Cancer Detection Clustering

Yufei Quan

Introduction

The objective of this analysis is to assist in utilizing personal health data as a diagnostic tool for early cancer detection. This capstone project focuses on implementing unsupervised learning techniques to identify outliers and clusters in patient health data, hypothesizing that such deviations may correlate with cancer diagnoses. The analysis integrates techniques including outlier detection, clustering, and dimensionality reduction to build a robust framework. This framework will be validated against labeled data, leveraging the sensitivity metric to minimize false negatives as well as precision, both of which are critical in cancer detection scenarios.

Here are the following clustering techniques explored:

- K Means
- Agnes
- Diana
- DBSCAN
- K Nearest Neighbors (KNN)
- Isolation Forest

In this analysis, parameters were mostly left to natural constants and methodologies. This way, we can prevent overfitting and loss of external validity. The sample itself was too small to be split effectively into training and testing so a conservative approach was taken. This way, external validity can be kept while keeping analytical power at a maximum.

Although high accuracy is important, an emphasis was placed on reducing false negatives and overall false diagnoses. This is done to reduce harmful diagnoses in favor of false positives. Therefore, many clustering techniques have been adjusted to achieve this goal.

```r
suppressPackageStartupMessages({
  library(dplyr)
  library(ggplot2)
  library(ggbiplot)
  library(factoextra)
  library(caret)
  library(fastDummies)
  library(knitr)
  library(tidyr)
  library(dbscan)
  library(isotree)
  library(cluster)
  library(gridExtra)
  library(cowplot)
  library(FNN)
  library(solitude)
  library(outliers)
  library(EnvStats)
  library(car)
  library(kableExtra)
  library(reactable)
  library(reactablefmtr)
})
```

Load Data

Data: The data is split into 31 columns with 30 being the data itself and the last being a label for outcomes. The outcome variable is binary, meaning 1 indicates the patient has cancer and 0 indicates the patient does not have cancer. When we load the data for the first time, we can see there are 378 observations. Given how many dimensions this dataset has, we can consider using PCA to reduce dimensionality. This will help with visualizing data and improve interpretability as well as enhancing clustering which we will run later.

```
dataraw <- read.csv("wbc.csv")

features <- as.data.frame(scale(dataraw %>% select(-y)))
labels <- dataraw$y
dataraw$y <- NULL
str(dataraw)
```

```
## 'data.frame':    378 obs. of  30 variables:
##  $ X1 : num  0.3104 0.2887 0.1194 0.2863 0.0575 ...
##  $ X2 : num  0.1573 0.2029 0.0923 0.2946 0.2411 ...
##  $ X3 : num  0.3018 0.2891 0.1144 0.2683 0.0547 ...
##  $ X4 : num  0.1793 0.1597 0.0553 0.1613 0.0248 ...
##  $ X5 : num  0.408 0.495 0.449 0.336 0.301 ...
##  $ X6 : num  0.1899 0.3301 0.1397 0.0561 0.1228 ...
##  $ X7 : num  0.1561 0.107 0.0693 0.06 0.0372 ...
##  $ X8 : num  0.2376 0.1546 0.1032 0.1453 0.0294 ...
##  $ X9 : num  0.417 0.458 0.381 0.206 0.358 ...
##  $ X10: num  0.162 0.382 0.402 0.183 0.317 ...
##  $ X11: num  0.0574 0.0267 0.06 0.0262 0.0162 ...
##  $ X12: num  0.0947 0.0856 0.1363 0.438 0.1318 ...
##  $ X13: num  0.0613 0.0295 0.0543 0.0195 0.0159 ...
##  $ X14: num  0.0313 0.0147 0.01662 0.01374 0.00262 ...
##  $ X15: num  0.2294 0.081 0.2683 0.0897 0.2466 ...
##  $ X16: num  0.0927 0.1256 0.0906 0.0199 0.1067 ...
##  $ X17: num  0.0603 0.0429 0.0501 0.0339 0.0401 ...
##  $ X18: num  0.249 0.123 0.269 0.22 0.112 ...
##  $ X19: num  0.168 0.125 0.174 0.265 0.251 ...
##  $ X20: num  0.0485 0.0529 0.0716 0.0305 0.0583 ...
##  $ X21: num  0.2554 0.2337 0.0818 0.191 0.0368 ...
##  $ X22: num  0.193 0.226 0.097 0.288 0.265 ...
##  $ X23: num  0.2455 0.2275 0.0733 0.1696 0.0341 ...
##  $ X24: num  0.1293 0.1094 0.0319 0.0887 0.014 ...
##  $ X25: num  0.481 0.396 0.404 0.171 0.387 ...
##  $ X26: num  0.1455 0.2429 0.0849 0.0183 0.1052 ...
##  $ X27: num  0.1909 0.151 0.0708 0.0386 0.055 ...
##  $ X28: num  0.4426 0.2503 0.214 0.1723 0.0881 ...
##  $ X29: num  0.2783 0.3191 0.1745 0.0832 0.3036 ...
##  $ X30: num  0.1151 0.1757 0.1488 0.0436 0.125 ...
```

Below, we can run PCA to reduce dimensionality:

```r
pca <- prcomp(features, center = TRUE, scale. = TRUE)
summary(pca)
```

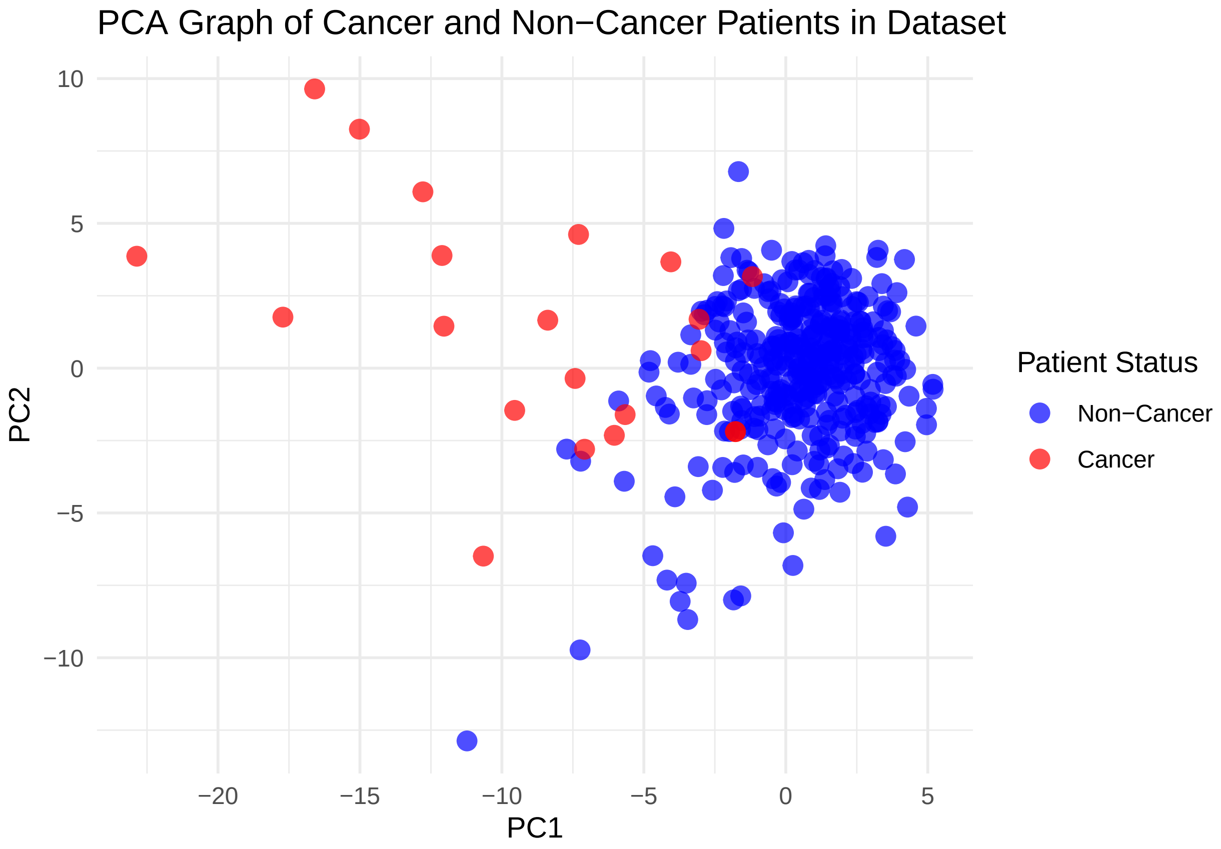
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.3582 2.6065 1.70096 1.48309 1.42571 1.09873 0.86315
## Proportion of Variance 0.3759 0.2265 0.09644 0.07332 0.06776 0.04024 0.02483
## Cumulative Proportion  0.3759 0.6024 0.69883 0.77215 0.83990 0.88014 0.90498
##                           PC8     PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.6929 0.66264 0.64145 0.5693 0.54596 0.45359 0.43020
## Proportion of Variance 0.0160 0.01464 0.01372 0.0108 0.00994 0.00686 0.00617
## Cumulative Proportion  0.9210 0.93561 0.94933 0.9601 0.97007 0.97693 0.98310
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.31743 0.31151 0.24159 0.20925 0.20575 0.19692 0.16504
## Proportion of Variance 0.00336 0.00323 0.00195 0.00146 0.00141 0.00129 0.00091
## Cumulative Proportion  0.98645 0.98969 0.99163 0.99309 0.99450 0.99580 0.99671
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16113 0.14315 0.12610 0.12035 0.10033 0.09392 0.04807
## Proportion of Variance 0.00087 0.00068 0.00053 0.00048 0.00034 0.00029 0.00008
## Cumulative Proportion  0.99757 0.99825 0.99878 0.99927 0.99960 0.99990 0.99997
##                           PC29    PC30
## Standard deviation     0.02568 0.01197
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

```r
top_components <- as.data.frame(pca$x[, 1:11])
colnames(top_components) <- paste0("PC", 1:11)

features <- top_components
```

