Final Project

Chen Bistra 318322492 Eden Ahady 318948106 Lynn Molga Nagar 319090965

Introduction

For this project we chose to work on the NKI Breast Cancer dataset. The dataset has 3 general attributes about the patient, 10 clinical attributes about the tumor and over 1500 gene expression data. One of the attributes in the data is wheter the patient died of the cancer or not, thus we chose to predict whether a patient will die from the tumor, focusing on the clinical data only.

Preprocessing the Data

First we have to import the data

```
NKI_data<-read.csv("NKI_cleaned.csv")
```

This data is already pre-cleaned, but we still wanted to make sure there are no null values in the columns.

```
print(paste("number of NAs in dataset:",sum(is.na(NKI_data))))
```

```
## [1] "number of NAs in dataset: 0"
```

Let's have a first look at the data

```
str(NKI_data)
```

```
## 'data.frame':    272 obs. of  1570 variables:
##  $ Patient       : chr  "s122" "s123" "s124" "s125" ...
##  $ ID            : int  18 19 20 21 22 23 24 25 26 27 ...
##  $ age           : int  43 48 38 50 38 42 50 43 47 39 ...
##  $ eventdeath    : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ survival      : num  14.82 14.26 6.64 7.75 6.44 ...
##  $ timerecurrence: num  14.82 14.26 6.64 7.75 6.32 ...
##  $ chemo         : int  0 0 0 0 0 1 1 1 1 0 ...
##  $ hormonal      : int  0 0 0 1 0 0 1 0 0 0 ...
##  $ amputation    : int  1 0 0 0 1 1 0 0 0 0 ...
##  $ histtype      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ diam          : int  25 20 15 15 15 10 25 15 18 17 ...
##  $ posnodes      : int  0 0 0 1 0 1 1 3 1 0 ...
##  $ grade         : int  2 3 2 2 2 1 1 2 3 3 ...
##  $ angioinv      : int  3 3 1 3 2 1 1 2 1 1 ...
##  $ lymphinfil    : int  1 1 1 1 1 1 1 1 2 1 ...
##  $ barcode       : int  6274 6275 6276 6277 6278 6279 6280 6281 6282 6283 ...
```

```
##  $ esr1          : num  -0.414 0.1953 0.5962 0.5013 -0.0668 ...
##  $ G3PDH_570     : num  -0.9542 0.2446 0.0824 -1.0716 -0.9823 ...
##  $ Contig45645_RC: num  0.051 -0.2 -0.156 -0.206 -0.515 ...
...
```

This dataset contains information on 272 patients, encompassing more than 1500 attributes. One of these attributes indicates whether a patient succumbed to cancer or not, making it a crucial element for our classification task. Our primary objective is to predict whether a patient is likely to face a cancer-related death.

To streamline the dataset for our analysis, we will exclude the "Patient" and "ID" columns, as they do not contribute to the learning process. Furthermore, we have made the decision to eliminate the "survival" column, as its inclusion could potentially influence the outcome of our predictions.

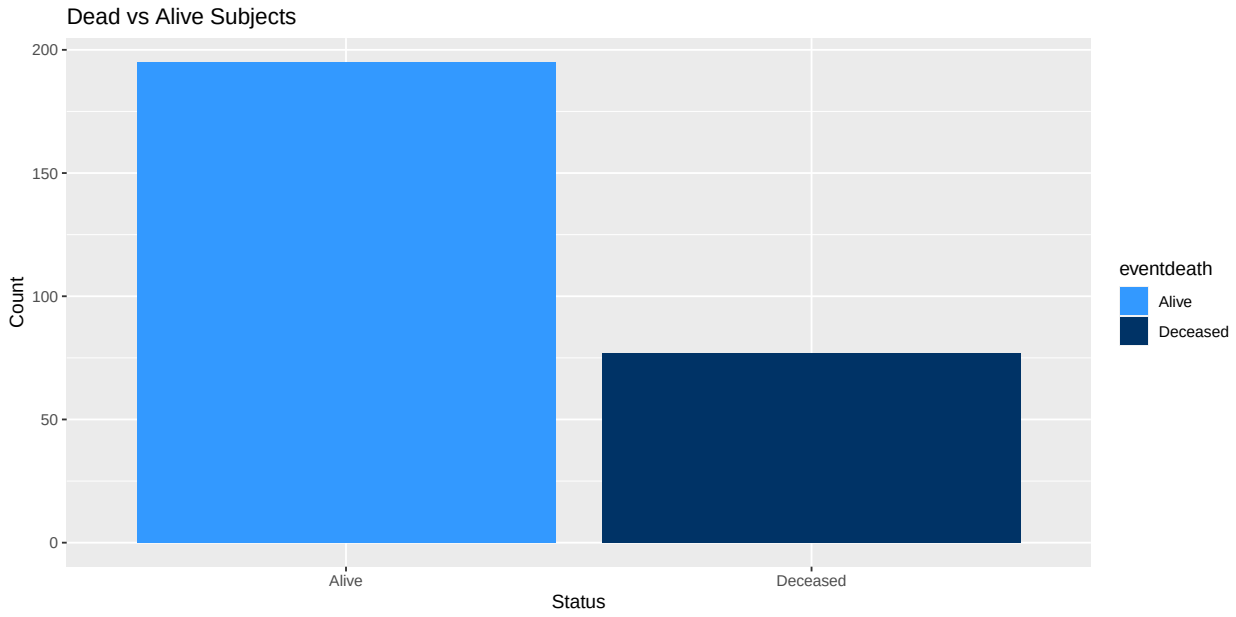
Additionally, we have modified the values in the "eventdeath" column, converting the original binary representation (0 and 1) into more intuitive labels, namely "Alive" and "Deceased." This adjustment will aid in comprehending and interpreting the results more effectively.

```
# create new column based on event death column such than 1 is Deceased and 0 is Alive
eventdeath_labled <- ifelse(NKI_data$eventdeath == 0, "Alive", "Deceased")

# replace current eventdeath column with the labled one
NKI_data$eventdeath<-as.factor(eventdeath_labled)

# removing survival colomn and all gene expression data
cleaned_data<-NKI_data[, c(3:4,6:15)] # with time recurrence

# removing the eventdeath label from the data to get only the features
features<-cleaned_data[-c(2)]
```

