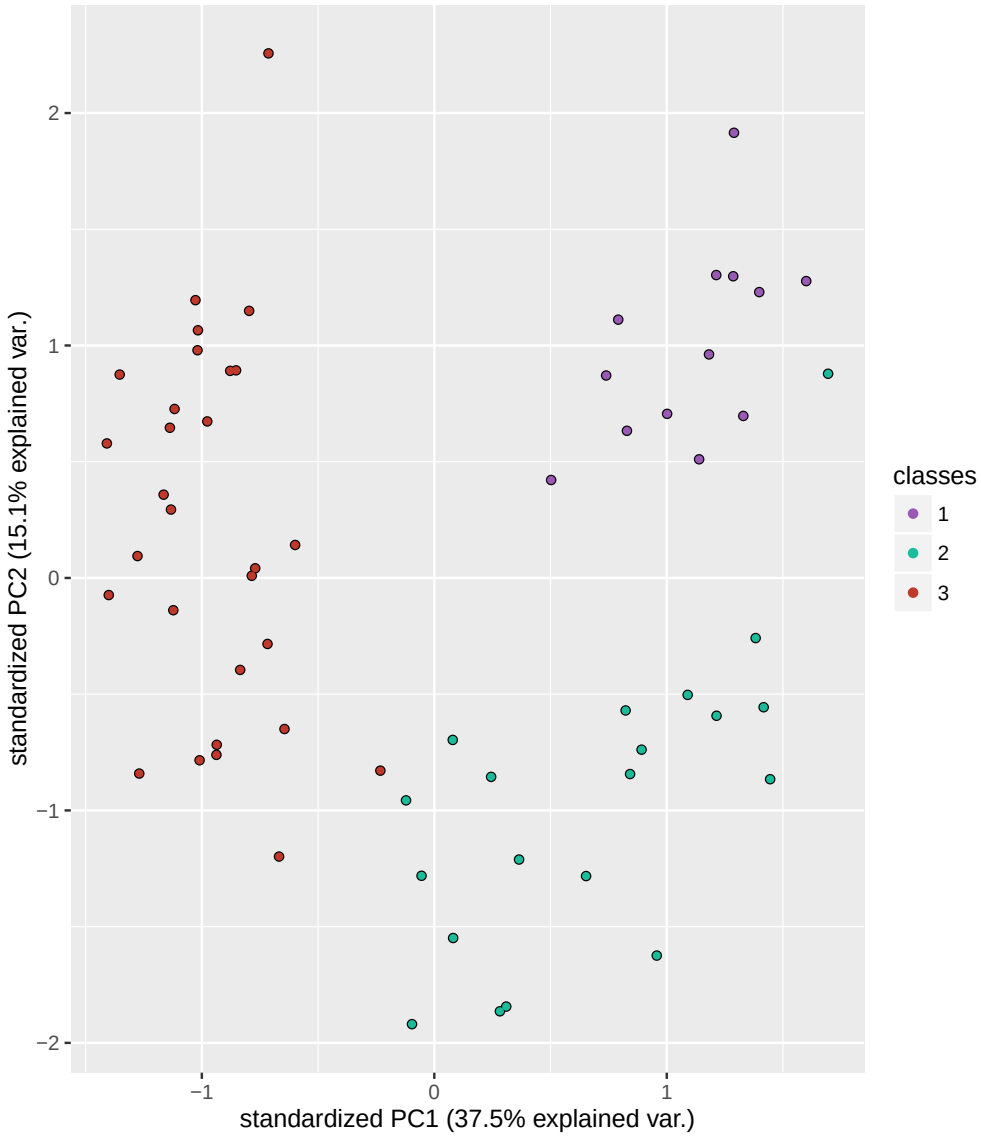

Figure 6: *Plot 3c.* 2D plot with class labelling for cluster three. Captures almost even distribution of all three classes.

1.2 Task1.1: Cluster Validity

The package *clValid*[3] is utilised from the CRAN repository to analyse and validify the clustering. *clValid* contains three internal validity measures: Connectivity, Silhouette Width and Dunn Index. It can be used to compare multiple clustering methods also to determine which clustering method is best for the data.

Connectivity indicates the degree of connectedness of the clusters, determined by k-nearest neighbours. It is a function that should be minimized [3].

Silhouette Width is the average of the degree of confidence in the clustering assignment of a particular observation. It is a function that should be maximised in the range[-1,1] [3].

The Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance. Dunn Index should be maximised and has values between zero and infinity [3].

Below, Table 1 shows the validity results for the unnormalised wine dataset. The R code which implements the clustering validity is shown in Figure 7

```
# Attempt to perform cluster validation
intern_unnorm <- clValid(wined[,2:14], 3, clMethods=c(
  "clara", "kmeans", "hierarchical"), validation="internal")
summary(intern_unnorm)
```

Figure 7: R code using *clValid* for clustering validity on wine dataset. Compares Clara, K-Means and Hierarchical clustering.

Measure	Value
Connectivity	10.0151
Dunn	0.0193
Silhouette	0.5533

Table 1: Internal clustering validity results for unnormalised wine dataset

1.3 Task1.2: Normalised PCA and Clustering

PCA on the normalised wine dataset splits points into groups almost entirely by class which shows that clustering will be an easy task. Normalisation and PCA is performed as detailed in Figure 8. The *scale* argument normalises the data. Figure 9 shows the application of PCA on the wine normalised dataset.

```
df.pca <- prcomp(df[,2:14], scale. = TRUE)
```

Figure 8: R code which normalises the wine dataset and applies PCA

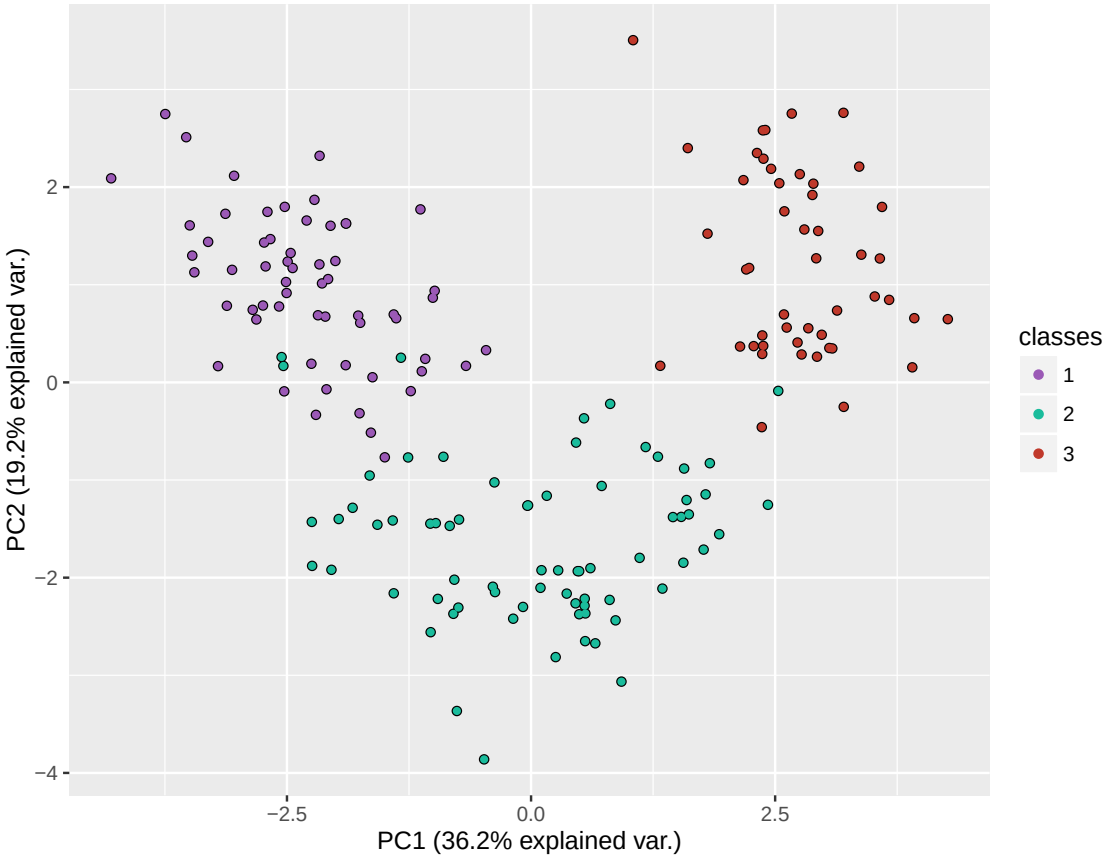

Figure 9: *Plot 1. Normalised.* PCA on Wine dataset after normalisation

The K-means clustering technique on the normalised wine dataset derives well defined clusters. The clusters are formed in a different shape from the unnormalised dataset, instead there exists a 'U shape'. Again, the R code calls *kmeans* function and passes in the normalised data instead this time. Figure 10 demonstrates K-Means on the normalised wine dataset. Figures 11, 12 and 13 demonstrate the class breakdown in each of the three clusters.