If we follow the cumulative proportion in the third row, we can see that 7 components accounts for over 90% of variance and 11 components account for 96%. This means that the last 19 components or so make up the last 4%. In this analysis, we can keep the first 11 since that accounts for the vast majority of our predictive power and will make this more computationally efficient.

```r
pca_data <- data.frame(
  PC1 = features[, 1],  # Extract the first principal component
  PC2 = features[, 2],  # Extract the second principal component
  Label = as.factor(labels)  # Add labels (0 = Non-Cancer, 1 = Cancer)
)

cancer_plot <- ggplot(pca_data, aes(x = PC1, y = PC2, color = Label)) +
  geom_point(size = 3, alpha = 0.7) +  # Use dots for data points
  scale_color_manual(
    values = c("0" = "blue", "1" = "red"),  # Map 0 to blue (Non-Cancer), 1 to red (Cancer)
    labels = c("0" = "Non-Cancer", "1" = "Cancer")  # Update legend labels
  ) +
  labs(
    color = "Patient Status",
    title = "PCA Graph of Cancer and Non-Cancer Patients in Dataset",  # Add plot title
    x = "PC1",   # Label for x-axis
    y = "PC2"    # Label for y-axis
  ) +
  theme_minimal() +                # Use a minimal theme for cleaner visuals
  theme(legend.position = "right")  # Position the legend to the right
```

```
cancer_plot
```



From the PCA graph above, we can see that a lot of cancer patients are outliers and non cancer patients are grouped together in a cluster. The difficulty is that many cancer patients are also found close to non cancer patients and vice versa.

For clustering techniques, most were kept at 5 clusters. Predictions were made by either choosing 2 clusters with the highest cancer counts or clusters with over 25% cancer positivity rate. The union of these two criteria were chosen to be predicted positives. In addition to this, many clustering techniques were chosen by evaluating precision, sensitivity, and specificity. Although false negatives are the primary focus, we wanted to build a tool that is useful for all cases, rather than overfitting and conforming to a single metric. Therefore, the sums of these three metrics are often used to set parameters.

```
compute_metrics <- function(labels, predictions, method = c("clustering", "outlier"), positive_class = N
  method <- match.arg(method)
  cluster_proportion_df <- NULL

  if (method == "clustering") {
    if (is.null(positive_class)) {
      # Calculate proportions and counts for each cluster
      cluster_proportion <- tapply(labels, predictions, mean)
      cluster_counts <- tapply(labels, predictions, length)
      positive_counts <- tapply(labels, predictions, sum)

      cluster_proportion_df <- data.frame(
        Cluster = names(cluster_proportion),
        Total = cluster_counts,
```

```r
        Positive_Count = positive_counts,
        Proportion_Positive = cluster_proportion
      )

      cluster_proportion_df$Cluster <- as.numeric(as.character(cluster_proportion_df$Cluster))

      cluster_proportion_df <- cluster_proportion_df[order(-cluster_proportion_df$Positive_Count), ]
      top_2_clusters <- head(cluster_proportion_df$Cluster, 2)
      threshold_clusters <- cluster_proportion_df$Cluster[cluster_proportion_df$Proportion_Positive >= (

      positive_class <- unique(c(top_2_clusters, threshold_clusters))
    }

    predicted_class <- ifelse(predictions %in% positive_class, 1, 0)

  } else if (method == "outlier") {
    if (is.null(positive_class)) {
      stop("For outlier detection, 'positive_class' must be specified.")
    }
    predicted_class <- ifelse(predictions %in% positive_class, 1, 0)
  }

  cfmatrix <- confusionMatrix(
    data = factor(predicted_class),
    reference = factor(labels)
  )
  if (output) {
    print(cfmatrix)
  }

  sensitivity <- cfmatrix$byClass["Sensitivity"]
  precision <- cfmatrix$byClass["Precision"]
  specificity <- cfmatrix$byClass["Specificity"]

  if (!is.null(cluster_proportion_df)) {
    return(list(
      sensitivity = sensitivity,
      precision = precision,
      specificity = specificity,
      proportions = cluster_proportion_df
    ))
  } else {
    return(list(
      sensitivity = sensitivity,
      precision = precision,
      specificity = specificity
    ))
  }
}
```

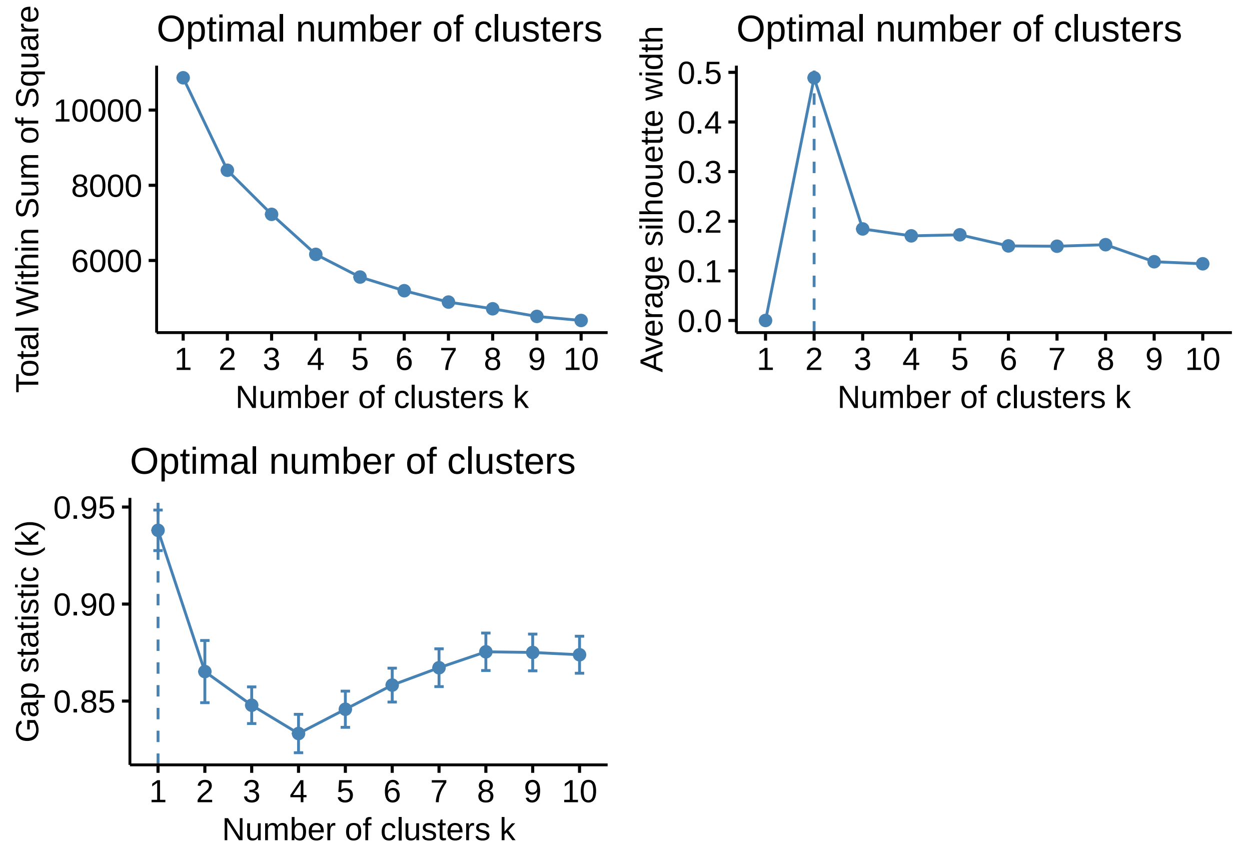
K Means Clustering

The first clustering technique is K Means. This method gathers data into k centroids and forms clusters around those points. This process continues until there are no more changes or the maximum number of iterations are reached. Then, we will select the cluster with the highest percent of cancer patients and use that as our prediction. The confusion matrix will be built using this assumption. Below is the implementation.