EDA

We analyzed the label distribution in the data to select appropriate evaluation metrics for our results.

```
ggplot(cleaned_data, aes(x = eventdeath, fill = eventdeath)) +
  geom_bar() +
  scale_fill_manual(values = c("#3399FF", "#003366")) +
  labs(x = "Status", y = "Count") +
  ggtitle("Dead vs Alive Subjects")
```

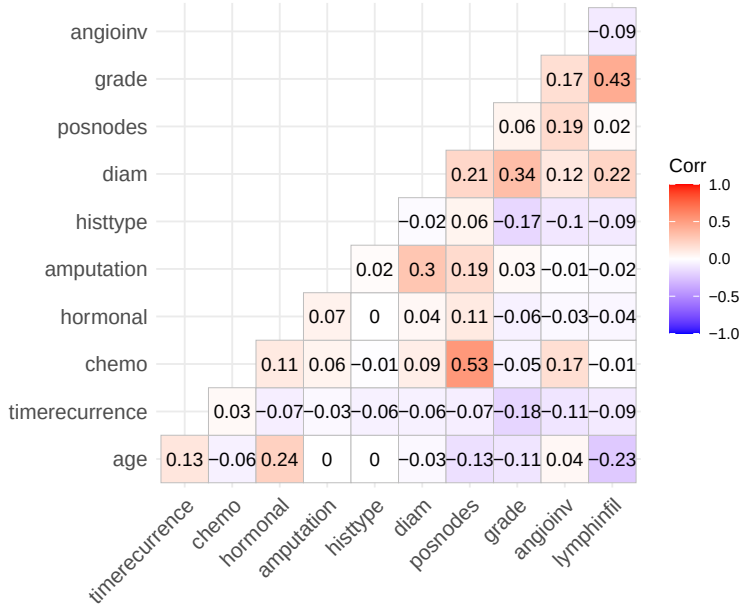


Due to the dataset's imbalanced nature, we will utilize the balanced accuracy metric instead of the regular accuracy metric when evaluating our models.

Dimetionalitiy Reduction

Using Correlation Matrix

We visualized a correlation matrix to understand how the features affect each other, and to remove highly correlated features.



checking for highly correlated features

```
highlyCorrelated <- findCorrelation(cor_matrix, cutoff=0.75)

print(highlyCorrelated, type = "lower", lab = TRUE)
```

```
## integer(0)
```

We can see that there are no features that have a high correlation with other features, so we will try to use rfe which is a dimenionality reduction method that uses random forests to determine the most important features.

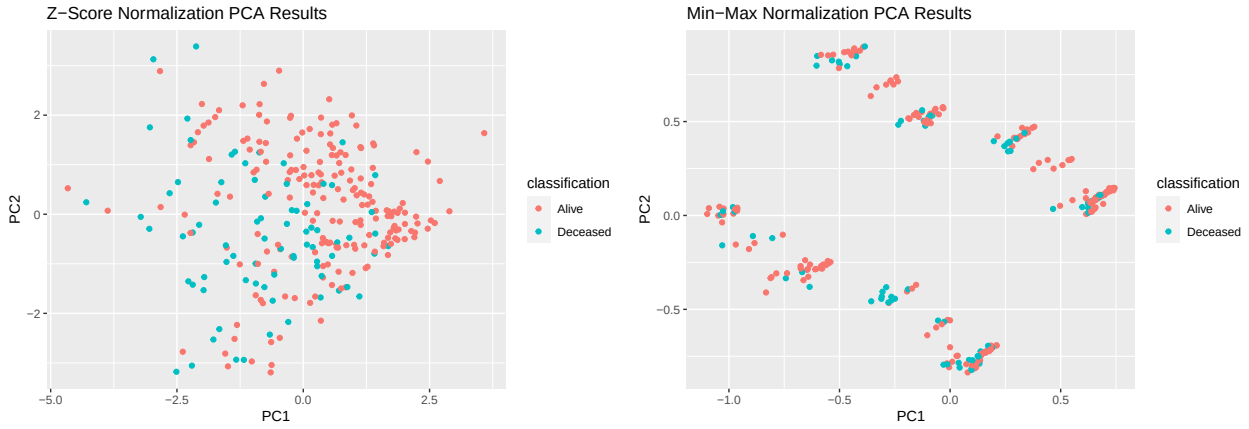
Using rfe

```
# define the control using a random forest selection function
filterCtrl <- rfeControl(functions=rfFuncs, method="cv", number=3)
# run the RFE algorithm
results <- rfe(x= features, y= cleaned_data$eventdeath, sizes = c(1:11), rfeControl=filterCtrl)

chosen_features<-c(results[["optVariables"]])
num_of_features<-length(chosen_features)

if (num_of_features > 5){
  chosen_features<-c(results[["optVariables"]])[1:5]
}

cat("chosen features are:", chosen_features)
```

```
## chosen features are: timerecurrence grade posnodes chemo lymphinfil
```

Normalizing the data

Some of the Machine learning algorithm requires normalized data, so here we try z-score normalization and min-max normalization and see how they separate our data using PCA.