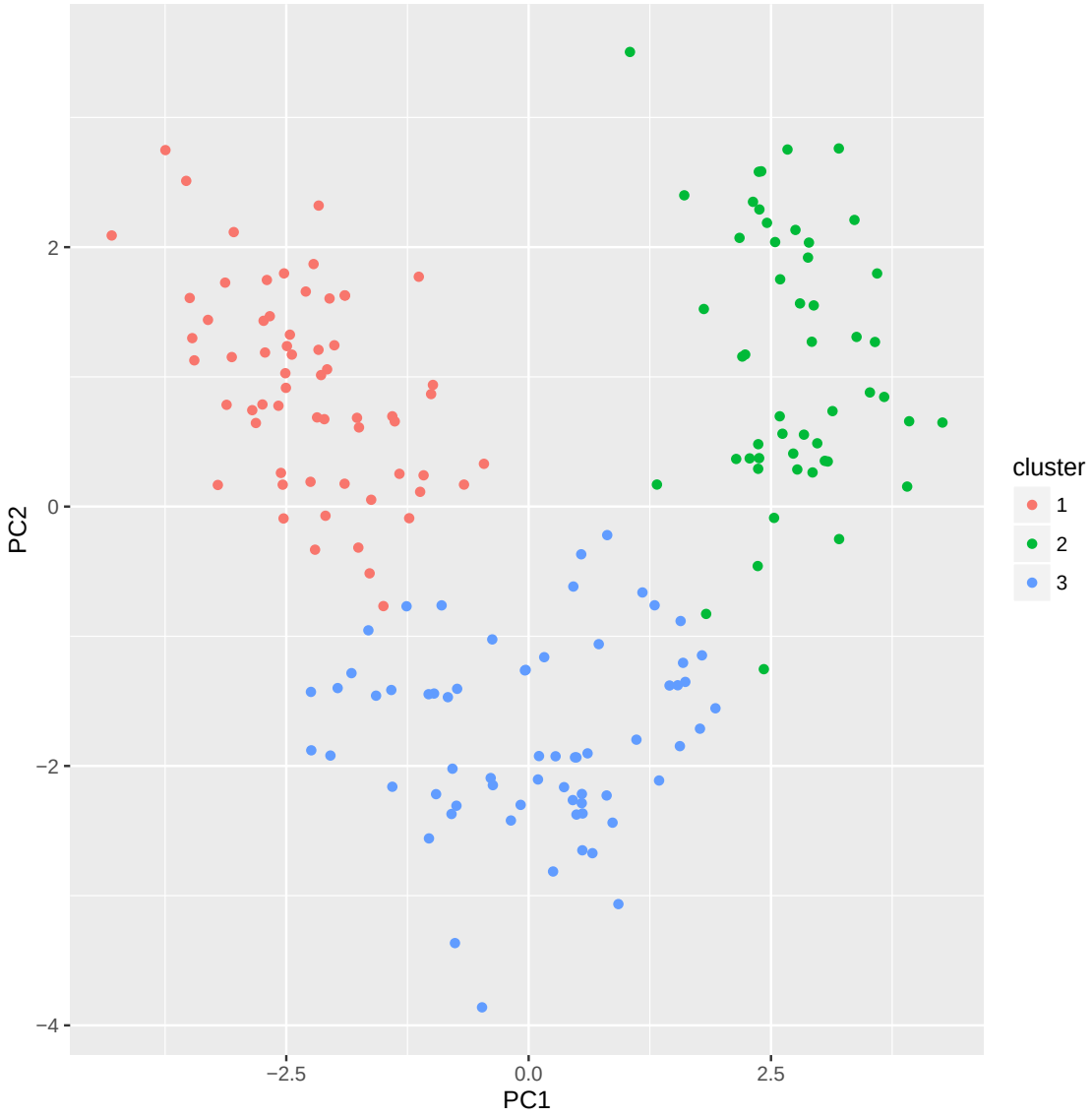

Figure 10: *Plot 2. Normalised.* K-means clustering on Wine dataset with normalisation

Cluster one contains the vast majority of class one and a few observations from class two. In comparison to cluster one for the unnormalised wine dataset, there exists strong classification towards class one where-as cluster one in the unnormalised dataset had zero points from class one and no definitive labelling towards one class.

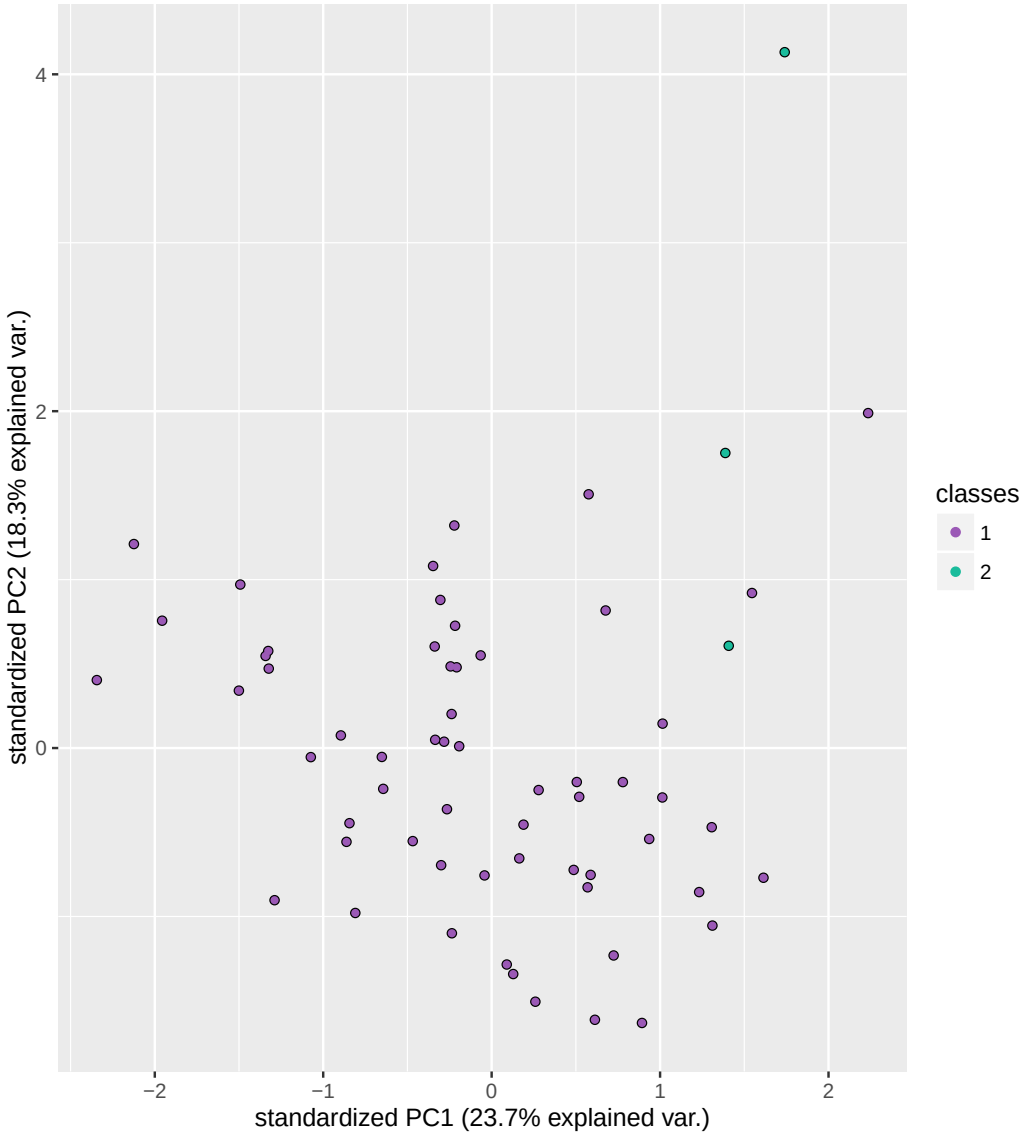

Figure 11: *Plot 3a. Normalised.* 2D plot with class labelling for cluster one after normalisation.