Determining Optimal K

The first task in K Means clustering is selecting the number of clusters. We can use elbow, silhouette, and gap stat graphs to determine what k should be.

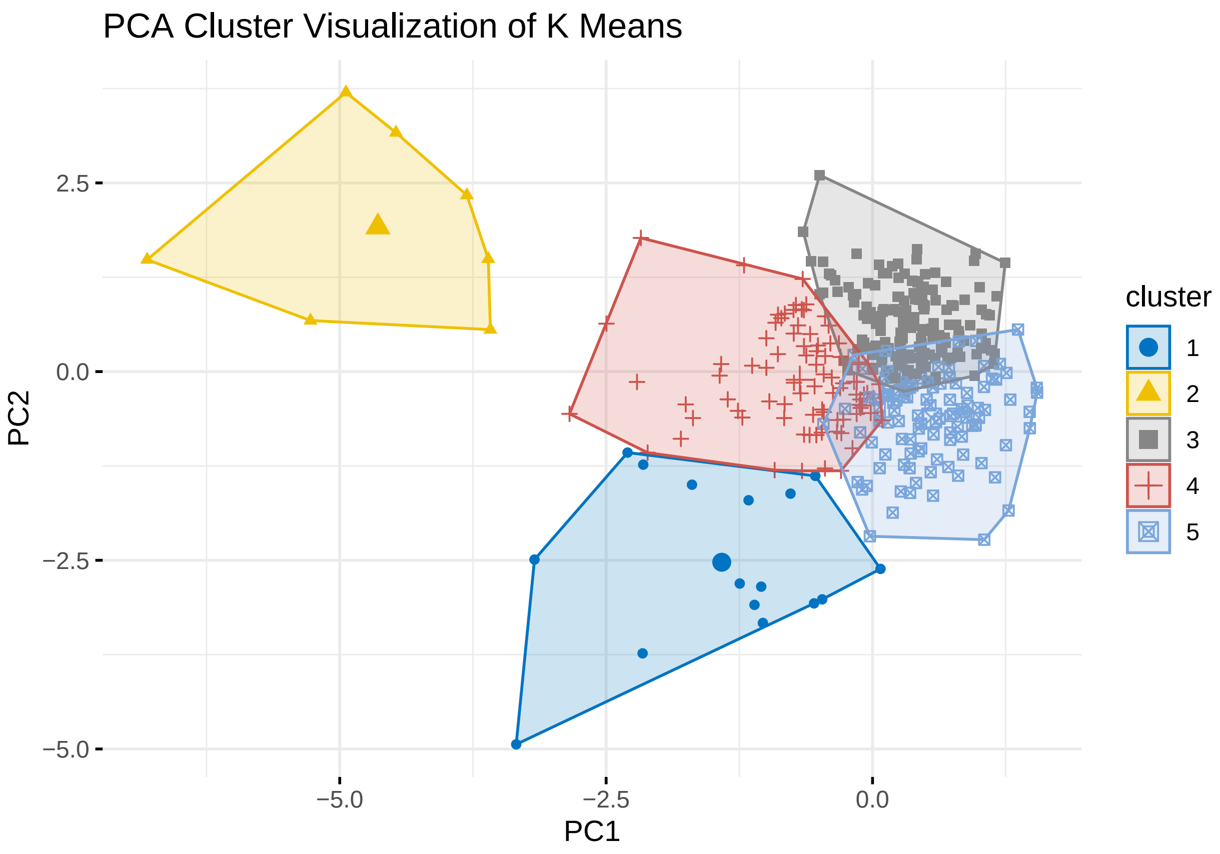
```r
wss <- fviz_nbclust(features, kmeans, method = "wss")
sil <- fviz_nbclust(features, kmeans, method = "silhouette")
gap <- fviz_nbclust(features, kmeans, method = "gap_stat")

grid.arrange(wss, sil, gap, nrow = 2, ncol = 2)
```



```r
k <- 5
```

From the graphs, we can see a wide range of candidates. Varying k, we find that 2 clusters works the best, providing a good prediction of cancer patients while keeping sensitivity at a reasonable level.

```r
set.seed(12345)
km <- kmeans(features, centers = k, nstart = 25)

kmeans_plot <- fviz_cluster(
  list(data = features[, 1:2], cluster = as.factor(km$cluster)),
  geom = "point",
  palette = "jco",  # Adjust colors
  ellipse.type = "convex",  # Add cluster ellipses
  ggtheme = theme_minimal(),
  main = "PCA Cluster Visualization of K Means"
)
kmeans_plot
```



From the cluster visualization above, we can see that cluster 2 (yellow cluster) captures a lot of the outliers who are all cancer patients while the main central cluster is divided into 3. In addition, the bottom most cluster comprises a mix of both cancer and non cancer patients.

Confusion Matrix and Results

```r
metrics_kmeans <- compute_metrics(
  labels = labels,
  predictions = km$cluster,
  method = "clustering"
)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   0   1
##          0 282   2
##          1  75  19
##
##                Accuracy : 0.7963
##                  95% CI : (0.7521, 0.8358)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2635
##
##  Mcnemar's Test P-Value : 2.303e-16
##
##             Sensitivity : 0.7899
##             Specificity : 0.9048
##          Pos Pred Value : 0.9930
##          Neg Pred Value : 0.2021
##              Prevalence : 0.9444
##          Detection Rate : 0.7460
##    Detection Prevalence : 0.7513
##       Balanced Accuracy : 0.8473
##
##        'Positive' Class : 0
##
```

```
kable(metrics_kmeans$proportions)
```

	Cluster	Total	Positive_Count	Proportion_Positive
4	4	87	12	0.1379310
2	2	7	7	1.0000000
1	1	17	1	0.0588235
3	3	159	1	0.0062893
5	5	108	0	0.0000000

```
sensitivity_kmeans <- metrics_kmeans$sensitivity
precision_kmeans <- metrics_kmeans$precision
specificity_kmeans <- metrics_kmeans$specificity

cat("KMeans Sensitivity:", sensitivity_kmeans, "\n")
```

```
## KMeans Sensitivity: 0.789916
```

```
cat("KMeans Precision:", precision_kmeans, "\n")
```

```
## KMeans Precision: 0.9929577
```

We can see that the model is fairly accurate and predicts the vast majority of patients who don’t have cancer. In this dataset, we only have 21 cancer patients out of 378 total observations. This is a very limited sample and K Means struggles to classify all of them correctly. 19 cancer patients were correctly classified while 2 were false negatives.

Agnes

AGNES, short for Agglomerative Nesting, is a hierarchical clustering method that builds clusters step-by-step from the bottom up. It starts by treating each data point as its own cluster and iteratively merges the closest clusters until all data points are grouped into a single cluster or a desired number of clusters is reached.

Choosing Linkage Method

One choice we have is the linkage method, governing how the distance between clusters are calculated. There are some popular choices such as average, complete, single, and ward.

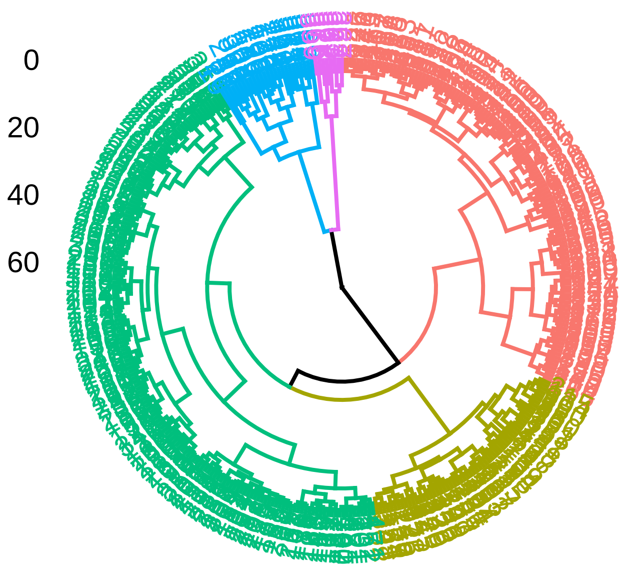
```r
average <- round(agnes(features, method = "average")$ac, 2)
complete <- round(agnes(features, method = "complete")$ac, 2)
single <- round(agnes(features, method = "single")$ac, 2)
ward <- round(agnes(features, method = "ward")$ac, 2)

wards <- agnes(features, method = "ward")
kable(data.frame(average, complete, single, ward))
```

