Anomaly Detection using PCA and kNN Clustering

Initial Setup

```
knitr::opts_chunk$set(echo = TRUE)

setwd("/Users/Tony/2020/Study/MA5810 Data Mining/Assessment/A3 Anomaly Detection")
Stamps = read.table("Stamps_withoutdupl_09.csv", header=FALSE, sep=",", dec=".")

PB_predictors = Stamps[,1:9] # 9 predictors, V1-V9
PB_class = Stamps[,10] # Class Labels V10
PB_class = ifelse(PB_class == 'no', 0,1) # Inliers (class "no") = 0, Outliers (class "yes") = 1
```

R Packages

```
# uncomment if system needs packages to be installed
# install.packages("wordspace")
# install.packages("plot3D")
# install.packages("pander")
```

R Libraries

```
library(tidyverse)
library(dplyr)
library(ggplot2)
library(reshape2)
library(dbscan)
library(plot3D)
library(plotly)
library(ROCR)
library(class)
library(pander)
```

Set Figure Margins

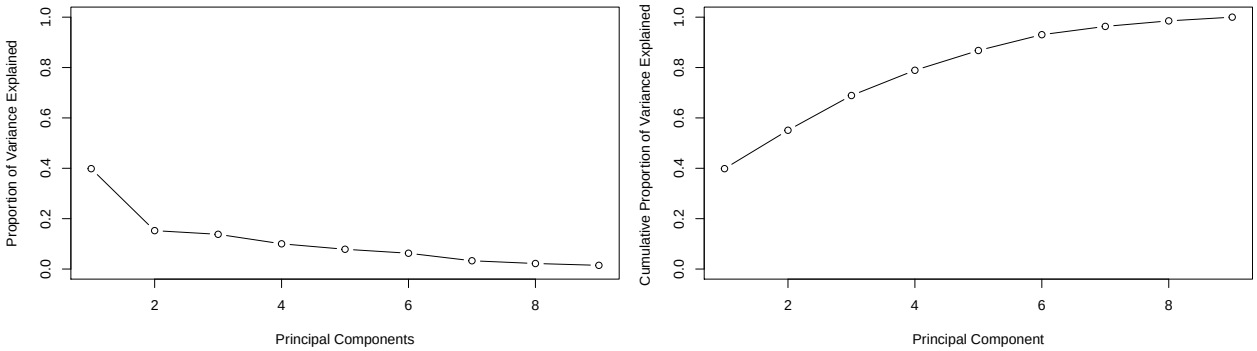
```
par(mar = c(1, 1, 1, 1))
```

Activity 1: Principal Component Analysis (PCA)

Question 1: Proportion of variance explained for 9 principal components (PC's).

```
# Perform PCA
PCA = prcomp(PB_predictors, scale = TRUE) # scale so variables have standard deviation = 1

# Variance explained by each PC
variance = PCA$sdev^2

# Proportion of variance explained (PVE) by each PC: divide variance by total variance
```

```
PVE = variance/sum(variance)

# Plot both plots together
par(mfrow=c(1,2), mar=c(4,4,4,1), oma=c(0.5,0.5,0.5,0))

# PC's and Proportion of Variance Explained
plot(PVE, xlab = "Principal Components",
     ylab = "Proportion of Variance Explained",
     ylim = c(0,1), type = 'b')

# Cumulative Sum Proportion of Variance Explained by PC's
plot(cumsum(PVE), xlab = " Principal Component ",
     ylab = "Cumulative Proportion of Variance Explained ",
     ylim = c(0,1), type = 'b')
```




```
# Proportion of variance explained (PVE) by each PC
PVE
```

```
## [1] 0.39850879 0.15249102 0.13791734 0.10016372 0.07854404 0.06277860
## [7] 0.03288279 0.02197946 0.01473424
```

```
# Cumulative sum PVE of all PC's
cumsum(PVE)
```

```
## [1] 0.3985088 0.5509998 0.6889171 0.7890809 0.8676249 0.9304035 0.9632863
## [8] 0.9852658 1.0000000
```

(a) How many components are needed to explain 90% of the total variance?

The first 6 components explain 93% of the total variance

(b) How much variance is explained by the first 3 PC's?