Cluster two contains strong classification towards class three.

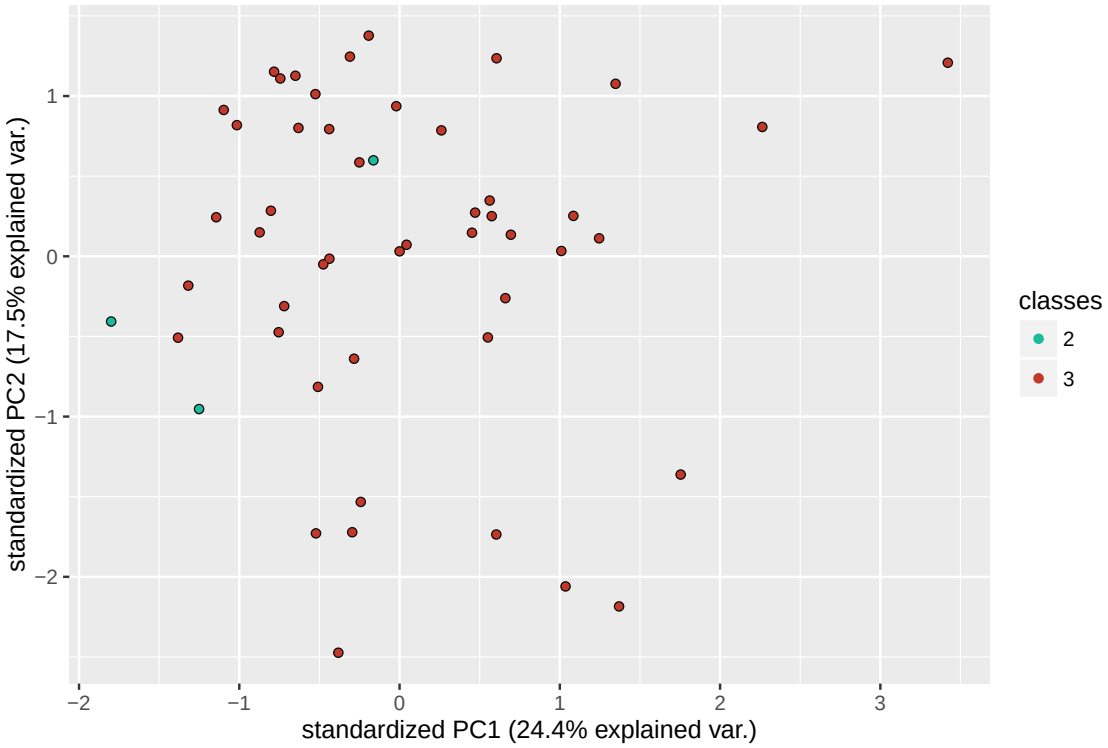

Figure 12: *Plot 3b. Normalised.* 2D plot with class labelling for cluster two after normalisation.

Cluster three contains points formed from class two. Again, one can observe that normalisation derives clusters which preserve classes better. The unnormalised wine dataset had an even distribution of all three classes in cluster three.

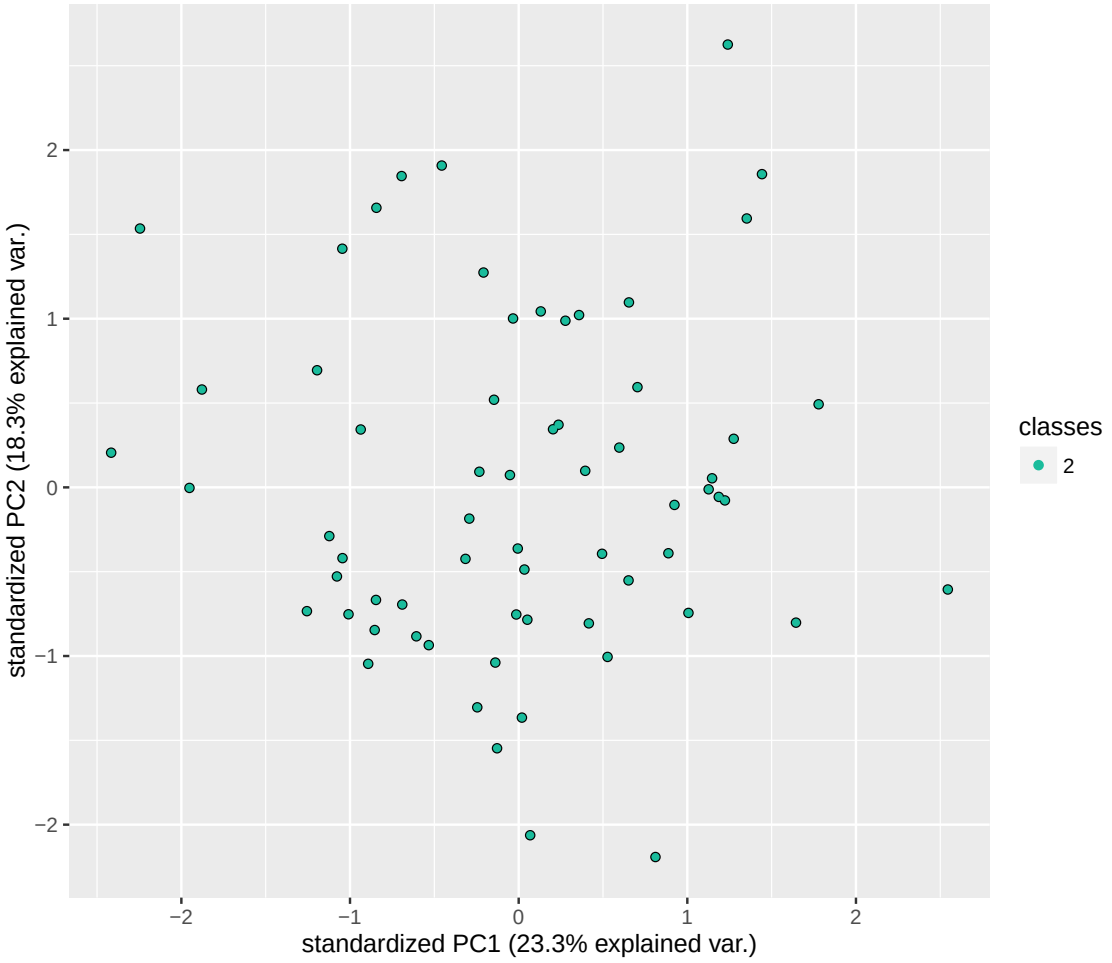

Figure 13: *Plot 3c. Normalised.* 2D plot with class labelling for cluster three after normalisation.

1.4 Task1.2: Cluster Validity

The connectivity for the normalised dataset after performing k-means clustering technique performs worse than the unnormalised dataset. However, Dunn Index and Silhouette both perform better on the normalised dataset. Thus, the normalised dataset clusters better overall on these measures with K-means technique as two measures out of three are better for the normalised dataset. Below in Table 2 are the validity measure values for the normalised wine dataset.