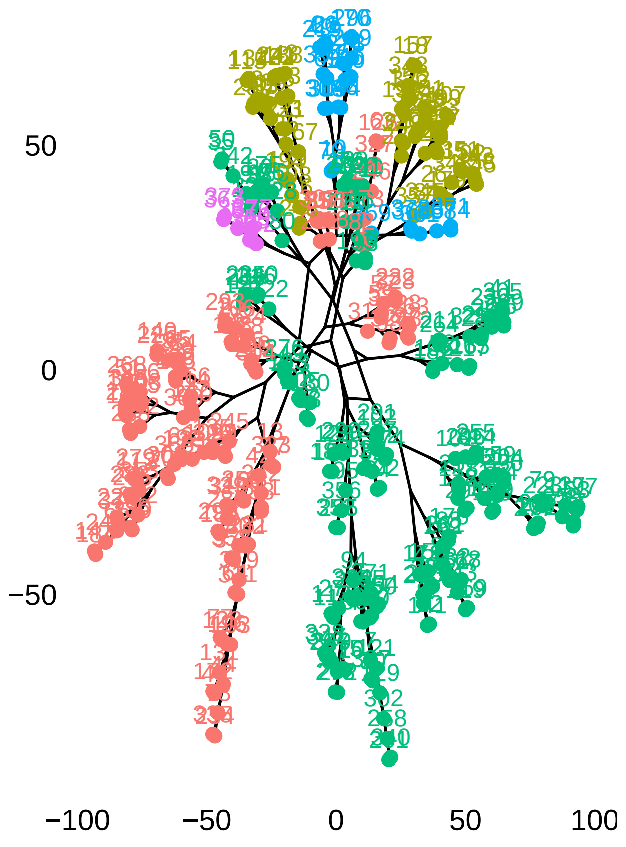
average	complete	single	ward
0.86	0.91	0.77	0.96

From the table, we see that ward is the best linkage method and we will use this for Agnes.

Cluster Dendrogram

```r
options(warn=-1)
k <- 5
agnes_clusters <- cutree(wards, k=k)
circular <- fviz_dend(wards, k=k, rect=TRUE, type = "circular")
branch <- fviz_dend(wards, k=k, rect=TRUE, type = "phylogenic")
grid.arrange(circular, branch, ncol=2)
```





From this dendrogram, we see that a lot of the data falls into one cluster and the rest into other smaller clusters. We will use these to build a confusion matrix with similar logic as k means. Other parameters such as distance, linkage method, k were experimented with and this was found to be the best. Each combination has a trade off of sensitivity, specificity, and precision but this method seems to strike a balance between all three.

```r
agnes_plot <- fviz_cluster(
  list(data = features[, 1:2], cluster = agnes_clusters),
  geom = "point",
  palette = "jco",  # Adjust colors
  ellipse.type = "convex",  # Add cluster ellipses
  ggtheme = theme_minimal(),
  main = "PCA  Visualization of Agnes Clusters"
)
```

Confusion Matrix

```r
metrics_agnes <- compute_metrics(
  labels = labels,
  predictions = agnes_clusters,
  method = "clustering"
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 338   6
##          1  19  15
##
##                Accuracy : 0.9339
##                  95% CI : (0.9039, 0.9567)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 0.8441
##
##                   Kappa : 0.5119
##
##  Mcnemar's Test P-Value : 0.0164
##
##             Sensitivity : 0.9468
##             Specificity : 0.7143
##          Pos Pred Value : 0.9826
##          Neg Pred Value : 0.4412
##              Prevalence : 0.9444
##          Detection Rate : 0.8942
##    Detection Prevalence : 0.9101
##       Balanced Accuracy : 0.8305
##
##        'Positive' Class : 0
##
```

```r
kable(metrics_agnes$proportions)
```

	Cluster	Total	Positive_Count	Proportion_Positive
5	5	8	8	1.0000000
4	4	26	7	0.2692308
1	1	120	5	0.0416667
3	3	164	1	0.0060976
2	2	60	0	0.0000000

```r
sensitivity_agnes <- metrics_agnes$sensitivity
precision_agnes <- metrics_agnes$precision
specificity_agnes <- metrics_agnes$specificity

cat("AGNES Sensitivity:", sensitivity_agnes, "\n")
```

```
## AGNES Sensitivity: 0.9467787
```

```r
cat("AGNES Precision:", precision_agnes, "\n")
```

```
## AGNES Precision: 0.9825581
```

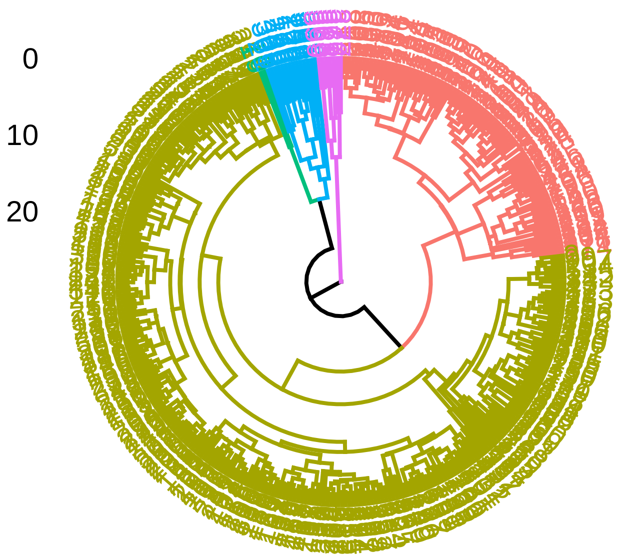
This confusion matrix summarizes the performance of a classification model. The accuracy of the model is 93.39%, indicating that the model correctly classified most instances. The sensitivity (recall) of 94.68% shows the model's ability to correctly identify true negatives, while the specificity of 71.43% reflects its performance in identifying true positives.

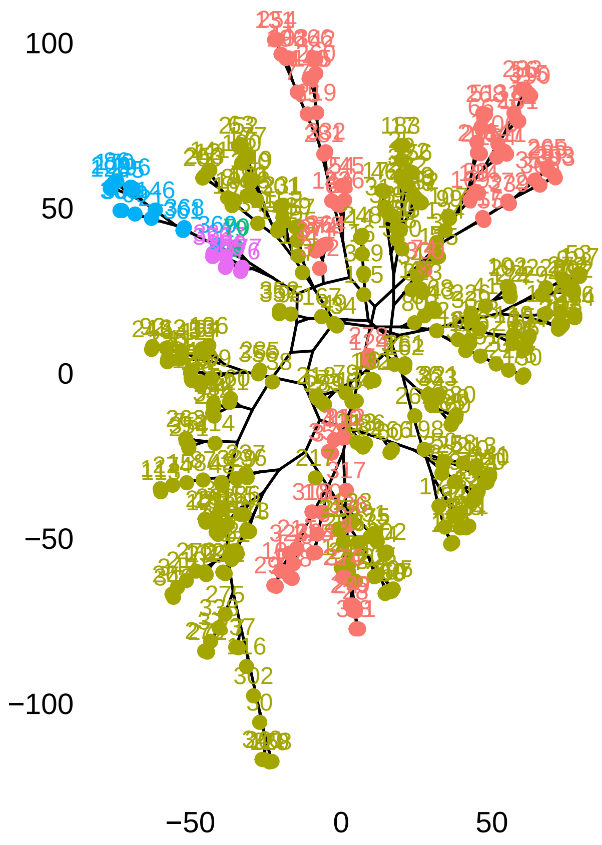
Diana

DIANA (Divisive Analysis Clustering) is a hierarchical clustering method that works in a top-down manner. It starts with all data points in a single cluster and iteratively splits them into smaller clusters based on dissimilarity, prioritizing the largest and most heterogeneous clusters first. This approach is particularly useful for identifying natural divisions in data, but it can be computationally expensive for large datasets.

Cluster Dendrogram

```r
diana <- diana(features)
k <- 5
diana_clusters <- cutree(diana, k=k)
circular <- fviz_dend(diana, k=k, rect=TRUE, type = "circular")
branch <- fviz_dend(diana, k=k, rect=TRUE, type = "phylogenic")
grid.arrange(circular, branch, ncol=2)
```





```
diana_plot <- fviz_cluster(
  list(data = features[, 1:2], cluster = diana_clusters),
  geom = "point",
  palette = "jco",  # Adjust colors
  ellipse.type = "convex",  # Add cluster ellipses
  ggtheme = theme_minimal(),
  main = "PCA Cluster Visualization of Diana Clusters"
)
```

From the dendrogram, we see most of the data falling into one cluster.