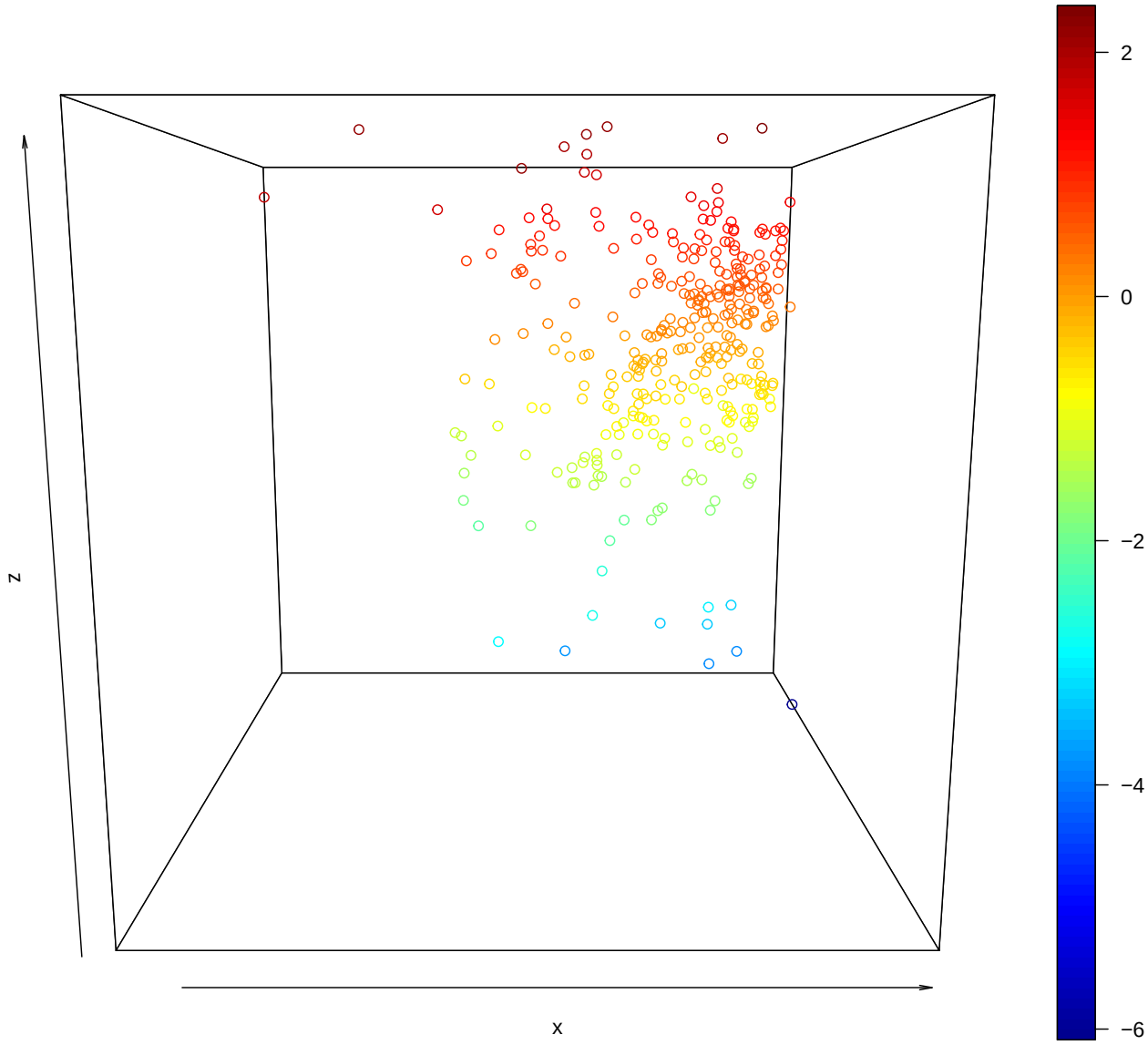
The first 3 components explain 68.9% of the total variance

Question 2: Plot a 3D scatterplot represented by the firs 3 PC's. Use class label to plot inliers (black) and outliers (red). Take at least 3 perspectives of the 3D plot. Would the outliers be easy to detect in an unsupervised problem, assuming that the 3D visualisation via PCA is a reasonable representation of the data in full space? What about in a supervised way?

```
# Create a new dataframe composing of the first 3 PC's with class label
PC123 = as.data.frame(PCA$x[,1:3])

PC_scatter_data = PC123 %>%
  cbind.data.frame(class = as.factor(PB_class))
```

```
# Unlabeled/Unsupervised 3D Scatterplot
fig1 = scatter3D(x = PC_scatter_data$PC1, y = PC_scatter_data$PC2,
          z = PC_scatter_data$PC3, phi = 10, theta = 0,
          groups = PC_scatter_data$class,
          main="Figure 1.1: Unlabelled 3D Scatterplot")
```

Figure 1.1: Unlabelled 3D Scatterplot




```
par(mfrow=c(2,2), mar=c(4,4,4,1), oma=c(0.5,0.5,0.5,0))
# Supervised (with Class Label) to Highlight labelled outliers (red) and inliers (black)
# View 1
fig2 = scatter3D(x = PC_scatter_data$PC1, y = PC_scatter_data$PC2,
          z = PC_scatter_data$PC3, phi = 10, theta = 0,
          colvar = as.numeric(PC_scatter_data$class),
          col = c("black", "red"), main="Figure 1.2: Labelled View 1")

# View 2
```

```r
fig3 = scatter3D(x = PC_scatter_data$PC1, y = PC_scatter_data$PC2,
          z = PC_scatter_data$PC3, phi = -40, theta = 0,
          colvar = as.numeric(PC_scatter_data$class),
          col = c("black", "red"), main="Figure 1.3: Labelled View 2")

# View 3
fig4 = scatter3D(x = PC_scatter_data$PC1, y = PC_scatter_data$PC2,
          z = PC_scatter_data$PC3, phi = -80, theta = 0,
          colvar = as.numeric(PC_scatter_data$class),
          col = c("black", "red"), main="Figure 1.4: Labelled View 3")

# View 4
fig5 = scatter3D(x = PC_scatter_data$PC1, y = PC_scatter_data$PC2,
          z = PC_scatter_data$PC3, phi = -160, theta = 0,
          colvar = as.numeric(PC_scatter_data$class),
          col = c("black", "red"), main="Figure 1.5: Labelled View 4")
```