As seen above, both z-score and min-max reduction are not doing well on our data, but we will use z-score because it looks slightly better. In addition, the PCA shows that our data is not linearly separable, so we will have to use non-linear machine learning algorithms.

Machine Learning Algorithms

We will apply the models on the reduced data and on all the data and at the end conclude which gave better results

KNN Algorithm

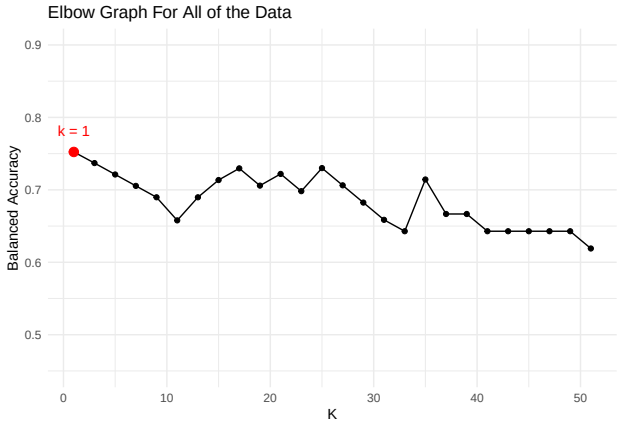
Splitting the Data to Test and Train

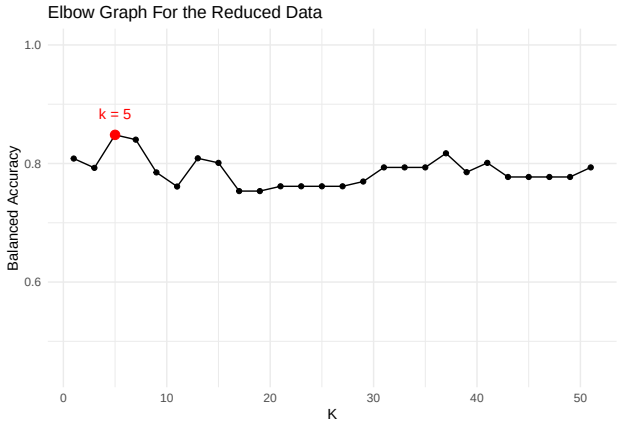
First we have to split our data to train and test. Here we use KNN algorithm which needs normalized data, so we will do the split on the z_score normalized data. We split the data to 190 train samples and 82 test samples (70%-30%). In addition after splitting the data we chose the chosen features from the feature reduction to create the same train and test for the reduced data.

```
# splitting z_score data
train_sample <- sample(272, 189)

zscore_train <- zscore_data[train_sample, ]
zscore_train_labels <- as.factor(cleaned_data$eventdeath[train_sample])

zscore_test <- zscore_data[-train_sample, ]
zscore_test_labels <- cleaned_data$eventdeath[-train_sample]

# applying the reduced data on the split
re_zscore_train <- zscore_train[c(chosen_features)]
re_zscore_train_labels<-zscore_train_labels

re_zscore_test<-zscore_test[c(chosen_features)]
re_zscore_test_labels<-zscore_test_labels
```

Choosing best k

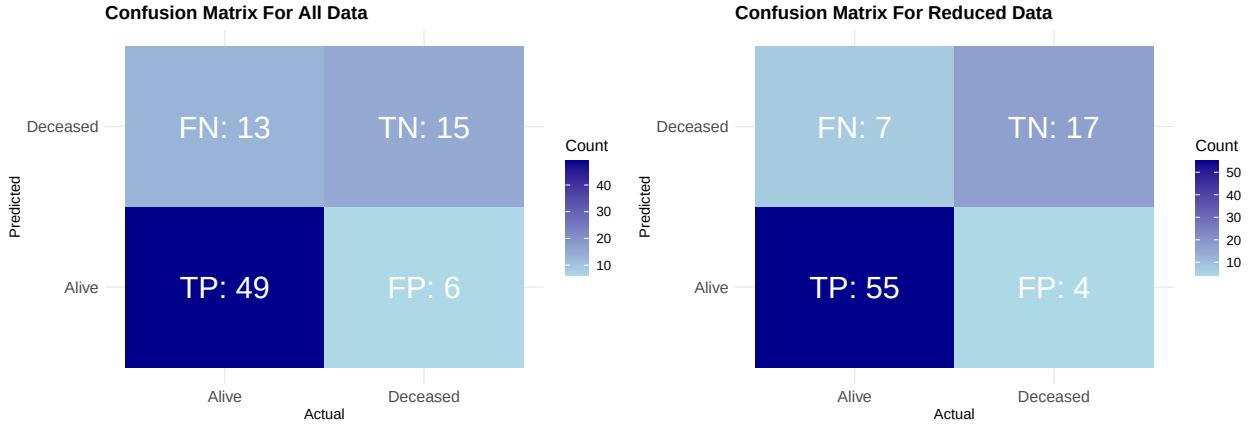
We looped over 25 k's from 1 to 51 (with jupms of 2) and chose the one that had the best balanced accuracy. In the graphs below we can see the results, the k which gave the best balanced accuracy is marked in red. On the right is the results on the reduced data and on the left is the results on the whole data.