Measure	Value
Connectivity	28.0504
Dunn	0.2323
Silhouette	0.2849

Table 2: Internal clustering validity results for normalised wine dataset

1.5 Conclusion

It is clear that normalisation for the wine dataset helps with classification and this is easy to see visually. Using a variety of clustering validity measures helps as you can take a majority vote on the *winners* from the measures. This helps to verify further that the clustering method is better than the other.

Utilizing PCA is good for visualizing high-dimensional data. PCA Analysis is critical within data mining as visualizing and exploring high dimensionality data is highly important during the pre-processing stage to see the variance in the data or more general attributes that *describe* the data as a whole.

A data scientist can begin to make important decisions in the pre-processing and data exploration stage which will derive a better result during classification or prediction in later stages. For instance, a data scientist may discover noise within the dataset upon applying PCA or other pre-processing techniques and begin to discard unimportant attributes also (feature selection).

2 Task 2: Comparison of Classification Models

KNN and Decision Trees are used to perform classification on the wine dataset. Ten-fold cross-validation is used to generate generalisation error and provide validity.

2.1 Ten-fold cross-validation

Ten-fold cross-validation is performed by holding out data across a dataset ten times, such that a 90%/10% split is obtained for train and test respectively until the entire dataset has been tested and trained. During each holdout, one performs classification and stores results for analysis after the iterative cycle of the ten folds has finished. Shuffling the data and picking data in the folds such that the class balance is preserved is important and utilized in ten-fold cross-validation techniques. There is a ten-fold cross-validation loop which makes folds and runs the process inside a for loop that iterates ten times. Ten-fold cross-validation has been studied in [8]. The implementation can be found inside *dm_2_dt.R* and *dm_2_knn.R*.

2.2 Decision trees

The package *rpart* is used to form the tree using Breiman et al's Classification and Regression Trees (CART) method[2]. The method minimizes the error in each leaf of the tree. No cost matrix or weighting is necessary in modelling as classes are well distributed. It's a good model for describing data. Other decision tree algorithms include ID3, C4.5 and C5.0. Figure 14 demonstrates application of CART Decision Tree technique on the wine dataset.

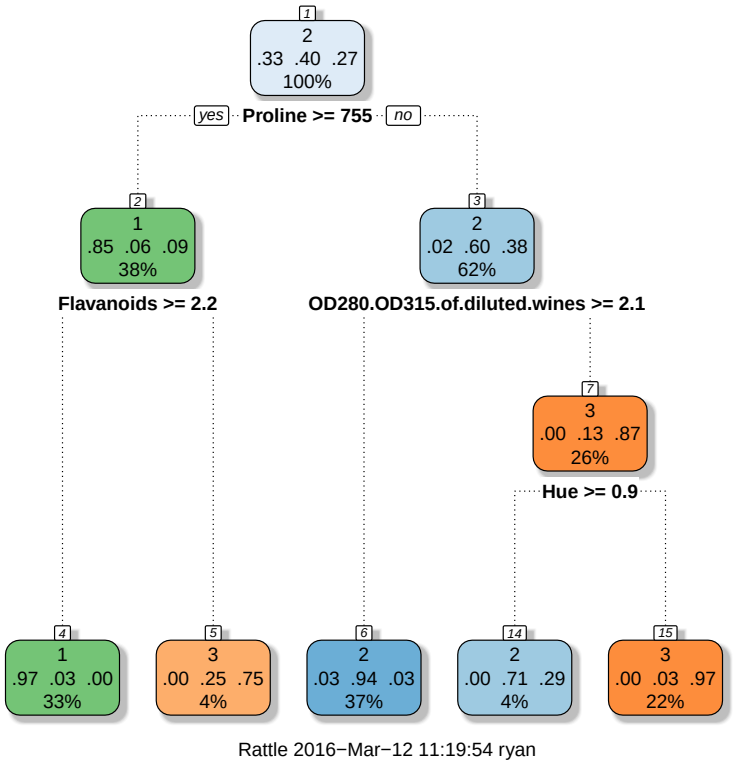

Figure 14: Decision tree model using *rpart* on wine dataset

Using *plotcp*, possible cost-complexity prunings can be visualized and interpreted accordingly. *plotcp* is used below on one fold. Figure 15 indicates that 0.017 is the best value to pick for pruning. Tree pruning has been studied in [4]. However, results show later that pruning was unnecessary as the *rpart* function had already taken care of pruning.

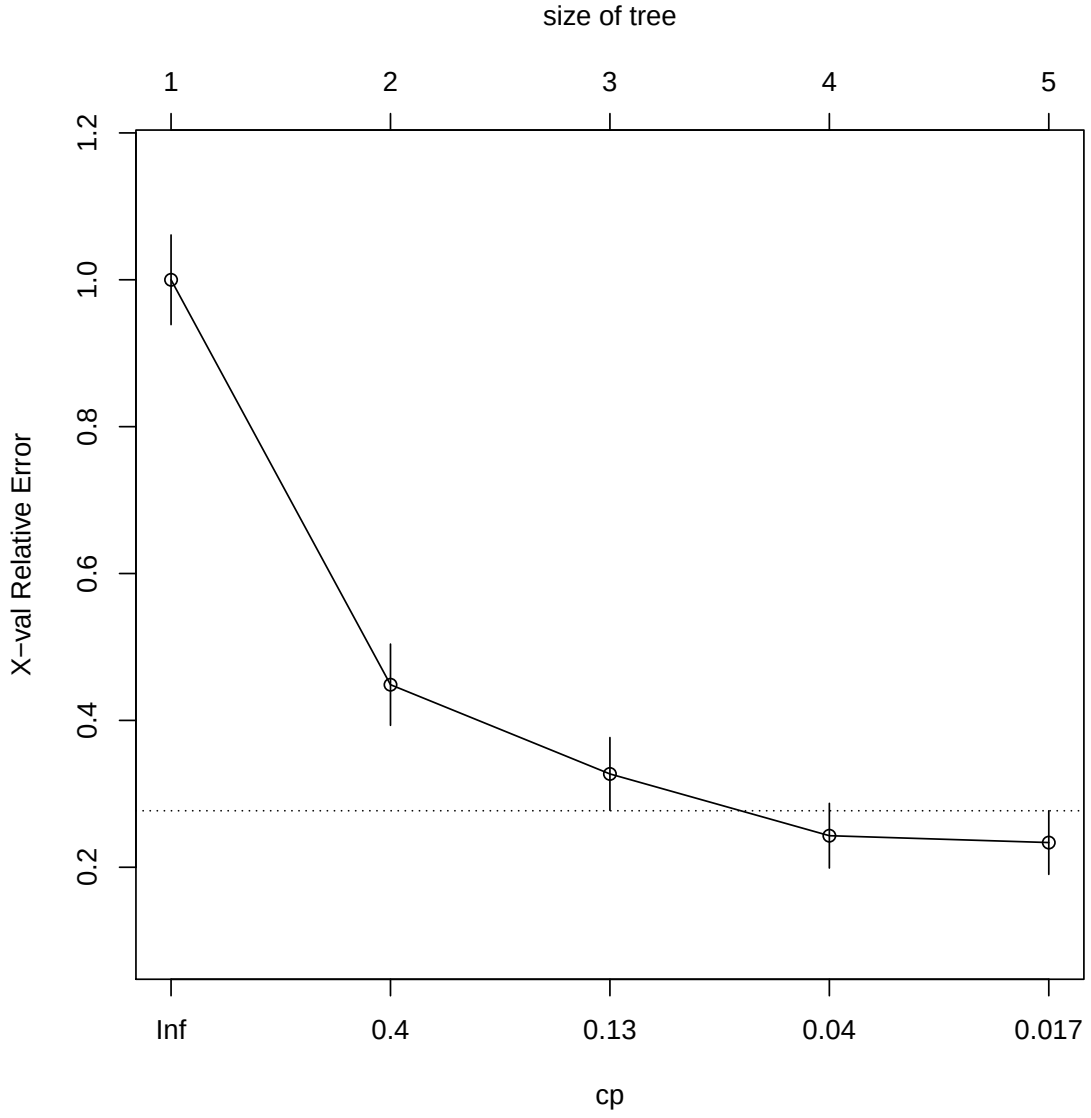

Figure 15: Tree pruning plot in R, useful for interpreting how to prune

2.3 KNN

KNN is an algorithm that slowly and accurately classifies data. It classifies by taking a majority class vote on K nearest neighbours. Getting K-selection correct is important and thus as a rule of thumb, $K = n^{\frac{1}{2}}$ is used. Using R's built-in *knn* function, a generalisation error rate of 0.03954248 is derived.