Figure 1.2: Labelled View 1

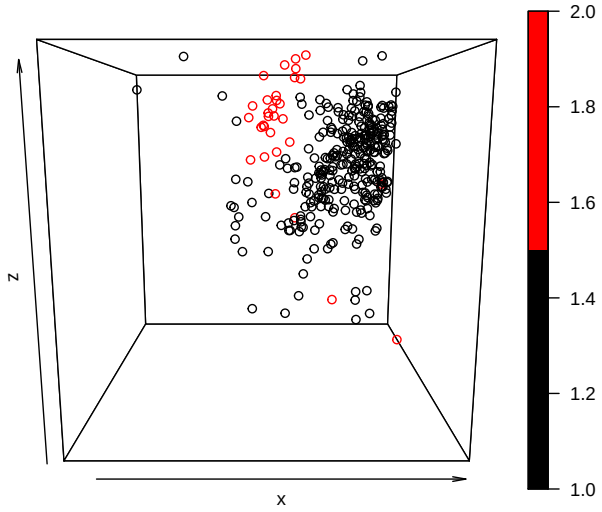


Figure 1.3: Labelled View 2

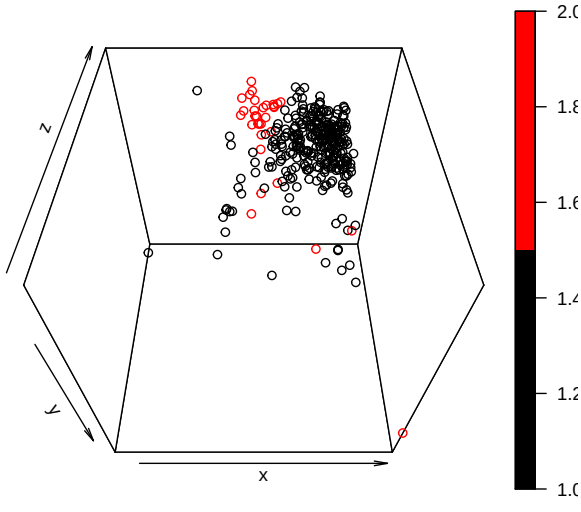


Figure 1.4: Labelled View 3

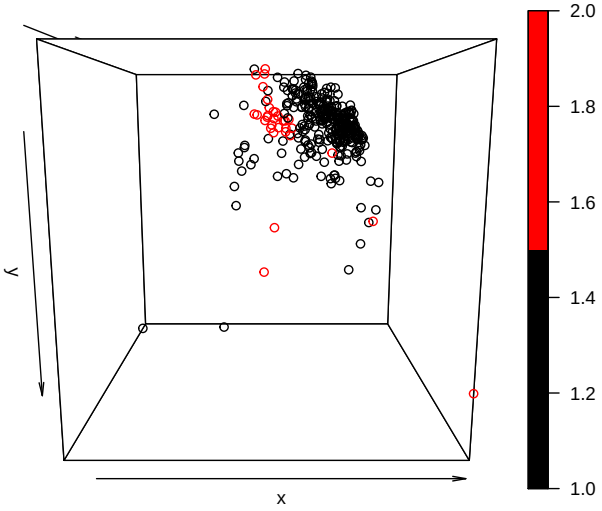


Figure 1.5: Labelled View 4

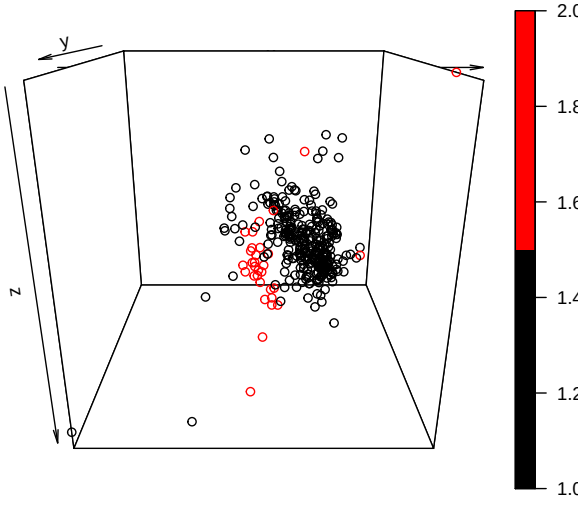
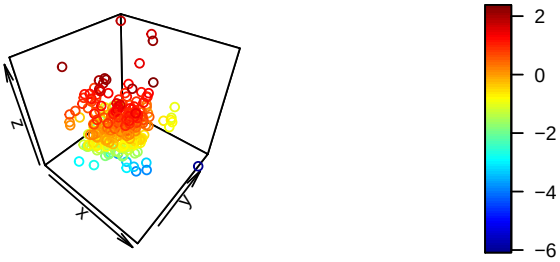


Figure 1.1 above shows a 3D scatterplot of the first three PC's without the class label. From initial visual inspection, a majority of observations are seen to cluster together in close proximity. This would lead one to believe that outliers would be easily detectable as these are assumed to fall outside the major cluster. As such, from an unsupervised perspective, outlier detection may appear to be easy enough when observing Figure 1.1, especially from different angles. However, when the class label is used to identify the labelled outliers (red) and inliers (black), a different observation is made. Figures 1.2 to 1.5 show differing angles of the same data, highlighting the labelled outliers as red and inliers as black, showing that some inliers are found to deviate the primary cluster, whilst a cluster of labelled outliers are situated adjacent to the main cluster of inliers. As such, the supervised method shows a very different picture that highlights the challenge in identifying outliers in a 3D visualisation. At first one may assume that 3D visualisation of the data via PCA is a reasonable representation of the data, but using a supervised method to label the outliers tells a different story - one that indicates that proximity to a primary cluster does not necessarily indicate whether or not an observation is an outlier.