KNN Cross Table

after choosing the best k we wanted to visulize the correlation table results and asses which type of data gave us better results. As we can see in the results below, the KNN algorithm preforms better on the reduced data.




```
print(paste("Balanced accuracy of the whole dataset:", best_bal_acc))
```

```
## [1] "Balanced accuracy of the whole dataset: 0.752304147465438"
```

```
print(paste("Balanced accuracy of the reduced dataset:", re_best_bal_acc))
```

```
## [1] "Balanced accuracy of the reduced dataset: 0.848310291858679"
```

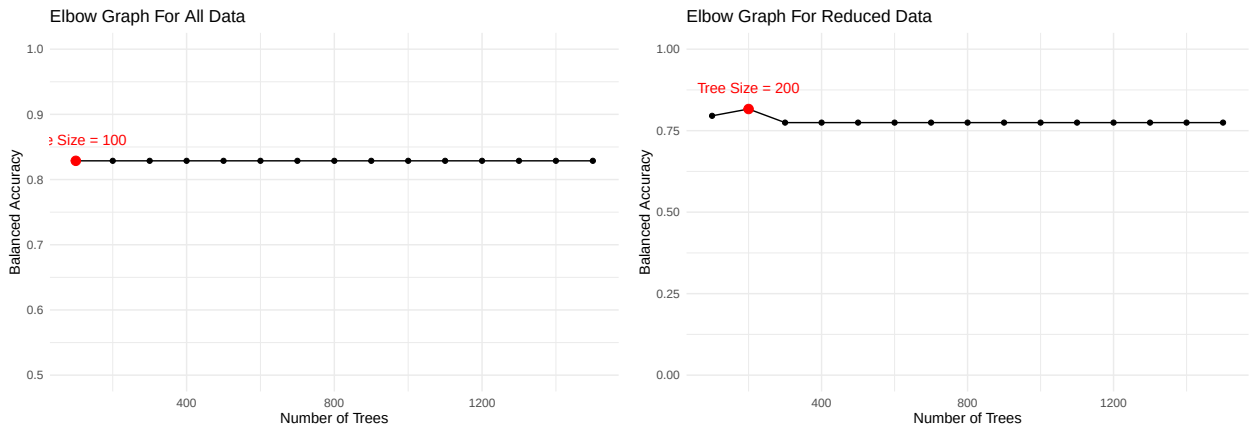
Random Forest Algorithm

Choosing best tree size

We created a for loop which goes through several tree sizes and chooses the one that has the best accuracy Here we also splitted the data to 30% test and 70% train.

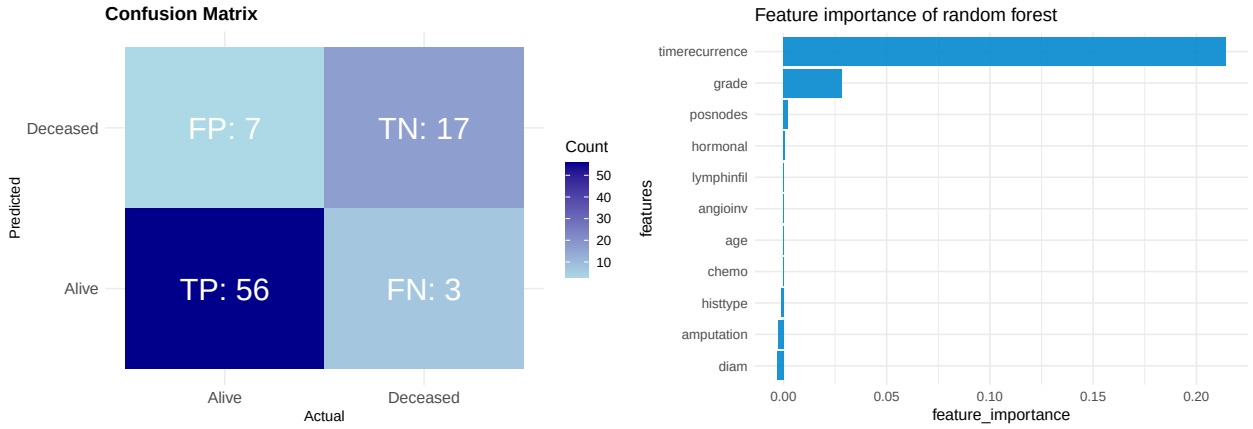
We can see the results below,

elbow graph of regular accuracy



Best Tree Results

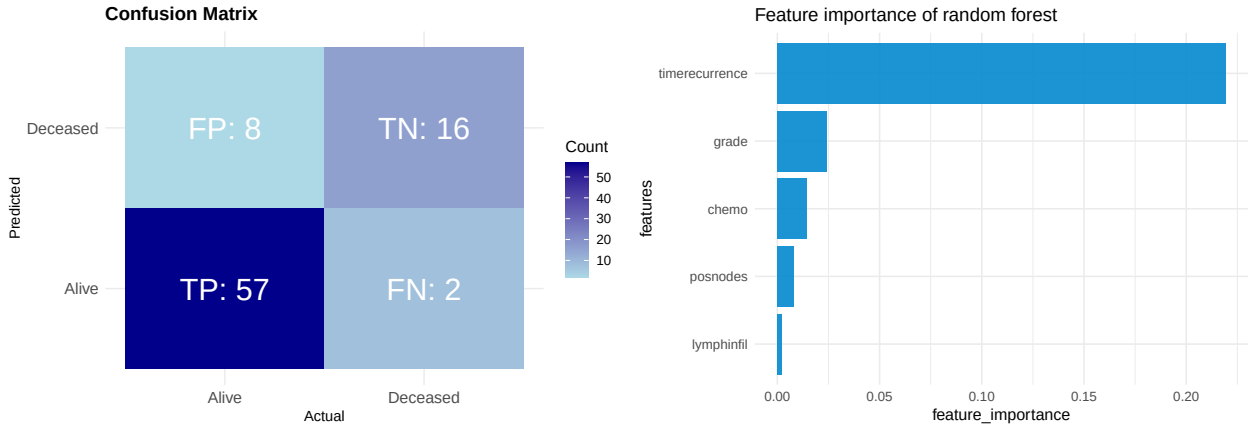
Here we show the conffusion matrix results and feature importace acorrding to all of the features




```
print(paste("Balanced accuracy of the whole dataset:", best_acc))
```

```
## [1] "Balanced accuracy of the whole dataset: 0.828742937853107"
```

Here we show the conffusion matrix results and feature importace acorrding to the reduced data.




```
print(paste("Balanced accuracy of the reduced dataset:", re_best_acc))
```

```
## [1] "Balanced accuracy of the reduced dataset: 0.81638418079096"
```

We can see that on both the reduced data and whole data the RF algorithm is not showing the best result. In addition we can see that both of them voted for "time recurrence" as the most important feature sighnificantly more than the other features.