2.4 Comparison Results

A ROC curve would have been useful for analysing the classification methods, however they require binary classification. A workaround is the usage of Multi-class Area Under the Curve (AUC) as described by Hand and Till [6]. The package *pROC* provides multi-class AUC in R.

Table 3 compares the two classification algorithms. The table demonstrates that KNN performs better classification on the dataset. Figure 14 shows Decision Tree's descriptive power.

Classifier	Measure	Value
KNN	Generalisation error	0.03954248
KNN	Multi-level AUC	0.979
Decision Tree	Generalisation error	0.0620915.
Decision Tree	Multi-level AUC	0.9701

Table 3: Comparison of Decision Trees and KNN on wine dataset

2.5 Conclusion

Comparing classification methods is useful for data scientists as different classification methods work for different problems. It could be that classification accuracy is not the only metric that is important and instead there is a need to describe the data. In this case, decision trees would be picked as opposed to KNN technique. Likewise, performance constraints could also affect choice. KNN is known to be slow on massive datasets and instead perhaps a Neural Network may run faster or indeed regression or decision trees. There is no one-size fits all solution and it's always good to compare and constrast solutions before deciding on using a classification technique.

3 Task 3: The Search for God Particle: a Binary Classification Challenge

A training dataset is provided such that there are 28 attributes and one class. The first 21 attributes comprise the low-level components measured as kinematic properties by particle detectors. The attributes thereafter comprise high level attributes which are made using a complex model as function of the low-level attributes (feature transformation). The task is to classify a dataset that only contains the 21 low-level attributes.

3.1 Imbalanced Dataset Fixing

A quick look at the dataset shows the distribution of classes as in Table 4. The dataset is too imbalanced. An even distribution is needed before running ten-fold cross-validation. Figure 16 shows R code that achieves a balancing of two classes. The class imbalance problem is studied in [7].

Class	Amount
Background	89999
Signal	10000

Table 4: Class distribution in Hadron Collider training dataset

```
even.class.distribution <- function () {
  train100k <- train100k[sample (nrow (train100k)),] # Shuffle data

  classd <- train100k[which (train100k[,29]==0),]
  based <- train100k[which (train100k[,29]==1),]

  ind <- sample (1: (100000-nrow (based)), 10000)
  train100k.evenSplit <<- rbind (classd[ind,],based)
  train100k.evenSplit <<- train100k.evenSplit[sample
                          (nrow (train100k.evenSplit)),] # Reshuffle
}
```

Figure 16: R code to split class distribution evenly

3.2 Forming model for High Level Components

An hypothesis is the seven high level components will help classification. Linear Regression is employed to estimate the high-level attributes using the *lm* function in R. The model is made using the first twenty-one low-level attributes. Table 5 shows the room mean squared errors (RMSE) for each of the 7 high-level components averaged across ten folds in ten-fold validation using the evenly split dataset formed from *even.class.distribution()*.

HL Attr.	Avg. RMSE
1	0.6497242
2	0.3587509
3	0.1530353
4	0.3611804
5	0.4137971
6	0.2930333
7	0.2559048
Total avg	0.3550609

Table 5: Average errors after linear regression on the 7 high level attributes during 10-fold cross-validation

3.3 Classification on train100k

Classification models are built and evaluated on the original dataset using ten-fold cross-validation with datasets containing just low-level values, just regressed high-level attributes, scaled original data and also a combination of original low-level and regressed high-level attributes. The classification methods are logistic regression and neural networks.

During the ten-fold cross-validation *for* loop, each different dataset is formed and evaluated using both aforementioned techniques. To form the datasets, one must regress the high level attributes during each loops and replace the high level data within the training and test datasets. This is accomplished by the function *get.high.levels*. The holdout data is reset after each iteration of the ten-fold cross-validation using a defined function *reset.holdout.data*. The two functions *logit.classification* and *nn.classification* run the logistic regression and neural network classifcation respectively.

3.4 Results for classification on train100k

A breakdown is shown in Table 6. All measure values are averaged across ten folds. ROC Curves are given in Figures 17, 18, 19 and 20. Accuracy curves are shown in Figures 21, 22, 23 and 24. A key statistic is that the best *low.reg.hl* accuracy was 0.613. It's

model is utilized later to predict the new dataset with missing components. The results infer which model will work best on the unknown data. It should be noted though that the scaled dataset is not feasible for application on the unknown data as the high level components are unknown in the unknown data.

Classifier	Dataset	Accuracy	Precision	Recall	F-measure
Logistic Regression	low	0.5619	0.5610363	0.5694674	1.708402
Logistic Regression	reg.hl	0.5756	0.5795118	0.5516499	1.65495
Logistic Regression	scaled	0.63415	0.6021862	0.7922165	2.376649
Logistic Regression	low.reg.hl	0.5619	0.5610363	0.5694674	1.708402
Neural Network	low	0.5961	0.598144	0.5858072	1.757421
Neural Network	reg.hl	0.5984	0.5980422	0.6013244	1.803973
Neural Network	scaled	0.6884	0.6890128	0.6878997	2.063699
Neural Network	low.reg.hl	0.5977	0.5993678	0.5914822	1.774447

Table 6: Classification results: four datasets, two classification methods and four measures

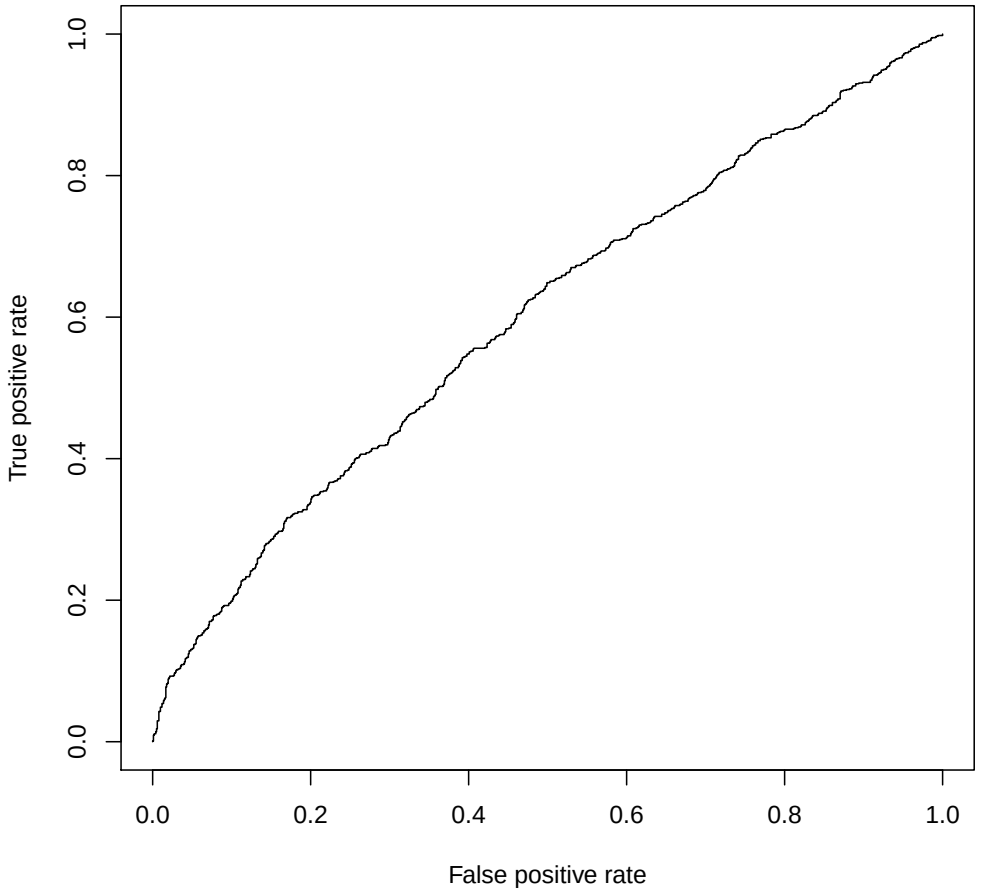

Figure 17: ROC Curve on low-level attributes logistic regression model during last fold of ten-fold cross-validation