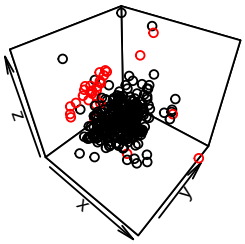
Activity 2: Unsupervised Outlier Detection using kNN

```
par(mfrow=c(4,2), mar=c(4,4,4,1), oma=c(0.5,0.5,0.5,0))

k_value = c(3, 5, 10, 20, 25, 30, 50, 100)

# Create a for loop to iterate over each k_value and produce plots for
n = length(k_value)
for (var in 1:n){
  knn_outlier = kNNdist(x = PB_predictors, k = k_value[var])

top_n = 31 # create top31 variable
knn_outlier_rank = order(x = knn_outlier, decreasing = TRUE) # sort by descending order

# Join kNN rank and score together
knn_output = data.frame(ID = knn_outlier_rank, score = knn_outlier[knn_outlier_rank])

# Join PC123, class label, kNN rank number, and kNN score into one table
PC_3d_data = cbind.data.frame(PC_scatter_data, knn_output)

# Top 31 outliers
knn_top31 = head(knn_output, top_n)

# Join top_n with PC data
pc_knn_ranked = merge(PC_3d_data, knn_top31, by = "ID", all.x = TRUE)

# The resulting dataframe contains first 3 PC's, class label, kNN score,
#   kNN rank ID, and a top 31 results which are marked as NA for inliers (black)

fig6 = scatter3D(x = pc_knn_ranked$PC1, y = pc_knn_ranked$PC2,
                 z = pc_knn_ranked$PC3,
                 main = paste("kNN Outlier Scores K-value:", k_value[var]))

fig7 = scatter3D(x = pc_knn_ranked$PC1, y = pc_knn_ranked$PC2,
                 z = pc_knn_ranked$PC3,
                 col = "red", colvar = pc_knn_ranked$score.y, NAcol = "black",
                 main = paste(" Class Labelled K-value:", k_value[var]))
}
```