Confusion Matrix

```
metrics_diana <- compute_metrics(
  labels = labels,
  predictions = diana_clusters,
  method = "clustering"
)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 280   4
##          1  77  17
##
##                Accuracy : 0.7857
##                  95% CI : (0.7409, 0.826)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2253
##
##  Mcnemar's Test P-Value : 1.244e-15
##
##             Sensitivity : 0.7843
##             Specificity : 0.8095
##          Pos Pred Value : 0.9859
##          Neg Pred Value : 0.1809
##              Prevalence : 0.9444
##          Detection Rate : 0.7407
##    Detection Prevalence : 0.7513
##       Balanced Accuracy : 0.7969
##
##        'Positive' Class : 0
##
```

```
kable(metrics_diana$proportions)
```

	Cluster	Total	Positive_Count	Proportion_Positive
1	1	87	10	0.1149425
5	5	7	7	1.0000000
4	4	14	3	0.2142857
2	2	268	1	0.0037313
3	3	2	0	0.0000000

```
sensitivity_diana <- metrics_diana$sensitivity
precision_diana <- metrics_diana$precision
specificity_diana <- metrics_diana$specificity

cat("DIANA Sensitivity:", sensitivity_diana, "\n")
```

```
## DIANA Sensitivity: 0.7843137
```

```
cat("DIANA Precision:", precision_diana, "\n")
```

```
## DIANA Precision: 0.9859155
```

This confusion matrix shows a model with moderate accuracy (78.57%) and balanced sensitivity (78.43%) and specificity (80.95%), indicating that the model performs relatively well in identifying both true negatives (no cancer) and true positives (cancer). However, the positive predictive value (98.59%) suggests that when the model predicts no cancer, it is highly reliable. Conversely, the negative predictive value (18.09%) highlights its limitations in accurately predicting cancer cases, as a significant proportion of actual cancer cases are missed.

DBSCAN

DBSCAN is a density-based clustering algorithm that groups data points into clusters based on their proximity and density. It identifies dense regions of points separated by sparser regions, making it particularly effective for detecting irregularly shaped clusters and outliers. In this analysis, outliers will be considered to be cancer patients while those in clusters will be considered to be no cancer.

Find optimal eps and min points using grid search

There are 2 main parameters we can change in dbscan: epsilon and min points. We can set the define a range of eps and minpoint values and loop through each combination. For each combination, the score is calculated from the sum of precision, sensitivity, and specificity scores, since those are the ones we're most interested in. The combination that maximizes this score is kept as our eps and min points value for our analysis.

```r
options(warn = -1)
best_eps <- NULL
best_minpts <- NULL
best_score <- -Inf

eps_values <- seq(2.0, 8.0, by = 0.1)
minpts_values <- 15:25

for (eps in eps_values) {
  for (minpts in minpts_values) {

    dbscan_result <- dbscan(features, eps = eps, minPts = minpts)

    metrics_dbscan <- compute_metrics(
      labels = labels,
      predictions = dbscan_result$cluster,
      method = "outlier",
      positive_class = 0,
      output = FALSE
    )

    sensitivity <- metrics_dbscan$sensitivity
    precision <- metrics_dbscan$precision
    specificity <- metrics_dbscan$specificity
    score <- sensitivity + precision + specificity

    if (is.na(score)) {
      next  # Skip this iteration
    }

    if (score > best_score) {
      best_eps <- eps
      best_minpts <- minpts
      best_score <- score
    }
  }
}

cat("The best eps value is:", best_eps, "\n")
```

```
## The best eps value is: 3
```

```r
cat("The best minPts value is:", best_minpts, "\n")
```

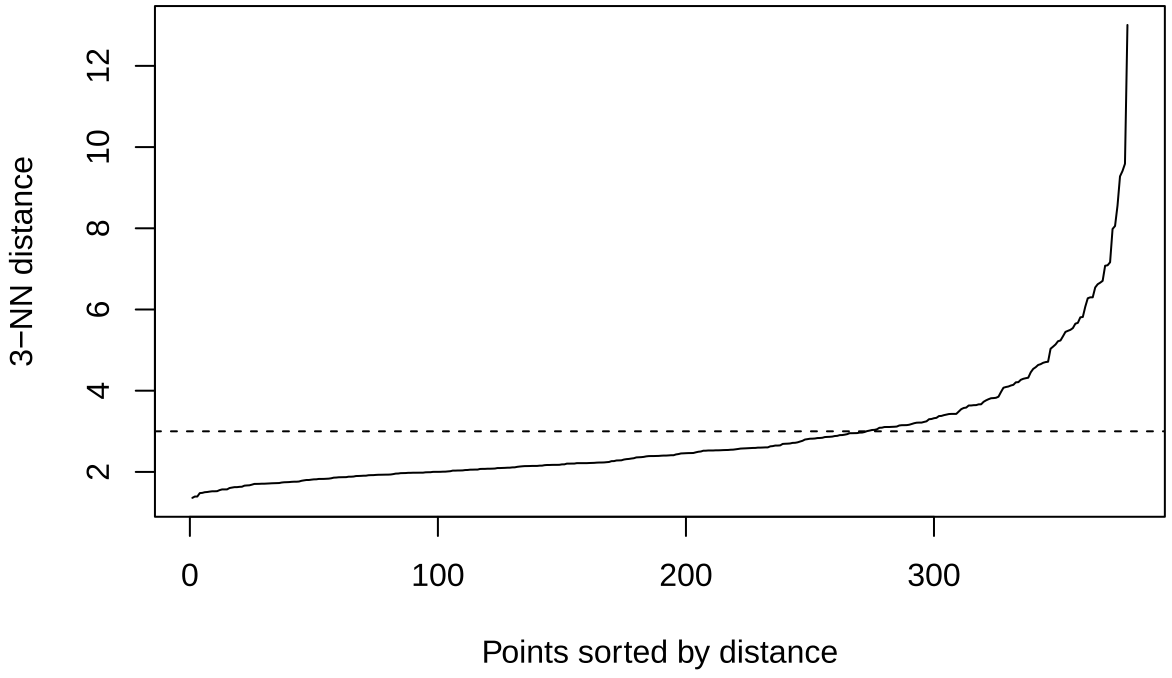
```
## The best minPts value is: 18
```

```r
cat("The best score is:", best_score, "\n")
```

```
## The best score is: 2.747899
```

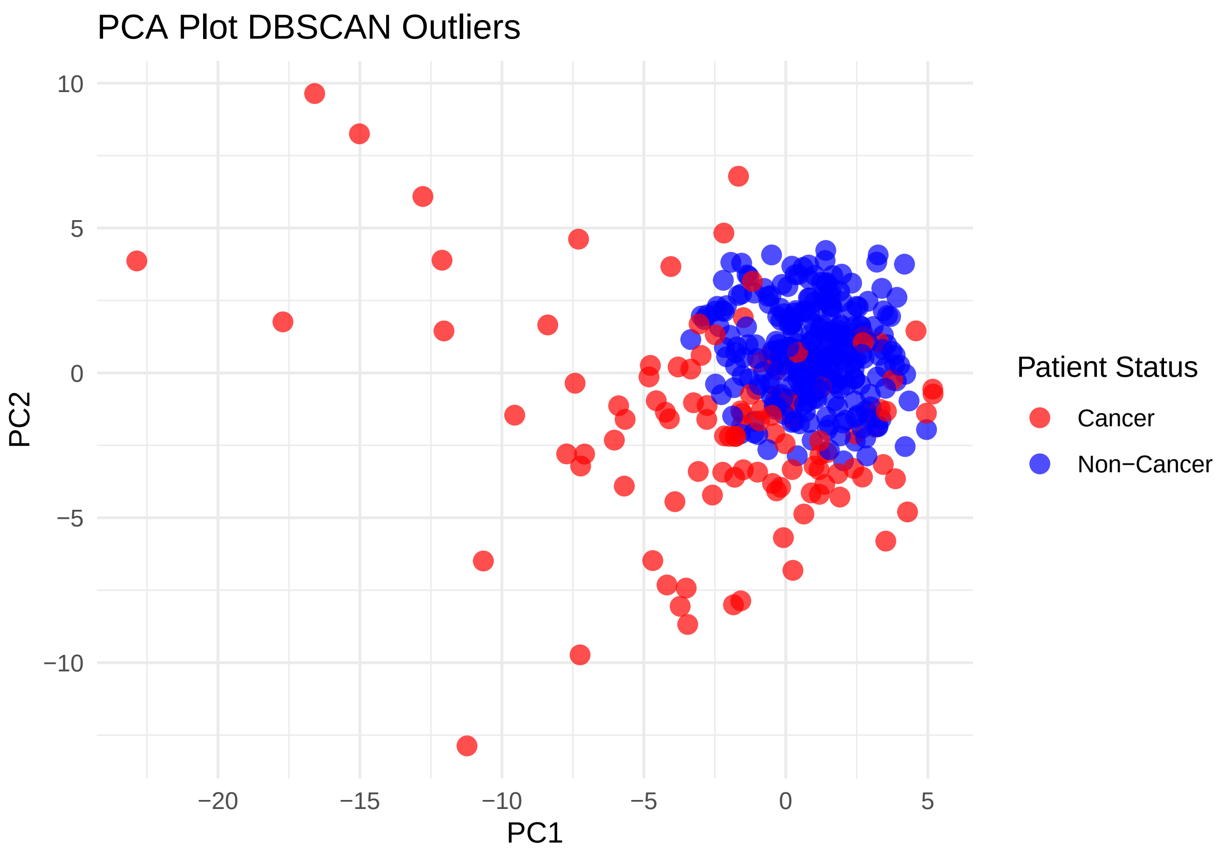
Plot of eps and min points

```r
eps <- best_eps
minpoints <- best_minpts
dbscan::kNNdistplot(features, k = 3); abline(h = eps, lty = 2)
```