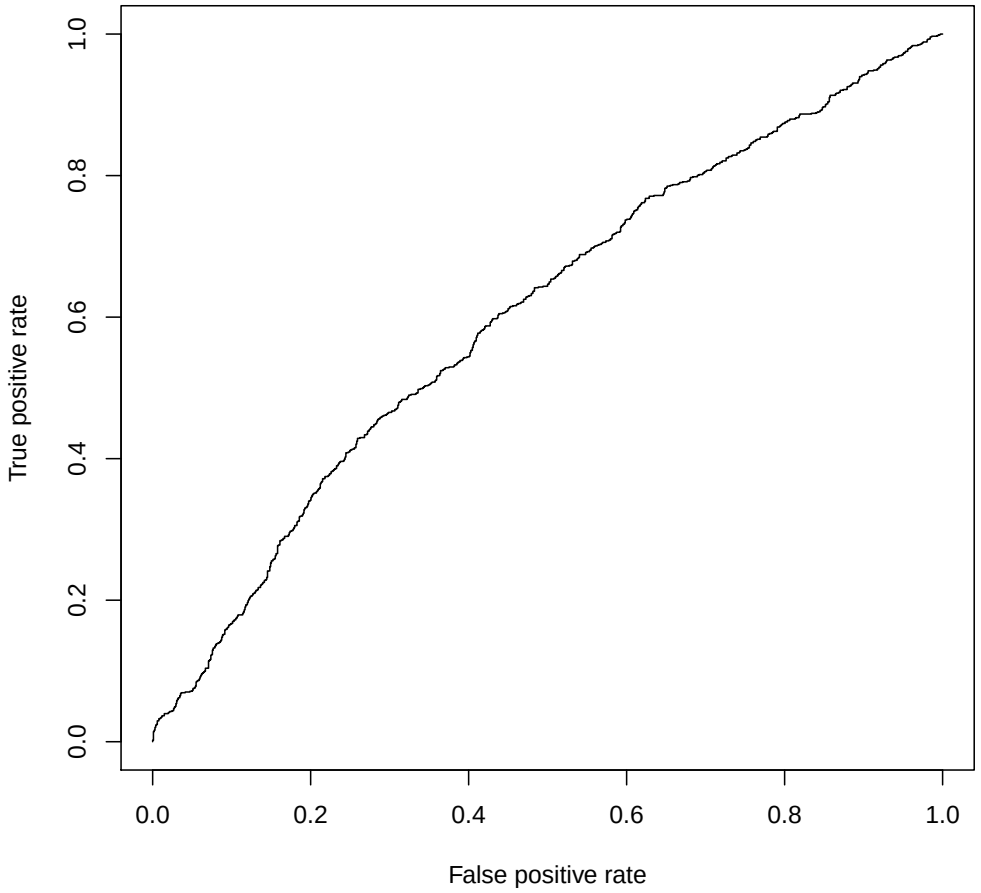

Figure 18: ROC Curve on just high level regressed attributes logistic regression model during last fold of ten-fold cross-validation

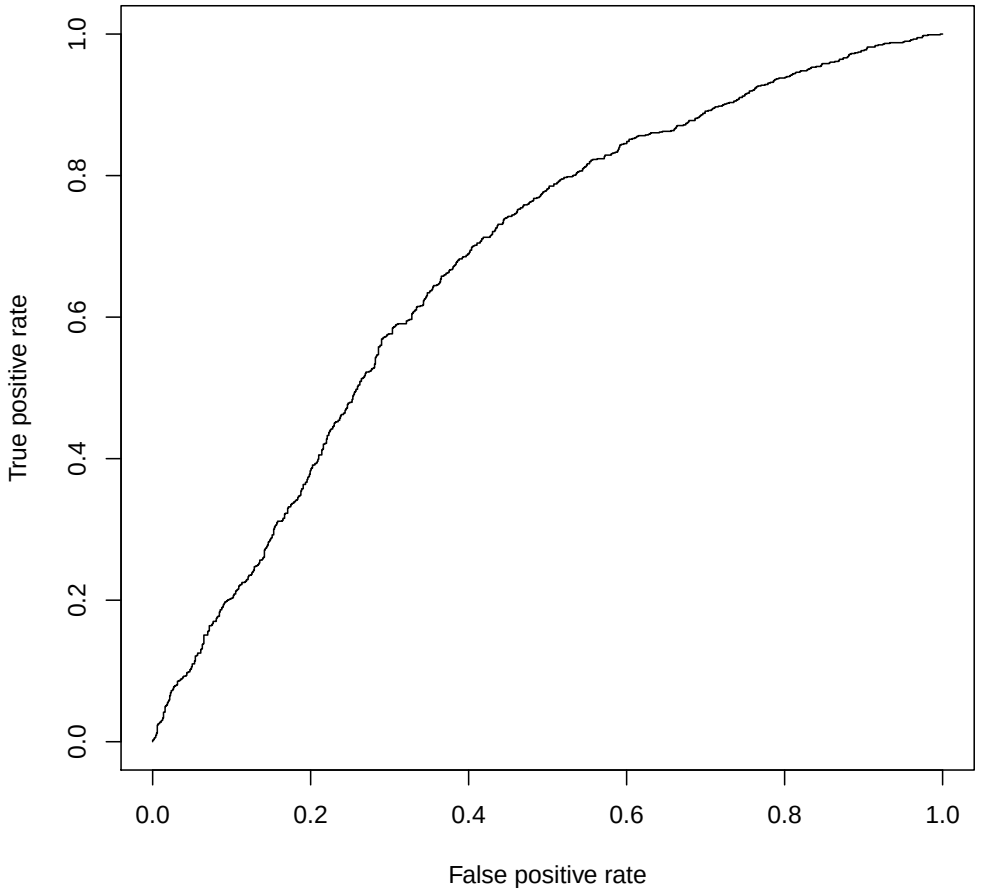

Figure 19: ROC Curve on scaled dataset logistic regression model during last fold of ten-fold cross-validation