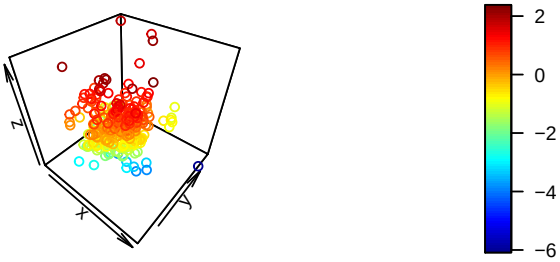
kNN Outlier Scores K−value: 3



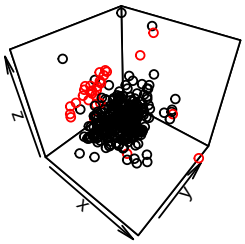
Class Labelled K−value: 3



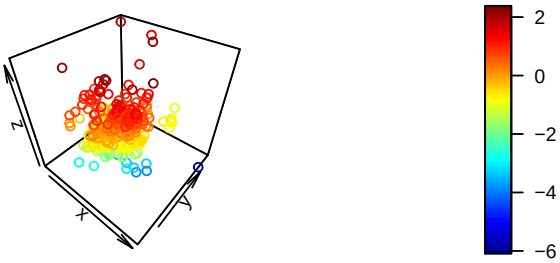
kNN Outlier Scores K−value: 5



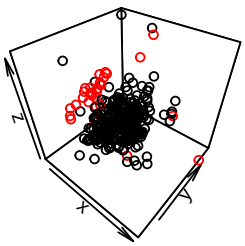
Class Labelled K−value: 5



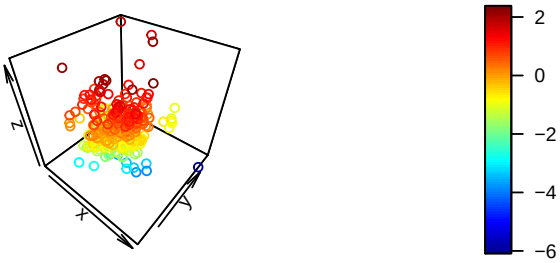
kNN Outlier Scores K−value: 10



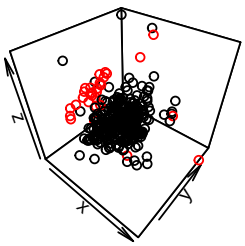
Class Labelled K−value: 10



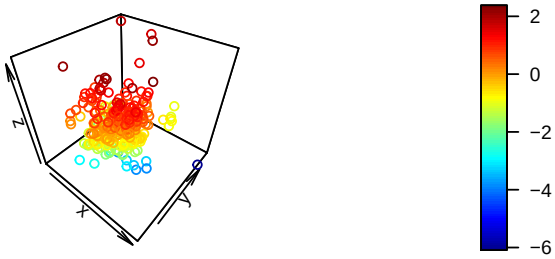
kNN Outlier Scores K−value: 20



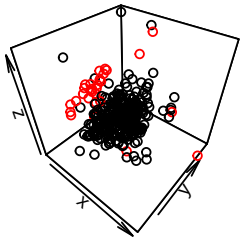
Class Labelled K−value: 20



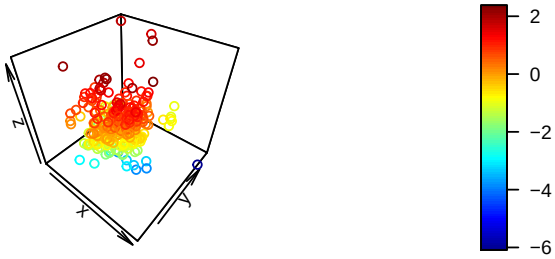
kNN Outlier Scores K−value: 25



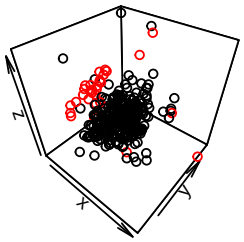
Class Labelled K−value: 25



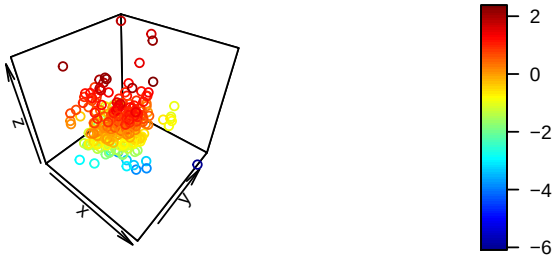
kNN Outlier Scores K−value: 30



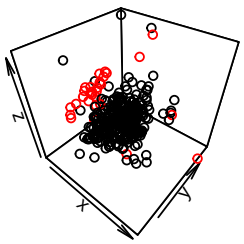
Class Labelled K−value: 30



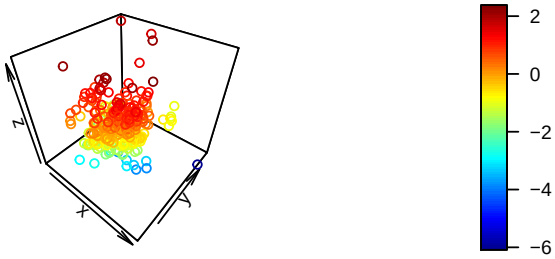
kNN Outlier Scores K−value: 50



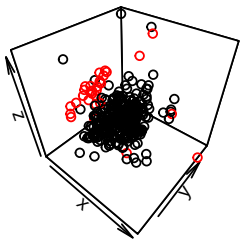
Class Labelled K−value: 50



kNN Outlier Scores K−value: 100



Class Labelled K−value: 100



Do these plots give you any insight on the values of k that look more or less appropriate from an unsupervised perspective (ignoring class labels)?

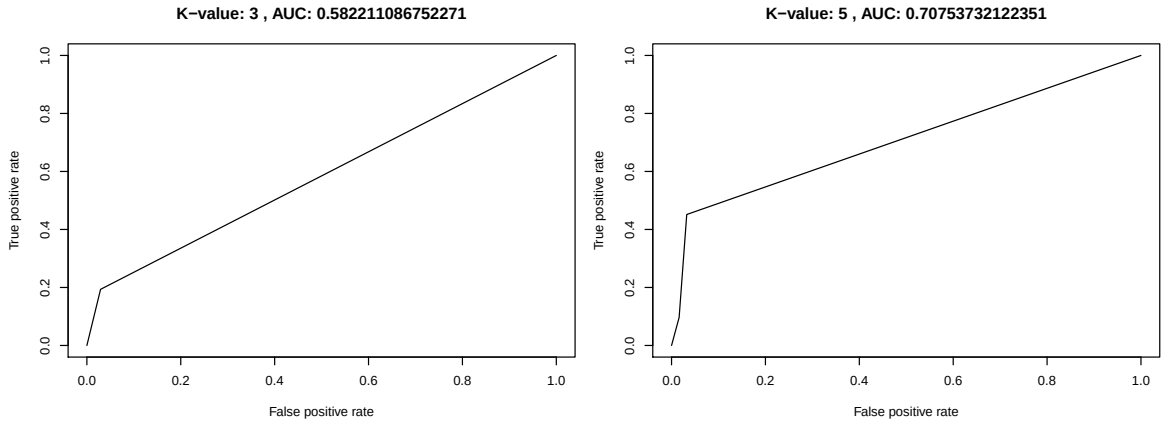
Micenkova et al. (2012) first reported the use of the stamps dataset, using supervised support vector machines to perform the same task at hand. These authors achieved ~95% specificity and sensitivity, but Campos et al. (2016) also achieved high classification performance (ROC-AUC score ~0.90) using unsupervised kNN. In the current study, k-values of 3, 5, 10, 20, 25, 30, 50, and 100 were used to produce 3D scatterplots of the first 3 principal components that describes ~68.9% of the total variance. Overall, no observable differences are detectable comparing scatterplots with differing k-values. Upon close inspection of the unlabelled scatterplots, colour ranges of dark blue and dark red indicate potential outliers. While outliers are typically thought of as objects at great distances from primary clusters, these plots from one specific viewpoint show that some outliers may be present in close proximity to the main cluster (Zhang et al. 2009). This is consistent with what is seen in their labelled counterparts, which show red objects (outliers) appearing in close proximity to the main cluster. As such, this can be accepted as an appropriate form of exploratory cluster analysis from an unsupervised perspective.

Top-n kNN outlier detection therefore ranks the n objects with highest values of k-distance as outliers (Zhang et al., 2009). In scattered datasets, top-n kNN can be ineffective, because if k is greater than the cardinality of one primary cluster, some objects in another cluster become neighbours of the objects in the former cluster (Zhang et al., 2009). Therefore, the k-distance of an observation can be larger than genuine outliers (Zhang et al., 2009). Hautamaki, et al. (2004) reports that smaller datasets may be prone to this issue.

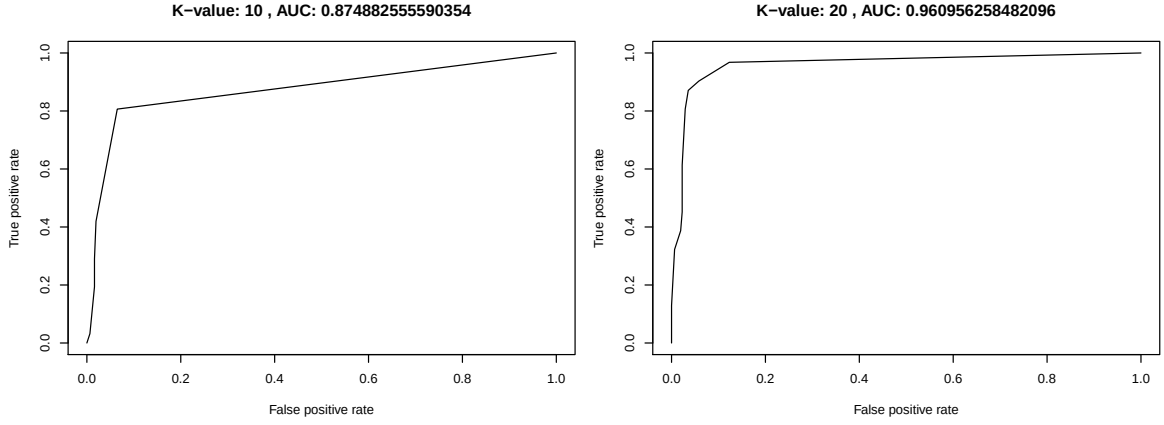
Activity 3: Supervised Anomaly Detection