Decision Trees

We split the data to 70% train and 30% test and ran the desicion tree algorithm using rpart. This time we split the non-normalized data because this type of algorithm doesn't require normalized data.

We used the hyper parameters minsplit=30, maxdepth=3 and cp=0.001

Here we ran the algorithm on the entire data

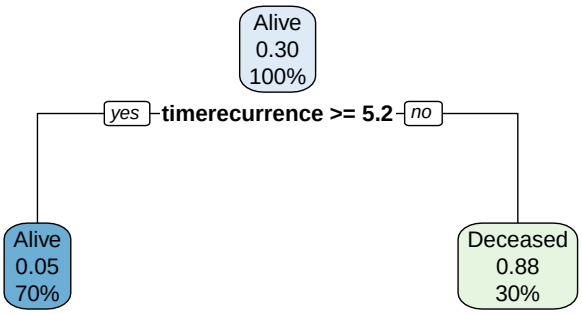
```
set.seed(0)

cart <- rpart(train_labels ~ .,
              data = train, method = "class",
              control=rpart.control(minsplit=30,
                                    maxdepth=3,
                                    cp=0.001))


p <- predict(cart, newdata=test, type="class")

CrossTable(test_labels, p, prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE, dnn = c('actual type', 'predicted type'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  83
##
##
##              | predicted type
##  actual type |     Alive |  Deceased | Row Total |
## -------------|-----------|-----------|-----------|
##        Alive |        56 |         7 |        63 |
##              |     0.675 |     0.084 |           |
## -------------|-----------|-----------|-----------|
##     Deceased |         4 |        16 |        20 |
##              |     0.048 |     0.193 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        60 |        23 |        83 |
## -------------|-----------|-----------|-----------|
##
##
```

```
bal_acc<-balanced_accuracy(p, test_labels)

print((paste('Balanced accuracy:',bal_acc)))
```

```
## [1] "Balanced accuracy: 0.844444444444444"
```

```
rpart.plot(cart)
```



Here we ran the algorithm on the reduced data

```
set.seed(0)

re_cart <- rpart(re_train_labels ~ .,
              data = re_train, method = "class",
              control=rpart.control(minsplit=30,
                                    maxdepth=3,
                                    cp=0.001))


p <- predict(re_cart, newdata=re_test, type="class")

CrossTable(re_test_labels, p, prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE, dnn = c('actual type', 'predicted type'))
```

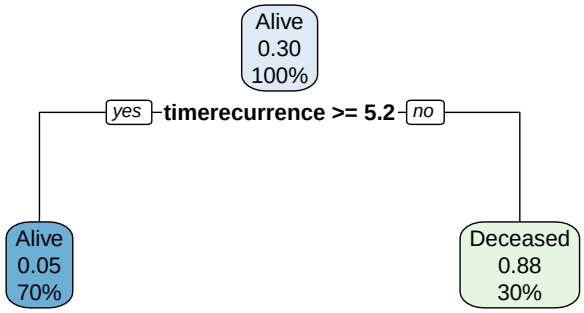
```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  83
##
##
##              | predicted type
##  actual type |     Alive |  Deceased | Row Total |
## -------------|-----------|-----------|-----------|
##        Alive |        56 |         7 |        63 |
##              |     0.675 |     0.084 |           |
```

```
## -------------|-----------|-----------|-----------|
##     Deceased |         4 |        16 |        20 |
##              |     0.048 |     0.193 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        60 |        23 |        83 |
## -------------|-----------|-----------|-----------|
##
##
```

```
bal_acc<-balanced_accuracy(p, re_test_labels)

print((paste('Balanced accuracy:',bal_acc)))
```

```
## [1] "Balanced accuracy: 0.844444444444444"
```

```
rpart.plot(re_cart)
```



Non-Linear SVM

We split the normalized data to 70% train and 30% test and ran non linear SVM using radial kernel. Here we can see the results on the whole data

```
set.seed(1)

regressor_svm <- svm(formula = zscore_train_labels ~ .,
                     data=zscore_train,
                     type = 'C-classification',
                     kernel = 'radial')

y_pred1 = predict(regressor_svm, newdata = zscore_test)
```