Display and Visualize Clusters

```r
dbscan_result <- dbscan(features, eps = eps, minPts = minpoints)

pca_data$CancerStatus <- ifelse(dbscan_result$cluster == 1, "Non-Cancer", "Cancer")

dbscan_plot <- ggplot(pca_data, aes(x = PC1, y = PC2, color = CancerStatus)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_manual(
    values = c("Cancer" = "red", "Non-Cancer" = "blue")
  ) +
  theme_minimal() +
  labs(
    title = "PCA Plot DBSCAN Outliers",
    x = "PC1",
    y = "PC2",
    color = "Patient Status"
  )
dbscan_plot
```



Outliers are those that do not fall into the main cluster and we will assume to have cancer patients.

Confusion Matrix

```
metrics_dbscan <- compute_metrics(
  labels = labels,
  predictions = dbscan_result$cluster,
  method = "outlier",
  positive_class = 0
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 267   0
##          1  90  21
##
##                Accuracy : 0.7619
##                  95% CI : (0.7157, 0.804)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2479
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7479
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.1892
##              Prevalence : 0.9444
##          Detection Rate : 0.7063
##    Detection Prevalence : 0.7063
##       Balanced Accuracy : 0.8739
##
##        'Positive' Class : 0
##
```

```
sensitivity_dbscan <- metrics_dbscan$sensitivity
precision_dbscan <- metrics_dbscan$precision
specificity_dbscan <- metrics_dbscan$specificity

cat("DBSCAN Sensitivity:", sensitivity_dbscan, "\n")
```

```
## DBSCAN Sensitivity: 0.7478992
```

```
cat("DBSCAN Precision:", precision_dbscan, "\n")
```

```
## DBSCAN Precision: 1
```

This model does well to capture all patients that have cancer, since all 21 patients were correctly identified. However, 90 other patients were incorrectly classified as cancer patients. On the bright side, we have no false negatives.

KNN

K-Nearest Neighbors (KNN) is a versatile algorithm often used for outlier detection by assessing the proximity of data points in a multidimensional space. In this approach, a point is considered an outlier if its distance to its nearest neighbors is significantly larger compared to other points in the dataset.

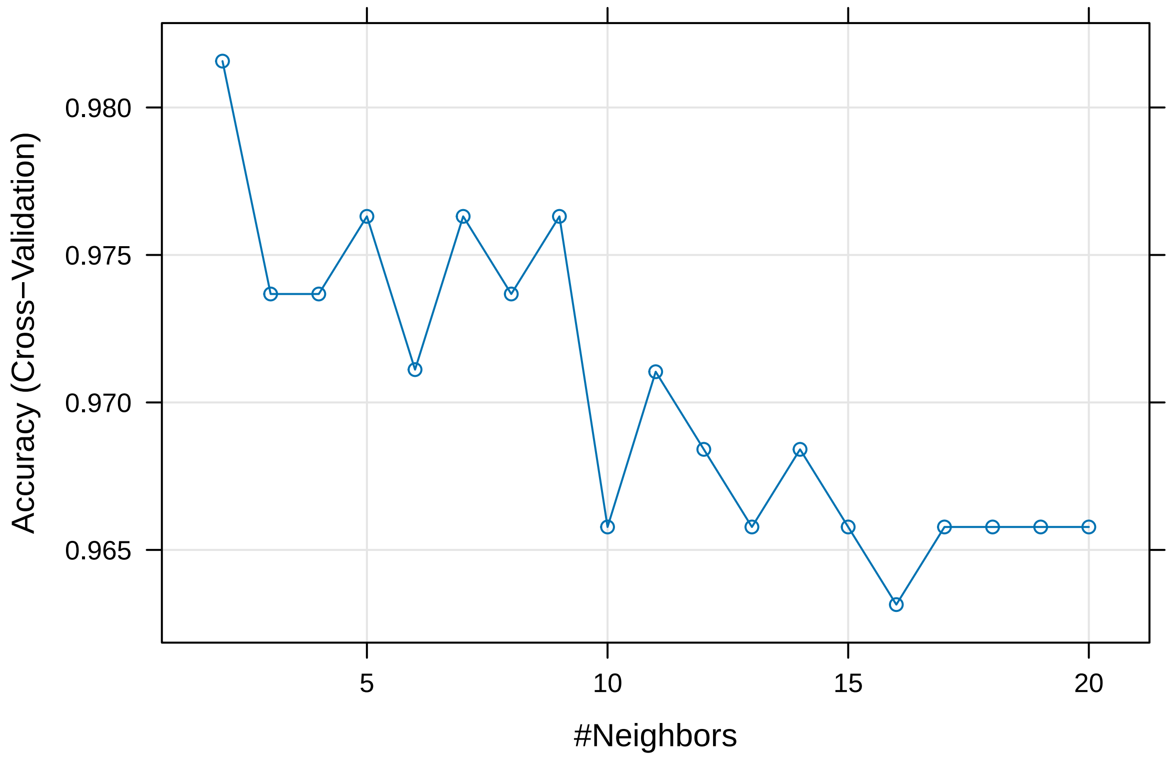
Choose K

The most important paramter in KNN is the K values. For this, we will go through a sequence ot 2 to 20 to find the most optimal values of K.

```
set.seed(12345)
k_values <- seq(2, 20)

train_control <- trainControl(method = "cv", number = 10)  # 10-fold cross-validation

knn_model <- train(
  x = features,               # Features data frame
  y = factor(labels),         # Labels vector converted to factor
  method = "knn",
  tuneGrid = data.frame(k = k_values),
  trControl = train_control
)

best_k <- knn_model$bestTune$k
cat("Optimal k:", best_k, "\n")
```

```
## Optimal k: 2
```

```
plot(knn_model)
```



The plot above shows the effect on accuracy as k increases. To maximize accuracy, we found k of 2 to be optimal. However, this seems quite low and 5 appears to have good accuracy as well.

Run KNN

The other parameter we can tune is the threshold. The threshold is dynamically chosen based on the distribution of KNN scores and the combined performance across key metrics.

```
options(warn = -1)
knn <- get.knn(data = features, k=5)
head(knn$nn.dist)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 2.013204 2.041896 2.055447 2.308181 2.383899
## [2,] 1.463893 1.839022 2.006484 2.091578 2.223112
## [3,] 2.477176 2.633732 2.635624 2.654951 2.735060
## [4,] 2.697160 2.967884 3.143031 3.157542 3.163935
## [5,] 2.244903 2.323041 2.390867 2.625192 2.688568
## [6,] 1.422496 1.714083 1.826968 1.960277 1.961638
```

```
knnscore <- rowMeans(knn$nn.dist)
summary(knnscore)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.365   1.943   2.346   2.782   3.029  12.583
```

```
thresholds <- seq(min(knnscore), max(knnscore), length.out = 100)  # Fine-grained threshold range

results <- sapply(thresholds, function(t) {
  knn_outlier_labels <- ifelse(knnscore > t, 1, 0)  # Assign labels based on the threshold
```

```r
  cfmatrix <- confusionMatrix(factor(knn_outlier_labels), factor(labels))  # Confusion matrix
  cfmatrix$byClass["Sensitivity"] + cfmatrix$byClass["Specificity"] + cfmatrix$byClass["Precision"]  #
})

threshold <- thresholds[which.max(results)]  # Threshold that maximizes the combined metrics
cat("The threshold that maximizes metrics for KNN is:", threshold, "\n")
```

```
## The threshold that maximizes metrics for KNN is: 3.518188
```

```r
knn_outlier_labels <- ifelse(knnscore > threshold, 1, 0)

metrics_knn <- compute_metrics(
  labels = labels,
  predictions = knn_outlier_labels,
  method = "outlier",
  positive_class = 1
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 312   4
##          1  45  17
##
##                Accuracy : 0.8704
##                  95% CI : (0.8323, 0.9025)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3562
##
##  Mcnemar's Test P-Value : 1.102e-08
##
##             Sensitivity : 0.8739
##             Specificity : 0.8095
##          Pos Pred Value : 0.9873
##          Neg Pred Value : 0.2742
##              Prevalence : 0.9444
##          Detection Rate : 0.8254
##    Detection Prevalence : 0.8360
##       Balanced Accuracy : 0.8417
##
##        'Positive' Class : 0
##
```

```r
sensitivity_knn <- metrics_knn$sensitivity
precision_knn <- metrics_knn$precision
specificity_knn <- metrics_knn$specificity

cat("KNN Outlier Sensitivity:", sensitivity_knn, "\n")
```

```
## KNN Outlier Sensitivity: 0.8739496
```

```r
cat("KNN Outlier Precision:", precision_knn, "\n")
```

```
## KNN Outlier Precision: 0.9873418
```

```r
knn_plot <- ggplot(features, aes(x = PC1, y = PC2, color = as.factor(knn_outlier_labels))) +
  geom_point(size = 3, alpha = 0.7) +  # Points for individuals
  scale_color_manual(
    values = c("0" = "blue", "1" = "red"),  # Map 0 to blue (Non-Cancer) and 1 to red (Cancer)
    labels = c("0" = "Non-Cancer", "1" = "Cancer")  # Update legend labels
  ) +
  labs(
    title = "PCA Plot KNN Outlier Detection",
    x = "PC1",
    y = "PC2",
    color = "Patient Status"  # Update legend title
  ) +
  theme_minimal() +
  theme(legend.position = "right")  # Adjust legend position
```