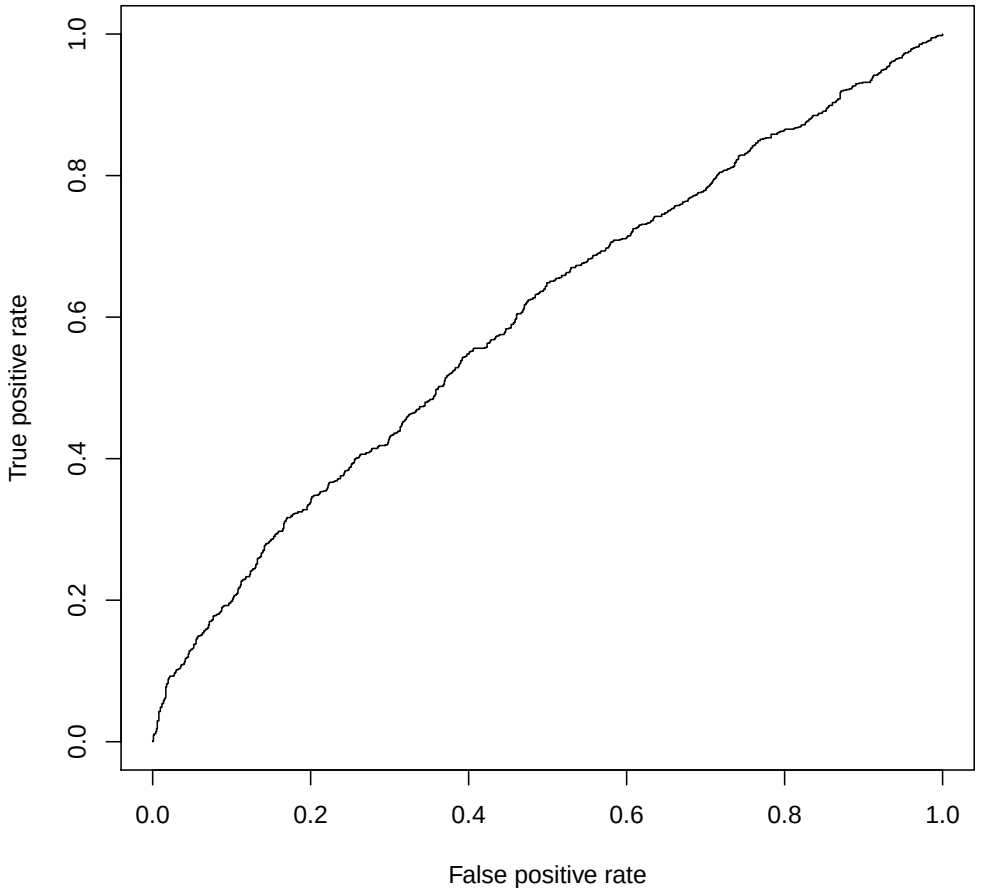

Figure 20: ROC Curve on combination of low-level attributes and regressed high-level attributes logistic regression model during last fold of ten-fold cross-validation

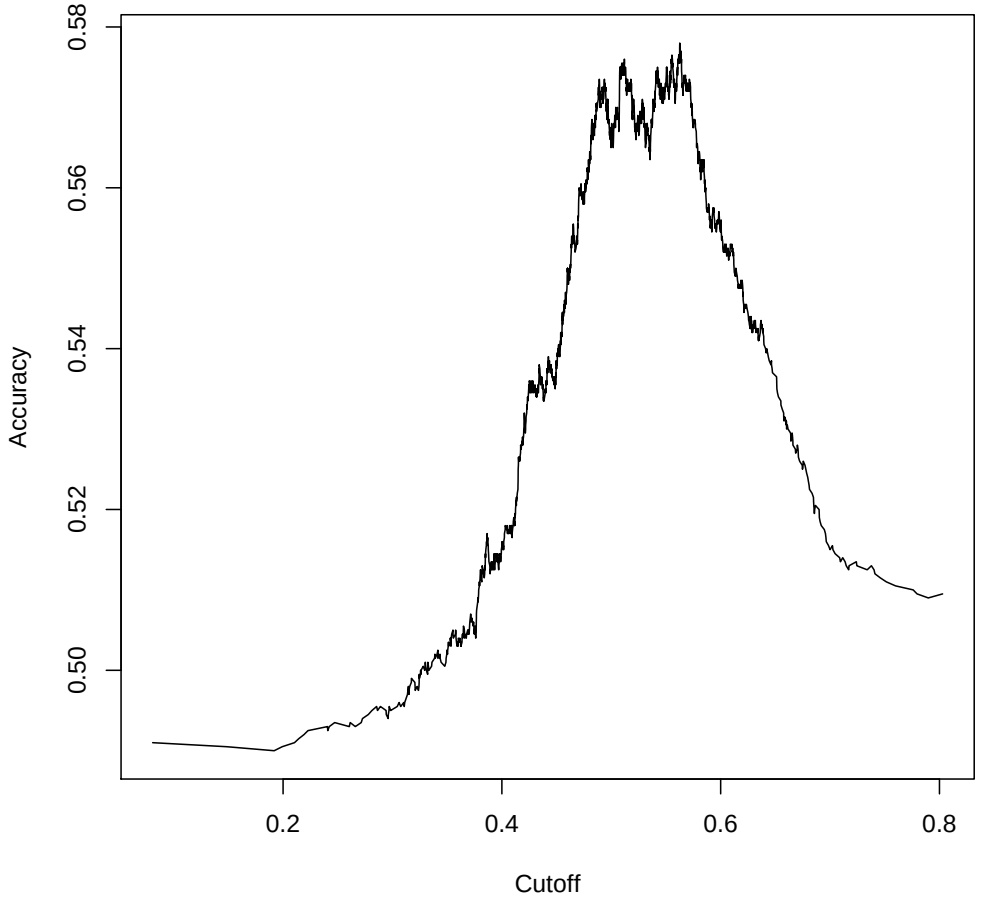

Figure 21: Accuracy Curve on low-level attributes logistic regression model during last fold of ten-fold cross-validation

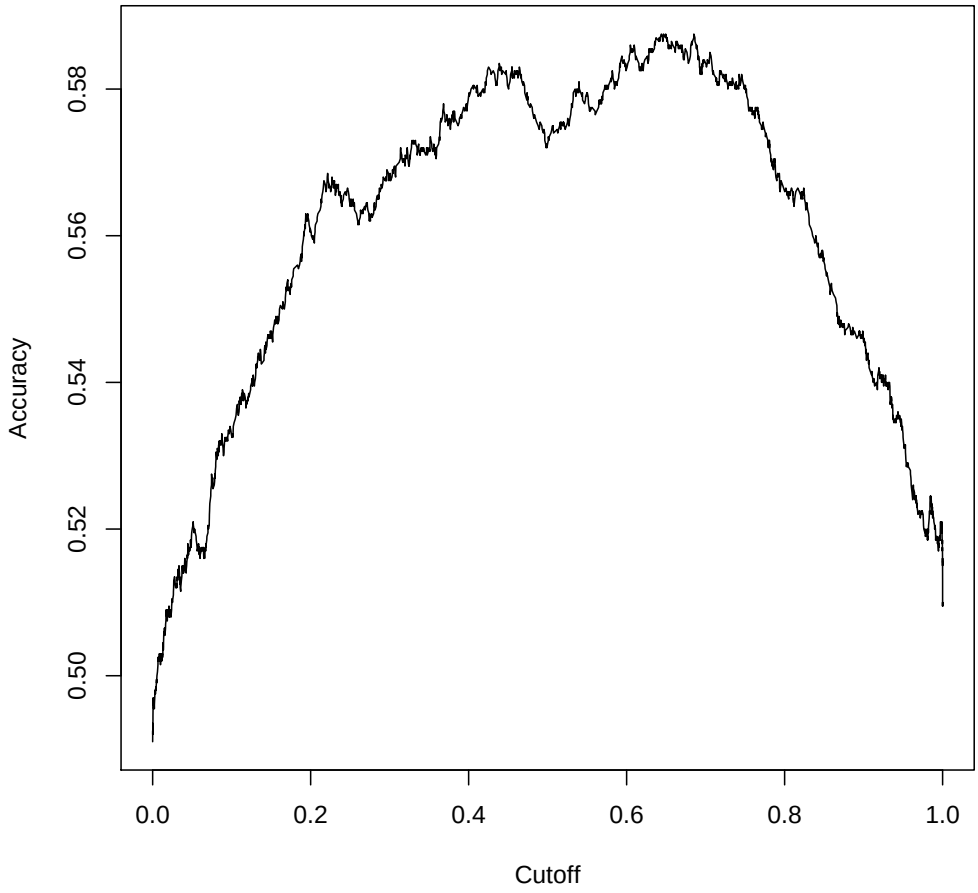

Figure 22: Accuracy Curve on just high level regressed attributes logistic regression model during last fold of ten-fold cross-validation