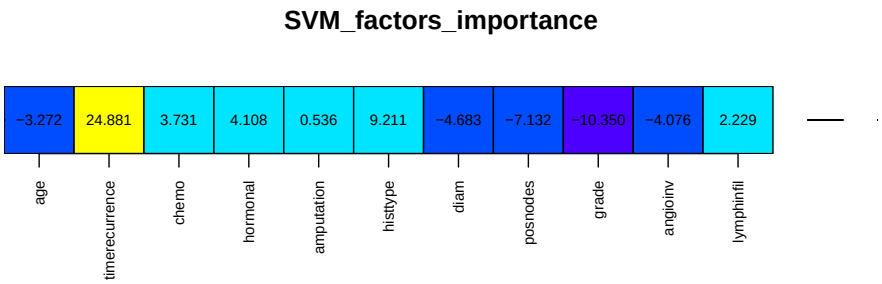
```
CrossTable(zscore_test_labels, y_pred1, prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE, dnn = c('actual type', 'predicted type'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  83
##
##
##              | predicted type
##  actual type |     Alive |  Deceased | Row Total |
## -------------|-----------|-----------|-----------|
##        Alive |        54 |         8 |        62 |
##              |     0.651 |     0.096 |           |
## -------------|-----------|-----------|-----------|
##     Deceased |         5 |        16 |        21 |
##              |     0.060 |     0.193 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        59 |        24 |        83 |
## -------------|-----------|-----------|-----------|
##
##
```

```
bal_acc<-balanced_accuracy(y_pred1, zscore_test_labels)

print(paste('Balanced accuracy:',bal_acc))
```

```
## [1] "Balanced accuracy: 0.816436251920123"
```

```
plot_dataframe <- zscore_train
plot_dataframe$diagnosis <- zscore_train_labels

par(mar=c(15, 0, 5, 0))
SVM_factors_importance = t(regressor_svm$coefs) %*% regressor_svm$SV
plot(SVM_factors_importance, ylab="", yaxt="n",
     xlab="", axis.col=list(side=1, las=2), fmt.cell='%.3f', cex=0.7, cex.axis= 0.7,
     width=400, height=20, main=NULL, col=topo.colors)
```



Here we can see the results on the reduced data

```
set.seed(1)

regressor_svm <- svm(formula = re_zscore_train_labels ~ .,
                     data=re_zscore_train,
                     type = 'C-classification',
                     kernel = 'radial')

y_pred1 = predict(regressor_svm, newdata = re_zscore_test)


# Svm's accuracy



CrossTable(re_zscore_test_labels, y_pred1, prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE, dnn = c('actual type', 'predicted type'))
```

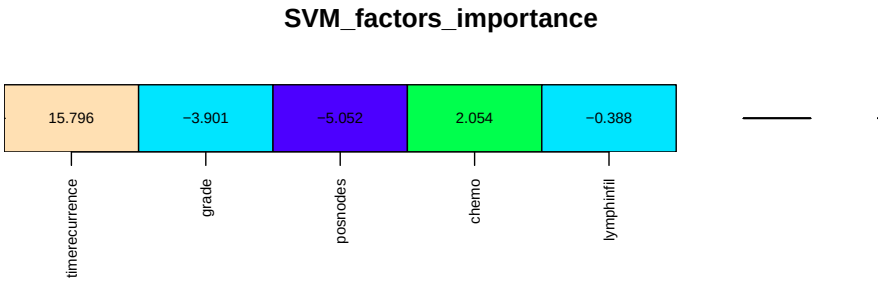
```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  83
##
##
##              | predicted type
##  actual type |     Alive |  Deceased | Row Total |
```

```
## -------------|-----------|-----------|-----------|
##        Alive |        51 |        11 |        62 |
##              |     0.614 |     0.133 |           |
## -------------|-----------|-----------|-----------|
##     Deceased |         4 |        17 |        21 |
##              |     0.048 |     0.205 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        55 |        28 |        83 |
## -------------|-----------|-----------|-----------|
##
##
```

```
bal_acc<-balanced_accuracy(y_pred1, re_zscore_test_labels)

print((paste('Balanced accuracy:',bal_acc)))
```

```
## [1] "Balanced accuracy: 0.81605222734255"
```

```
plot_dataframe <- re_zscore_train
plot_dataframe$diagnosis <- re_zscore_train_labels

par(mar=c(15, 0, 5, 0))
SVM_factors_importance = t(regressor_svm$coefs) %*% regressor_svm$SV
plot(SVM_factors_importance, ylab="", yaxt="n",
     xlab="", axis.col=list(side=1, las=2), fmt.cell='%.3f', cex=0.7, cex.axis= 0.7,
     width=400, height=20, main=NULL, col=topo.colors)
```



Conclusions

In this project, we used four different machine learning algorithms—Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Decision Trees—to predict patient outcomes based on various factors. We evaluated their performance on both the complete dataset and a smaller subset of data.

When applied to the reduced dataset, the K-Nearest Neighbors algorithm performed well, achieving a higher balanced accuracy compared to the complete dataset.

On the other hand, the Random Forest algorithm didn't fare as well on the reduced dataset, showing a lower balanced accuracy than the complete dataset. Interestingly, "timereccurrence" emerged as a consistently significant feature across both datasets for all algorithms.

Surprisingly, the Decision Trees algorithm showed identical balanced accuracy on both the reduced and complete datasets, due to its focus on the "timereccurrence" feature.

The non-linear Support Vector Machine performed better on the reduced dataset, displaying a higher balanced accuracy than the complete dataset. Once again, "timereccurrence" was identified as crucial by this algorithm too.