Question 1: Produce an AUC-ROC in a LOOCV Scheme for each k

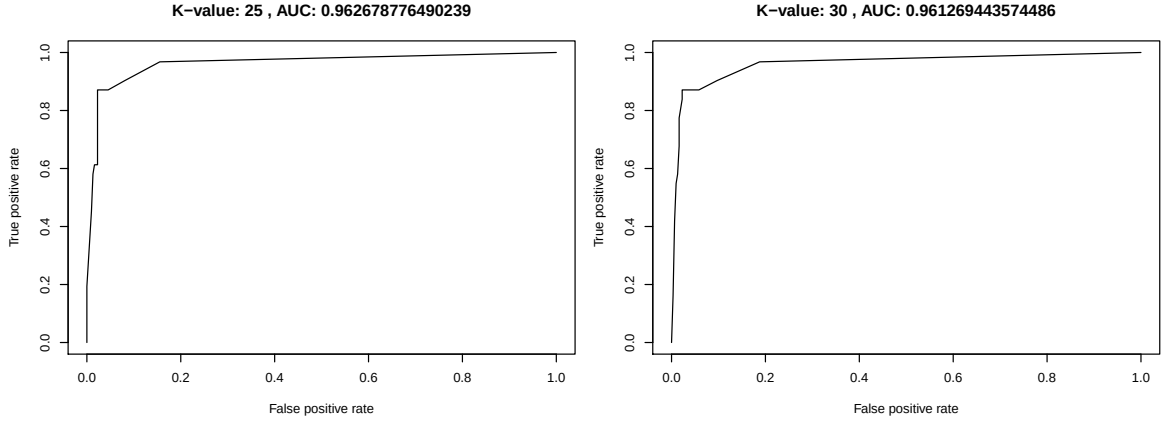
```
par(mfrow=c(4,2), mar=c(4,4,4,1), oma=c(0.5,0.5,0.5,0))

# ROC Plot function to calculate Area Under the Curve (AUC)
# (Stack Overflow, 2017)
rocplot_fun = function(pred_vals, truth){
  predict = prediction(pred_vals, truth)
  ROC = performance(predict, "tpr", "fpr")
  AUC = performance(predict, measure = "auc") # use "AUC" as measure for prediction
  AUC = AUC@y.values[[1]]
  # Plot ROC curve with title
  plot(ROC, main = paste("K-value:", k_value[var], ", AUC:", AUC))
  return(AUC) # Return the Area Under the Curve ROC
}

for (var in 1:n){
  # kNN cross-validation classification from training subset
  pred_class = knn.cv(train = PB_predictors,
                      cl = PB_class, k = k_value[var], prob = TRUE)
  pred_prob <- attr(pred_class, "prob")
  # Ensure probabilities are for class "+"
  pred_prob = ifelse(pred_class == '+', pred_prob, 1 - pred_prob)
  AUC = rocplot_fun(pred = pred_prob, truth = PB_class)
}
```


K−value: 3 , AUC: 0.582211086752271
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

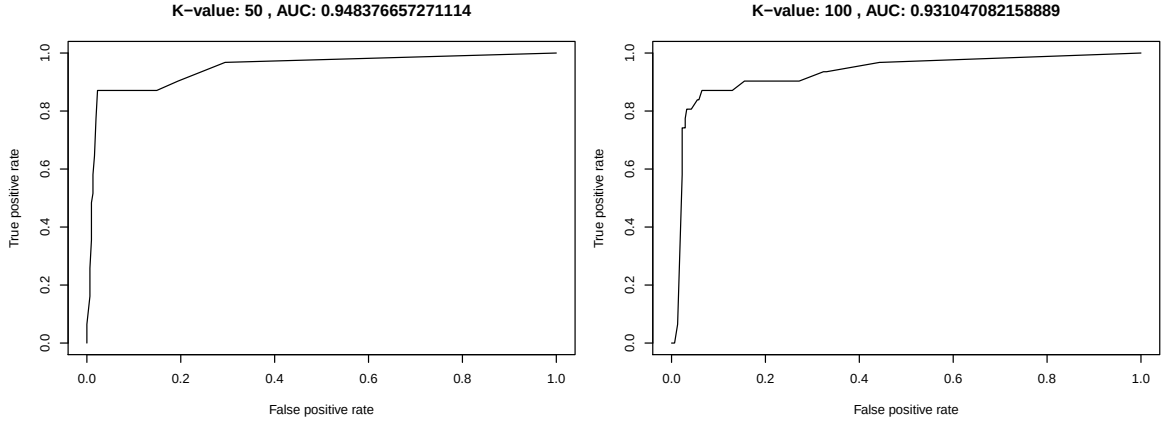
K−value: 5 , AUC: 0.70753732122351
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


K−value: 10 , AUC: 0.874882555590354
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

K−value: 20 , AUC: 0.960956258482096
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


K−value: 25 , AUC: 0.962678776490239
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