The KNN outlier detection model achieves a high sensitivity of 87.39%, indicating that it effectively identifies most true negatives (non-cancer cases). Its specificity of 80.95% shows a balanced ability to detect true positives (cancer cases) while minimizing false positives. The precision of 98.73% highlights that the majority of predictions for non-cancer cases are accurate, making the model highly reliable in this regard.

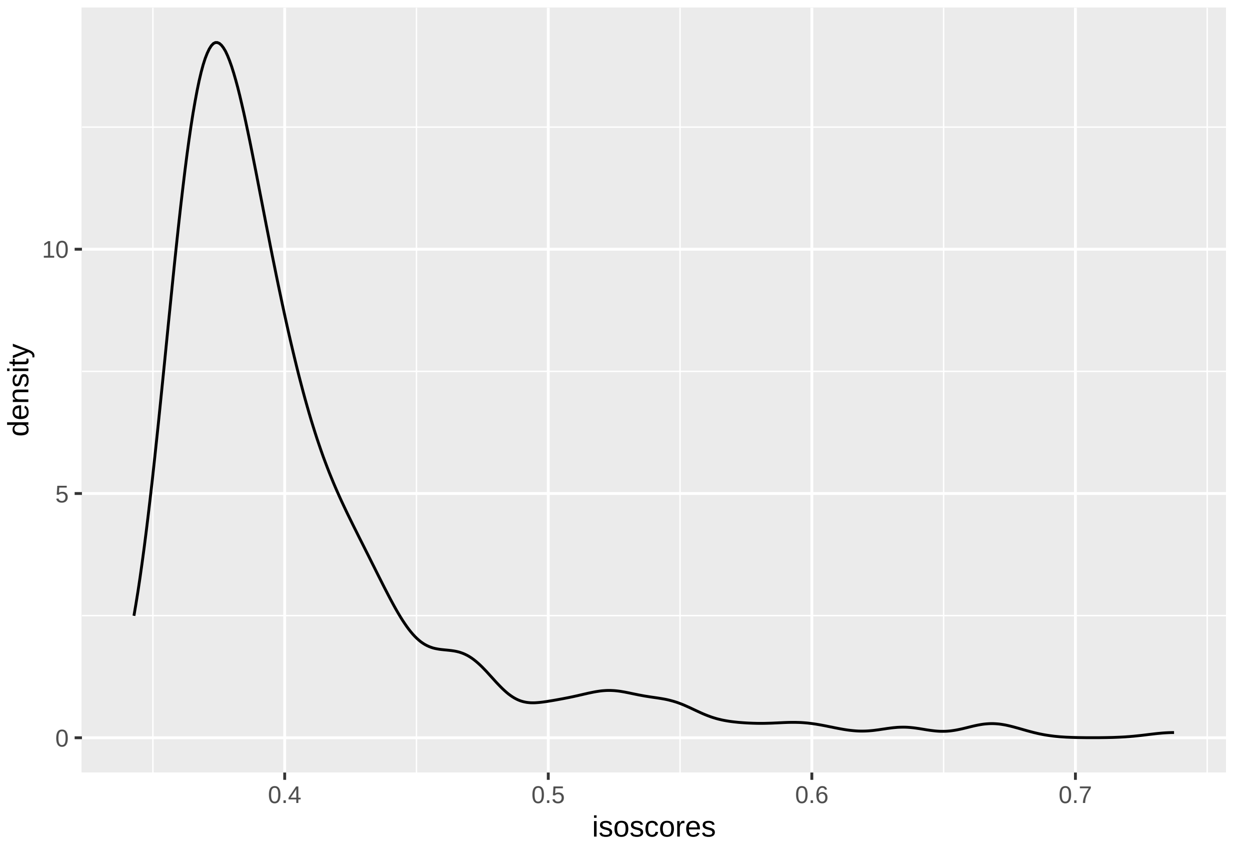
The model's balanced accuracy of 84.17% reflects its relatively strong performance in detecting both classes (cancer and non-cancer). However, the negative predictive value of 27.42% reveals a limitation in identifying all cancer cases, as some true positives may still be missed.

Overall, KNN offers a good balance between sensitivity and specificity, making it suitable for minimizing false negatives, which is critical in cancer detection. However, there is still room for improvement to enhance the detection of cancer cases and further reduce false negatives.

Isolation Forest

Isolation Forest is an unsupervised anomaly detection algorithm designed to identify outliers by isolating data points. It works by constructing random decision trees and measuring the number of splits needed to isolate a point. Outliers, being rare and different, require fewer splits compared to inliers.

```r
iso <- isolation.forest(data = as.matrix(dataraw), sample_size = 378)
dataraw$isoscores <- predict(iso, newdata = as.matrix(dataraw), type = "score")

ggplot(dataraw) + aes(x=isoscores) + geom_density()
```



Confusion Matrix

The threshold is calculated by enumerating through 0.01 to 1 and calcualting a sum of metrics. The best one is chosen.

```
options(warn = -1)
thresholds <- seq(0.01, 1, by = 0.01)
results <- sapply(thresholds, function(t) {
  outlier_labels <- ifelse(dataraw$isoscores > t, 1, 0)
  cfmatrix <- confusionMatrix(factor(outlier_labels), factor(labels))
  cfmatrix$byClass["Sensitivity"] + cfmatrix$byClass["Specificity"] + cfmatrix$byClass["Precision"]
})
threshold <- thresholds[which.max(results)]
cat("The threshold that maximizes metrics is:", threshold, "\n")
```

```
## The threshold that maximizes metrics is: 0.42
```

```
iso_outlier_labels <- ifelse(dataraw$isoscores > threshold, 1, 0)

metrics_iso <- compute_metrics(
  labels = labels,
  predictions = iso_outlier_labels,
  method = "outlier",
  positive_class = 1
)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   0   1
##          0 285   1
##          1  72  20
##
##                Accuracy : 0.8069
##                  95% CI : (0.7634, 0.8455)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2897
##
##  Mcnemar's Test P-Value : 2.55e-16
##
##             Sensitivity : 0.7983
##             Specificity : 0.9524
##          Pos Pred Value : 0.9965
##          Neg Pred Value : 0.2174
##              Prevalence : 0.9444
##          Detection Rate : 0.7540
##    Detection Prevalence : 0.7566
##       Balanced Accuracy : 0.8754
##
##        'Positive' Class : 0
##
```

```r
sensitivity_iso <- metrics_iso$sensitivity
precision_iso <- metrics_iso$precision
specificity_iso <- metrics_iso$specificity

cat("Isolation Forest Sensitivity:", sensitivity_iso, "\n")
```

```
## Isolation Forest Sensitivity: 0.7983193
```

```r
cat("Isolation Forest Precision:", precision_iso, "\n")
```

```
## Isolation Forest Precision: 0.9965035
```

```r
iso_plot <- ggplot(features, aes(x = PC1, y = PC2, color = as.factor(iso_outlier_labels))) +
  geom_point(size = 3, alpha = 0.7) +  # Points for individuals
  scale_color_manual(
    values = c("0" = "blue", "1" = "red"),  # Map 0 to blue (Non-Cancer) and 1 to red (Cancer)
    labels = c("0" = "Non-Cancer", "1" = "Cancer")  # Update legend labels
  ) +
  labs(title = "PCA Plot Isolation Forest Outliers",
       x = "PC1",
       y = "PC2",
       color = "Patient Status") +  # Legend title
  theme_minimal() +
  theme(legend.position = "right")  # Adjust legend position
```

The results for the Isolation Forest model, with a threshold of 0.42, show a moderate sensitivity of 79.83%, meaning the model effectively identifies most true negatives (non-cancer cases). Its specificity of 95.24% indicates strong performance in minimizing false positives, making it reliable at correctly identifying true positives (cancer cases). The model's precision of 99.65% further highlights its high reliability in predicting non-cancer cases.