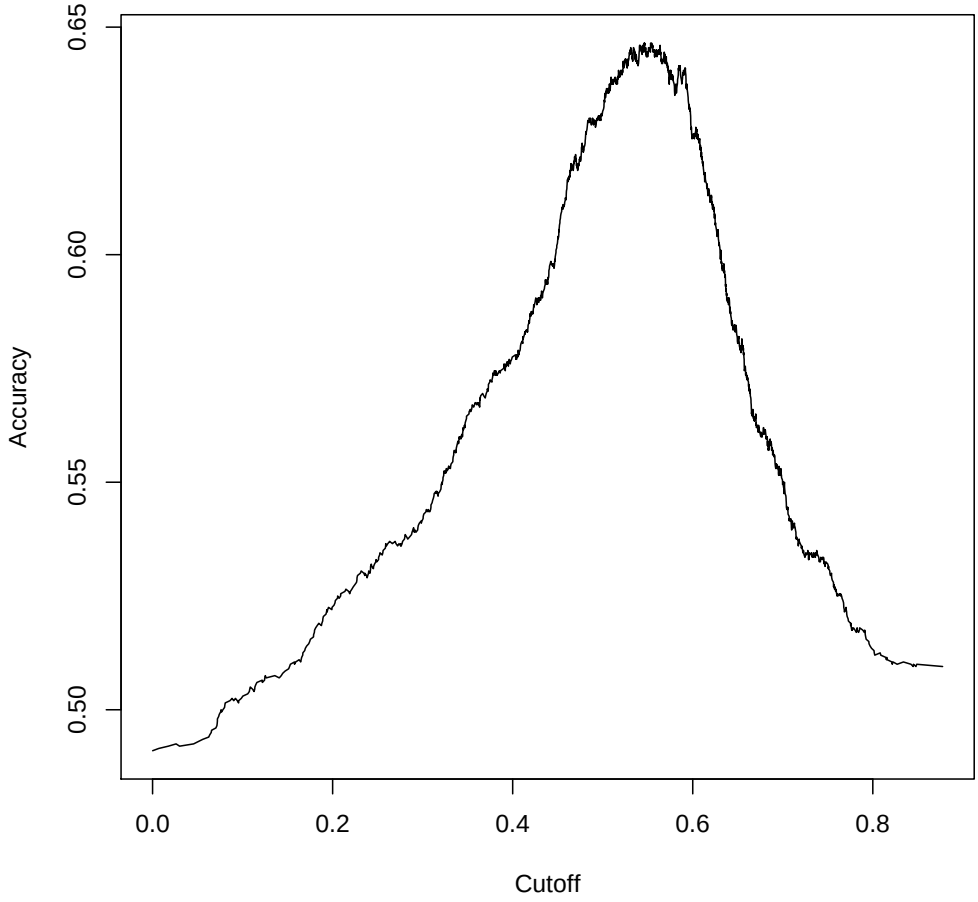

Figure 23: Accuracy Curve on scaled dataset logistic regression model during last fold of ten-fold cross-validation

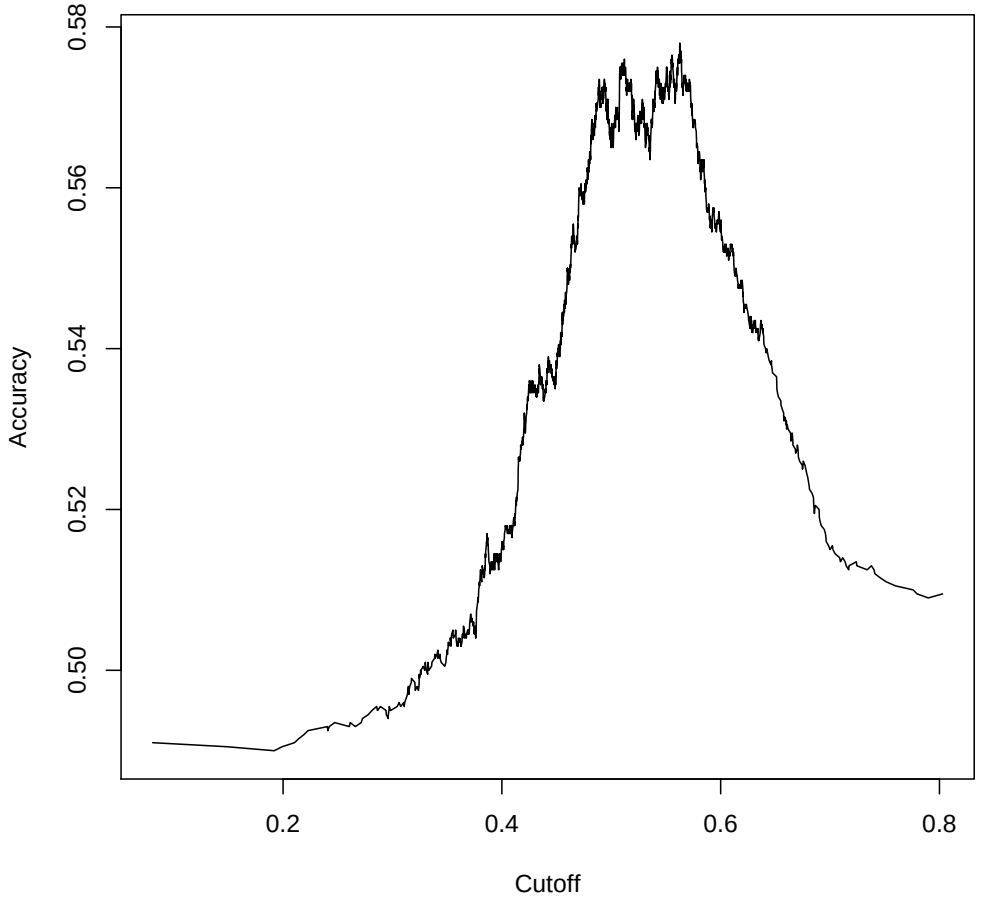

Figure 24: Accuracy Curve on combination of low-level attributes and regressed high-level attributes logistic regression model during last fold of ten-fold cross-validation

3.5 Predicting the unclassified dataset

Prediction is made by using the best model built from the original dataset utilizing the 7 regressed high-level components and 21 low-level components inside ten-fold cross-validation. This model is saved at *models/best_nnet.rda* and loaded to perform classification on the unknown data. Since a result of 0.613 was formed using the evenly distributed train100k dataset during ten-fold cross-validation, one can expect a similar result in classifying the unknown dataset.

Using *lm* gives the new seven high-level components for the new dataset and prediction is performed by applying the aforementioned model and results are saved to a CSV file.

3.6 Conclusion

Results show that combination of low-level attributes and regressed high-level attributes is slightly better than just low-level attributes. However, one should look into improving the calculation for the high-level components by refining the linear regression technique used. Feature interaction and better feature selection could provide help also.

Using all the data is clearly useful but further understanding of the data is necessary to get better classification. It's clear that classification of *H1* needs to be fine-tuned to reduce the overall average misclassification rate of high level attributes.

The problem is hard to classify due to small sample size and dataset imbalance. Advanced techniques may help such as distributed parallel techniques with a much larger dataset[9].

References

[1] Kohei Arai and Ali Ridho Barakbah. Hierarchical k-means: an algorithm for centroids initialization for k-means. *Reports of the Faculty of Science and Engineering*, 36(1):25–31, 2007.

[2] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[3] Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta, et al. clvalid, an r package for cluster validation. *Journal of Statistical Software (Brock et al., March 2008)*, 2011.

[4] Bojan Cestnik and Ivan Bratko. On estimating probabilities in tree pruning. In *Machine LearningEWSL-91*, pages 138–150. Springer, 1991.

[5] George H Dunteman. *Principal components analysis*, volume 69. Sage, 1989.

[6] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[7] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[8] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[9] T Maeno. Panda: distributed production and distributed analysis system for atlas. *Journal of Physics: Conference Series*, 119(6):062036, 2008.

[10] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, 2000.