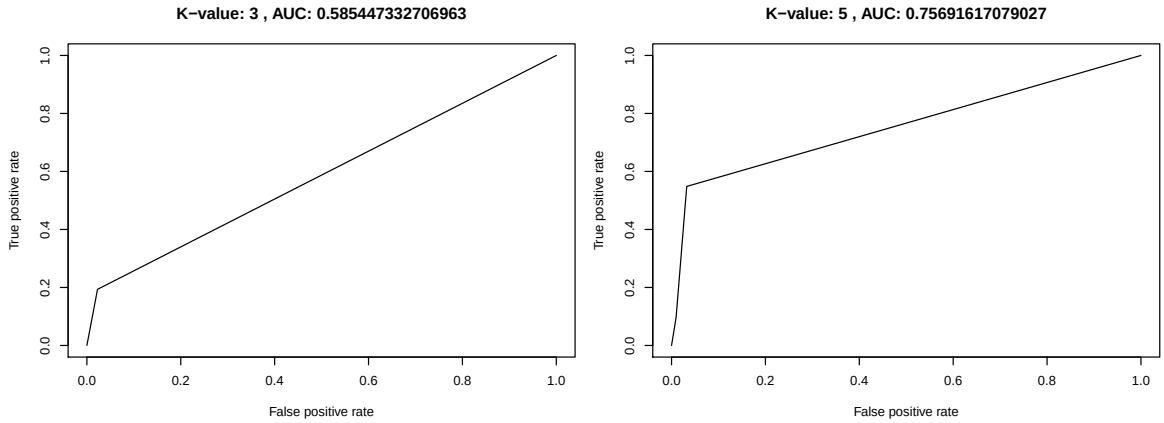
K−value: 30 , AUC: 0.961269443574486
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


K−value: 50 , AUC: 0.948376657271114
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

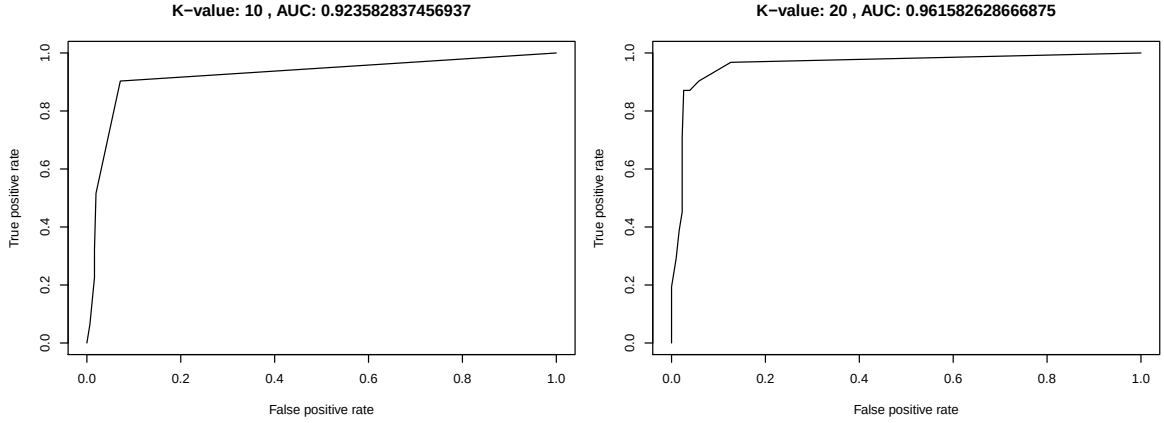
K−value: 100 , AUC: 0.931047082158889
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

Queston 2: Compare kNN performance of supervised (classification) versus unsupervised (clustering) methods with same k values. Rescale kNN outlier scores (from activity 2) into [0,1] interval, so they are interpreted as probabilities to then be compared with class labels (ground truth) and compute AUC-ROC curve.

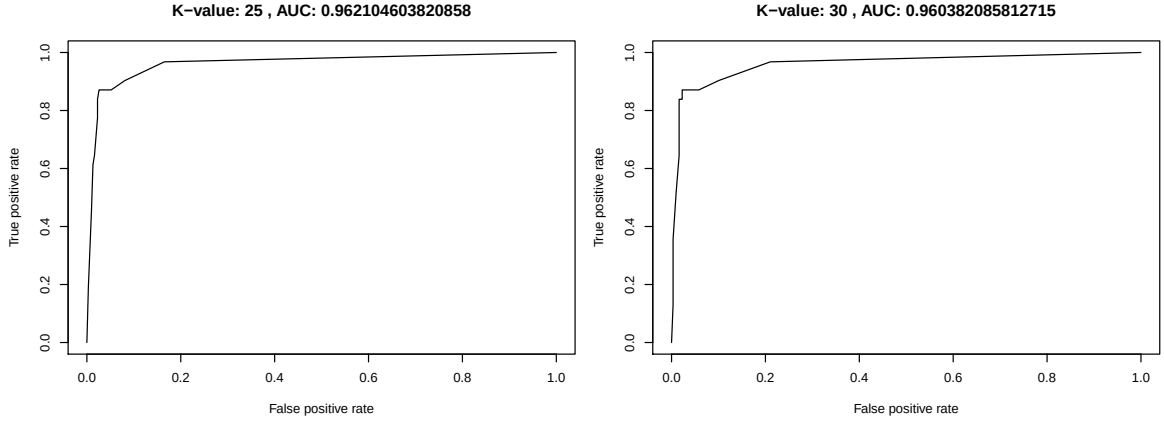
```
par(mfrow=c(4,2), mar=c(4,4,4,1), oma=c(0.5,0.5,0.5,0))

# Function for normalisation (Stack Overflow, 2019)
normalise_fun <- function(x) {
    return ((x - min(x)) / (max(x) - min(x)))
}

# Create a new data frame containing the normalised data
PB_predictors_norm = as.data.frame(lapply(PB_predictors, normalise_fun))

# k_value and n (length of k_value) are taken from loop defined above in activity 2
for (var in 1:n){ # Loop through each element
  # kNN cross-validation classification from training subset
  Pred_class = knn.cv(train = PB_predictors_norm,
                      cl = PB_class, k = k_value[var], prob = TRUE)

  # Get Prob attribute from Pred_Class
  Pred_prob = attr(Pred_class, "prob")
  # Make sure probabilities are for class "+"
  Pred_prob = ifelse(Pred_class == '+', Pred_prob, 1 - Pred_prob)

  # Calculate AUC value
  AUC = rocplot_fun(pred = Pred_prob, truth = PB_class)

}
```