The balanced accuracy of 87.54% reflects a strong overall performance in detecting both classes (cancer and non-cancer) with reasonable balance. However, the negative predictive value of 21.74% reveals a limitation in its ability to correctly predict cancer cases, with some true positives still being misclassified as non-cancer.

Overall, this Isolation Forest model balances sensitivity and specificity well, making it a suitable candidate for cancer detection. However, its lower sensitivity compared to some other models indicates room for improvement to minimize false negatives, which is critical for ensuring that cancer cases are not missed. Its high specificity and precision make it particularly useful in reducing unnecessary follow-up testing or misclassifications of non-cancer cases.

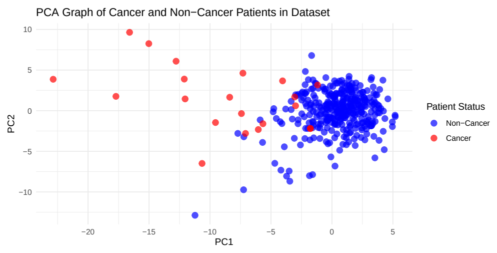
Comparing Models

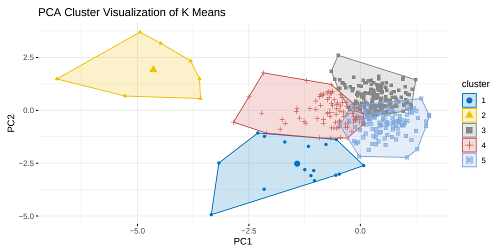
```r
Models <- c("K Means", "Agnes", "Diana", "DBSCAN", "KNN", "Isolation Forest")
Sensitivities <- c(sensitivity_kmeans, sensitivity_agnes, sensitivity_diana, sensitivity_dbscan, sensiti
Specificities <- c(specificity_kmeans, specificity_agnes, specificity_diana, specificity_dbscan, specifi
Precisions <- c(precision_kmeans, precision_agnes, precision_diana, precision_dbscan, precision_knn, pre

Sums <- Sensitivities + Specificities + Precisions

results <- data.frame(
  Model = Models,
  Sensitivity = round(Sensitivities, 2),
  Specificity = round(Specificities, 2),
  Precision = round(Precisions, 2),
  Sum = round(Sums, 2)
)

highlight_max <- function(value, max_value) {
  if (value == max_value) {
    return("#90EE90")
  } else {
    return(NA)
  }
}

reactable(
  results,
  columns = list(
    Sensitivity = colDef(
      style = function(value) {
        list(background = highlight_max(value, max(results$Sensitivity, na.rm = TRUE)))
      }
    ),
    Specificity = colDef(
      style = function(value) {
        list(background = highlight_max(value, max(results$Specificity, na.rm = TRUE)))
      }
    ),
    Precision = colDef(
      style = function(value) {
        list(background = highlight_max(value, max(results$Precision, na.rm = TRUE)))
      }
    ),
    Sum = colDef(
      style = function(value) {
        list(background = highlight_max(value, max(results$Sum, na.rm = TRUE)))
      }
    )
  ),
  defaultColDef = colDef(
    align = "center"
  ),
  bordered = TRUE,
  highlight = TRUE
)
```

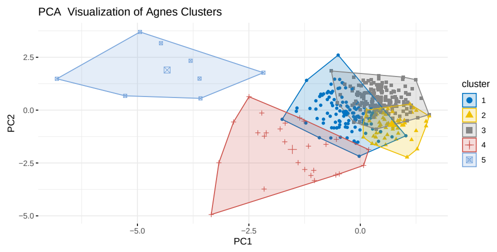
```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

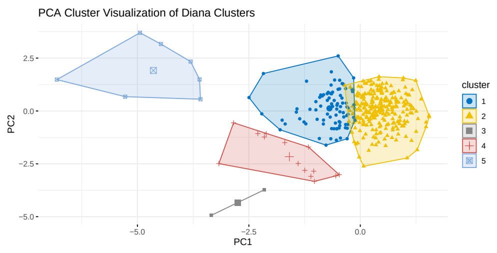
Below is a plot of all clustering and outlier techniques compared to the original:

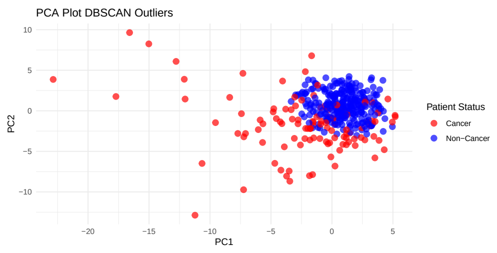
```r
grid.arrange(
  cancer_plot, kmeans_plot, agnes_plot, diana_plot, dbscan_plot, knn_plot, iso_plot,
  ncol = 1,
  nrow = 7,
  heights = c(1, 1, 1, 1, 1, 1, 1) # or another proportional set of values,
)
```

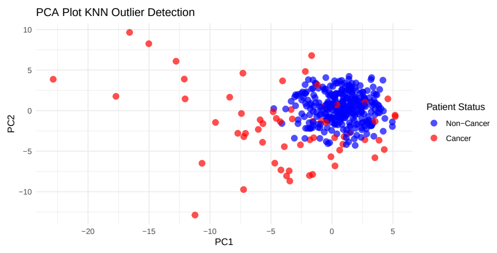

PCA Graph of Cancer and Non−Cancer Patients in Dataset
PC1
PC2
Patient Status
Non−Cancer
Cancer

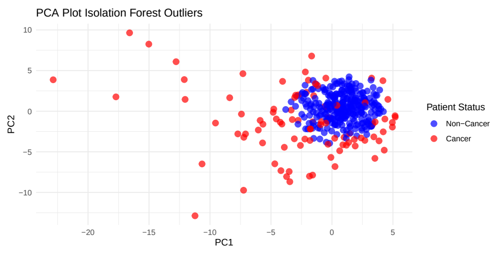

PCA Cluster Visualization of K Means
PC1
PC2
cluster
1
2
3
4
5


PCA Visualization of Agnes Clusters
PC1
PC2
cluster
1
2
3
4
5


PCA Cluster Visualization of Diana Clusters
PC1
PC2
cluster
1
2
3
4
5


PCA Plot DBSCAN Outliers
PC1
PC2
Patient Status
Cancer
Non−Cancer


PCA Plot KNN Outlier Detection
PC1
PC2
Patient Status
Non−Cancer
Cancer


PCA Plot Isolation Forest Outliers
PC1
PC2
Patient Status
Non−Cancer
Cancer

We can see that most of the techniques were able to identify the outliers. Between the clustering methods, there is a noticeable distinction in how the main cluster of non cancer patients was split up. The outlier detection methods are both similar in the points it labeled as outliers with the graphs looking remarkably similar.

Conclusion

The table presents a comparison of multiple clustering and outlier detection models, highlighting their performance across sensitivity, specificity, precision, and a combined metric sum. Among the models, Agnes demonstrates the highest sensitivity at 0.95, making it the most effective at minimizing false negatives—critical for ensuring that cancer cases are not missed. KNN and Isolation Forest also perform well in terms of sensitivity, achieving 0.87 and 0.80 respectively, which makes them strong candidates for cancer detection. On the other hand, DBSCAN achieves perfect specificity (1.0) and precision (1.0), meaning it is highly effective at minimizing false positives and ensuring highly reliable predictions. However, DBSCAN's lower sensitivity of 0.75 means it may miss a significant number of actual cancer cases, which is a limitation in the context of this task.

From a precision standpoint, DBSCAN and Isolation Forest achieve perfect scores (1.0), indicating that when these models predict cancer, the predictions are highly reliable. However, since the primary objective is to minimize false negatives, sensitivity takes precedence over precision in this application. Agnes strikes the best balance between sensitivity and overall performance, as reflected in its strong combined metric sum of 2.64. KNN also performs well overall, with a combined sum of 2.67, reflecting good sensitivity and reasonable specificity.

Overall, this project effectively evaluates multiple approaches to cancer detection, with a clear focus on minimizing false negatives. Agnes stands out as the most suitable model for this task due to its high sensitivity, ensuring that most cancer cases are correctly identified. DBSCAN, with its excellent specificity, could serve as a complementary tool to confirm positive cases and reduce unnecessary testing. The combination of these models provides a robust strategy for detecting cancer while balancing the trade-offs between sensitivity, specificity, and precision.

The patient data shows strong potential for cancer detection when combined with unsupervised learning methodologies such as clustering and outlier detection, although limitations and cautions are still present