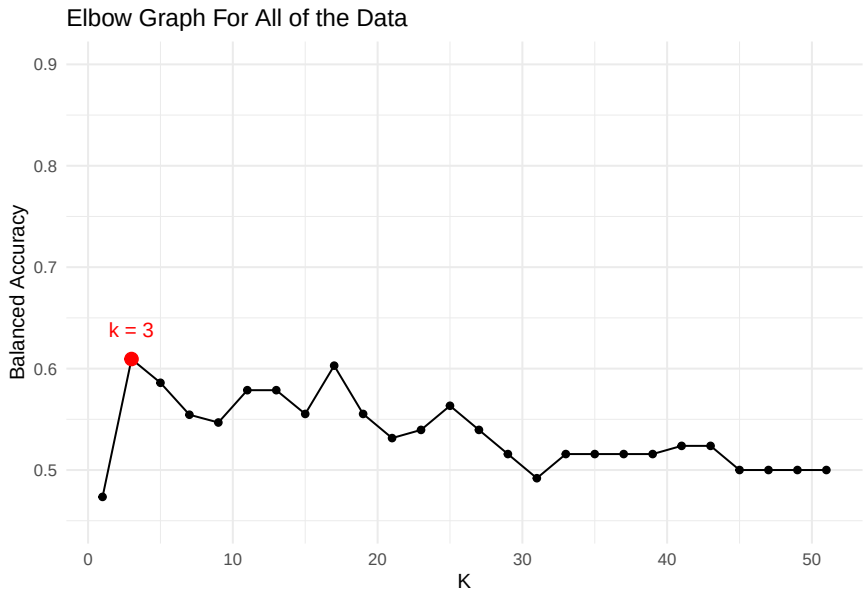
The reduced dataset generally produced better results across most algorithms. Additionally, the unanimous selection of "timereccurrence" as a vital feature by all algorithms raises concerns about potential bias. To address this, we assessed the dataset without this feature, which resulted in less conclusive and weaker outcomes, as detailed in the supplementary section.

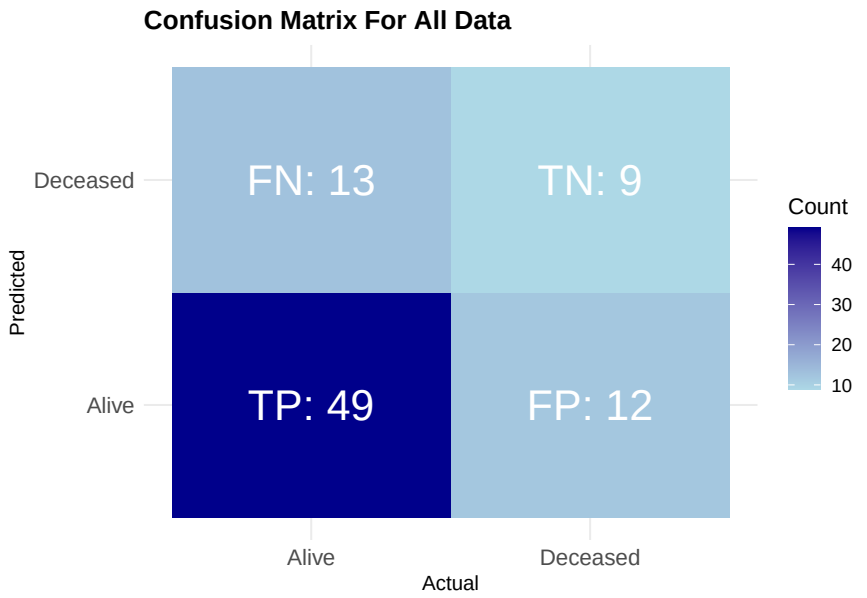
In conclusion, while the "timerecurrence" feature is crucial for accurate predictions in our current study, its strong influence needs careful thought. Our findings also highlight that a larger dataset could bring valuable advantages for future research, possibly revealing new details in predicting patient outcomes.

Supplementry Section

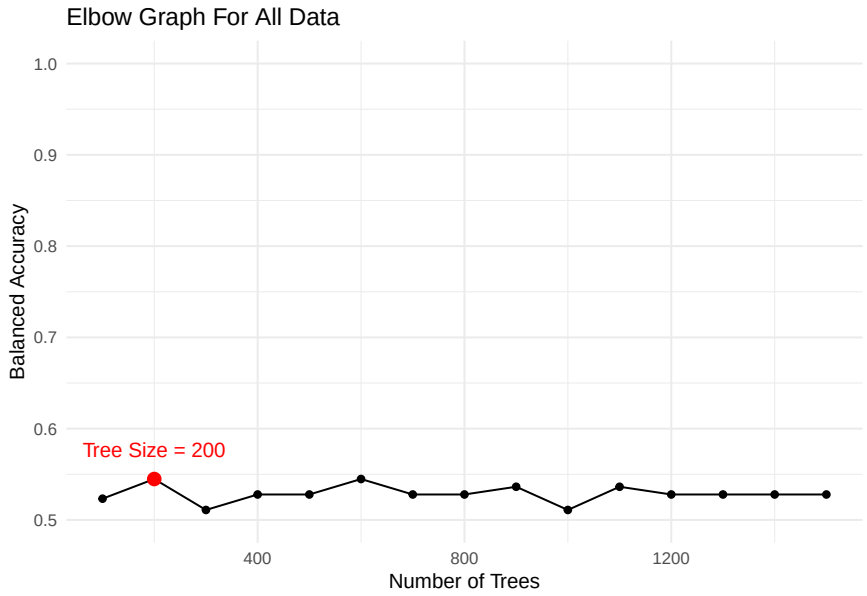
algorithms on the dataset without the timerecurrence feature