K−value: 3 , AUC: 0.585447332706963
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

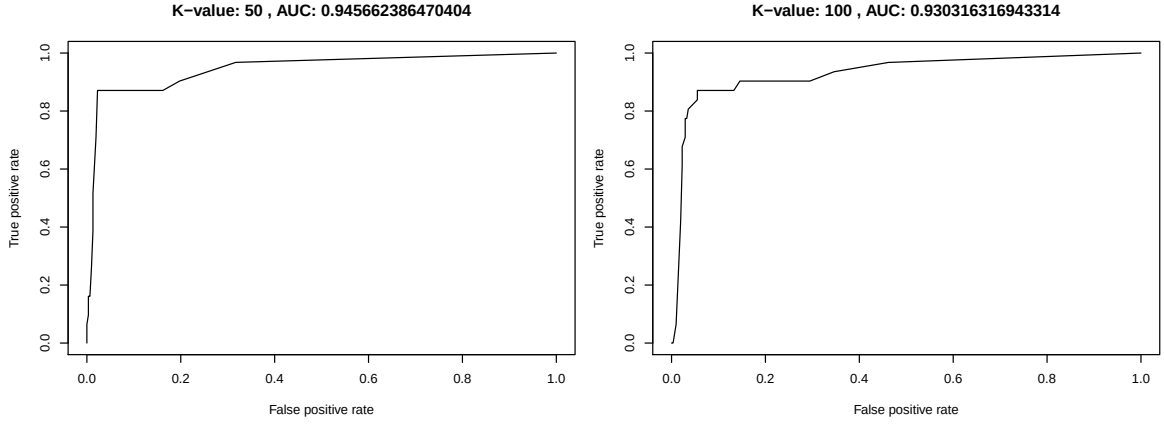
K−value: 5 , AUC: 0.75691617079027
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


K−value: 10 , AUC: 0.923582837456937
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

K−value: 20 , AUC: 0.961582628666875
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


K−value: 25 , AUC: 0.962104603820858
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

K−value: 30 , AUC: 0.960382085812715
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


K−value: 50 , AUC: 0.945662386470404
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

K−value: 100 , AUC: 0.930316316943314
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

Compare, for each k value, the AUC-ROC of supervised vs. unsupervised kNN classifiers. Since the supervised method uses the class label, what can you conclude considering there are applications where class labels are not available?

The assessment in performance of unsupervised (clustering) versus supervised (classification) kNN outlier detection methods rely on visual, subjective analysis of 3D scatterplots and ROC-AUC calculations, respectively. In Activity 2, it was observed that unsupervised kNN showed no observable difference for increasing k-values in terms of outlier scoring, and therefore colour-coding. However, application of the class label (ground truth) revealed labelled outlier objects that were located in close proximity to labelled inliers, which was not clearly observable without the class label in the initial outlier scoring plots. Compared to supervised learning, where the use of an ROC-AUC value helps measure model performance, unsupervised kNN protocols can be misleading with regards to model performance. The similarity between both supervised and unsupervised kNN outlier detection is that very little difference is observed between differing k-values, going from k=20 to k=50. Outside of these ranges, supervised kNN show some small differences.

The ROC-AUC value, ranging from 0 (worst) to 1 (best), is the most popular evaluation measure on unsupervised outlier detection (Campos et al., 2016). It can be described as the average recall at n (true positive rate over the n top-ranked objects), with n taken over the ranks of all inlier objects (Campos et al., 2016). Campost et al. (2016) reports that while normalisation of data significantly impacts cluster analysis, yet is rarely discussed in the literature.

The ROC-AUC results are summarised below in Table 3.1. A peak classification performance score of 0.9621 was achieved at k=25 using supervised kNN on the normalised stamps dataset. For the same k value, kNN achieved a slightly higher ROC-AUC score of 0.9627 on the unnormalised data. However, for k-values of 3, 5, 10, and 20, the kNN outlier classifier performed much better on normalised data. Since the difference in performance is negligible, it can be concluded that unsupervised kNN outlier detection is a suitable and appropriate tool in applications of cluster analysis where the class labels are not available.

K-Value	Unnormalised Data	Normalised Data
3	0.5822111	0.5854473
5	0.7075373	0.7569162
10	0.8748826	0.9235828
20	0.9609563	0.9615826
25	0.9626788	0.9621046
30	0.9612694	0.9603821
50	0.9483767	0.9456624
100	0.9310471	0.9303163

Table 3.1: AUC-ROC results of supervised kNN classification at varying k-values for unnormalised and normalised stamps data.

References

Campos, G.O., Zimek, A., Sander, J., Campello, R.J.G.B., Micenková, B., Schubert, E., Assent, I., and Houle, M.E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery, 30(4), 891-927. doi:10.1007/s10618-015-0444-8

Hautamaki, V., Karkkainen, I., & Franti, P. (2004). Outlier detection using k-nearest neighbour graph. doi:10.1109/ICPR.2004.1334558

Micenkova B., van Beusekom J., Shafait F. (2012). Stamp verification for automated document authentication. In: 5th International workshop on computational forensics.

Stack Overflow. (2017). How to compute AUC with ROCR package. Retrieved from https://stackoverflow.com/questions/41523761/how-to-compute-auc-with-rocr-package

Stack Overflow. (2019). min max scaling/normalization in r for train and test data. Retrieved from https://stackoverflow.com/questions/44050028/min-max-scaling-normalization-in-r-for-train-and-test-data

Zhang, K., Hutter, M., & Jin, H. (2009). A new local distance-based outlier detection approach for scattered real-world data. LNAI(5476), 813-822.