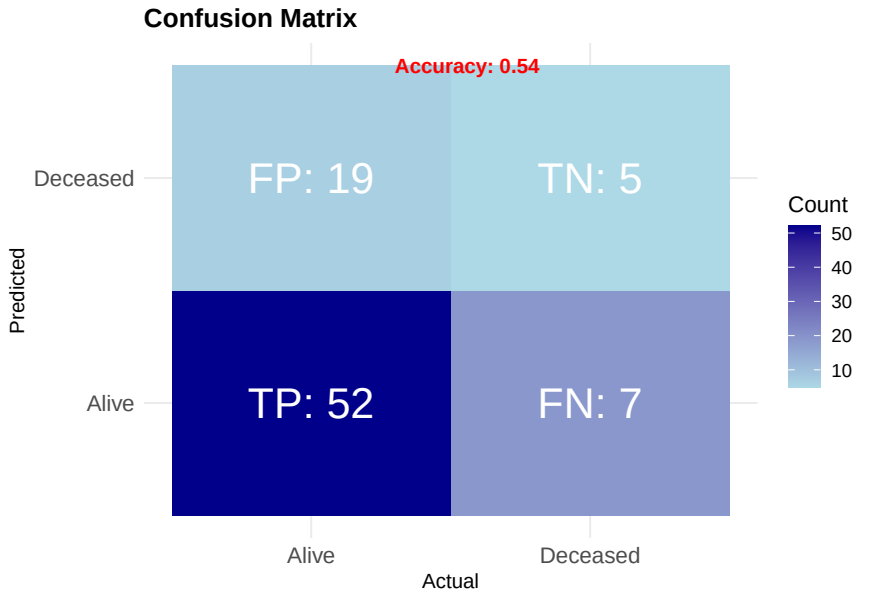
KNN results

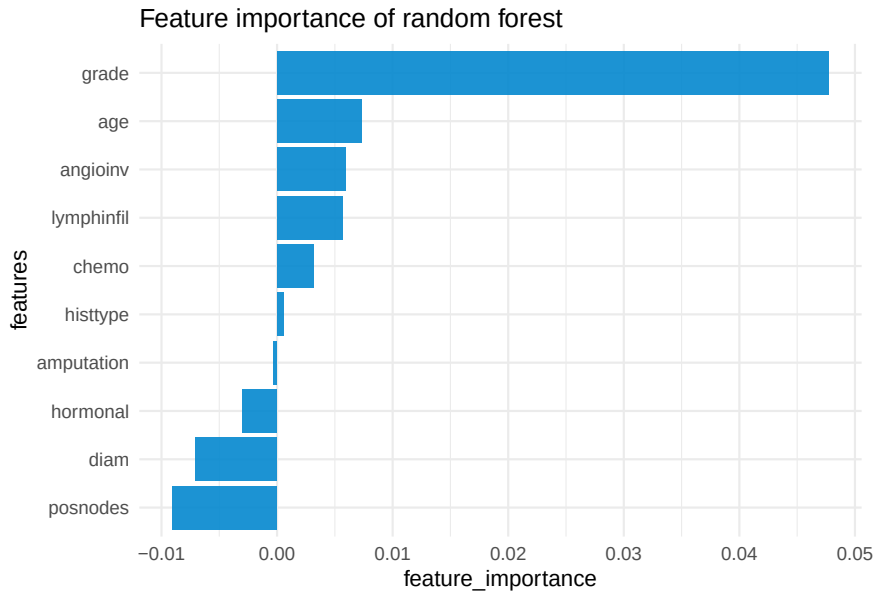



Confusion Matrix For All Data
Deceased
Alive
Predicted
FN: 13
TN: 9
TP: 49
FP: 12
Count
40
30
20
10
Alive
Deceased
Actual

RF Results


Elbow Graph For All Data
1.0
0.9
0.8
0.7
0.6
0.5
Balanced Accuracy
Tree Size = 200
400
800
1200
Number of Trees



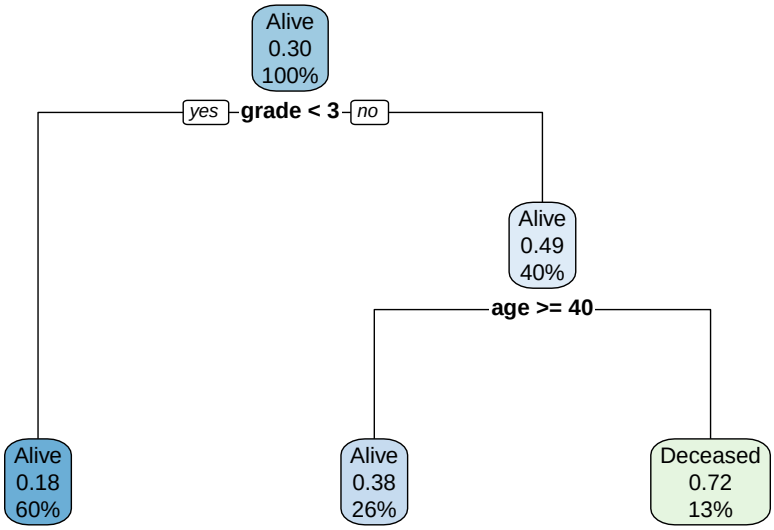


Decision Trees Results

Here we ran the algorithm on the entire data

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
```

```
##
## Total Observations in Table:  83
##
##
##               | predicted type
##  actual type |     Alive |  Deceased | Row Total |
## -------------|-----------|-----------|-----------|
##        Alive |        57 |         6 |        63 |
##              |     0.687 |     0.072 |           |
## -------------|-----------|-----------|-----------|
##     Deceased |        17 |         3 |        20 |
##              |     0.205 |     0.036 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        74 |         9 |        83 |
## -------------|-----------|-----------|-----------|
##
##

## [1] "Balanced accuracy: 0.527380952380952"
```



SVM Results

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  83
```

```
##
##
##              | predicted type
##  actual type |     Alive |  Deceased | Row Total |
## -------------|-----------|-----------|-----------|
##        Alive |        55 |         7 |        62 |
##              |     0.663 |     0.084 |           |
## -------------|-----------|-----------|-----------|
##     Deceased |        14 |         7 |        21 |
##              |     0.169 |     0.084 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        69 |        14 |        83 |
## -------------|-----------|-----------|-----------|
##
##
```

```
## [1] "Balanced accuracy: 0.610215053763441"
```