Various Models to Predict Deaths Caused by Heart Failure

DA5030

Jennifer Nguyen

Spring 2024

Contents

Introduction

My project is focused on machine learning classification methods, specifically through kNN, decision trees, and logistic regression modeling. After constructing each model, I will try to improve each model by cross validation, boosting, tuning, and more. Each method will be trying to predict the class of the target variable. Additionally, ensemble models will also be utilized to make predictions. Specifically, an ensemble model will be constructed as a function that incorporates majority voting of multiple models. A random forest model will also be implemented to showcase homogenous learning. Confusion matrices and performance statistics of each model will be used to compare and contrast for model evaluation.

About the data

The dataset used for my final project contains information about heart failure clinical records from the Faisalabad Institute of Cardiology and the Allied Hospital in Faisalabad in Pakistan (Heart Failure Clinical Records, 2020). Data was originally collected to analyze the effects of serum creatinine and ejection fraction on heart failure patients.

The dataset has 299 observations and 13 variables. There are 6 categorical variables and 7 numerical variables. A short summary of what each variable is provided below.

Integer or Numerical Variables:

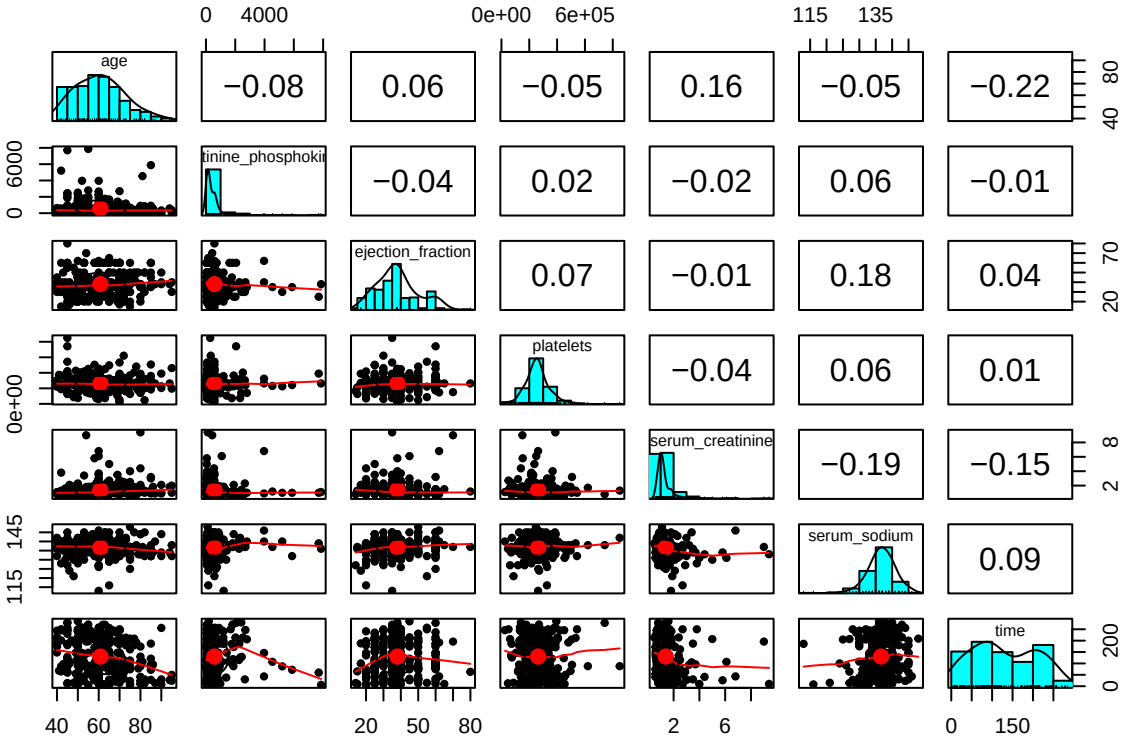
- `age`: age of the patient in years
- `creatinine-phosphokinase`: CPK levels of blood in mcg/L
- `ejection_fraction`: amount of blood leaving heart after each contraction as a percentage
- `platelets`: number of platelets in blood
- `serum_creatinine`: creatinine levels of blood in mg/dL
- `serum_sodium`: sodium levels of blood in mEq/L
- `time`: followup period in days

Categorical Variables:

- `anaemia`: decrease of red blood cells or hemoglobin (1 = anemic, 0 = no anemia)
- `diabetes`: diabetic status of patient (1 = diabetic, 0 = non-diabetic)
- `high blood pressure`: blood pressure status of patient (1 = hypertension, 0 = no hypertension)
- `sex`: sex of patient (1 = woman, 0 = man)
- `smoking`: smoking status of patient (1 = smoker, 0 = non-smoker)
- `death_event`: if patient died during follow up period (1 = death, 0 = survived)

The goal of this project is to explore multiple classification models to predict the target variable `death event`. Specific machine learning algorithms that will be utilized are kNN classification, decision trees, binomial logistic regression and ensemble modeling.

Assessing data quality



To assess data quality, the distributions of the data were viewed. Looking at the histograms of the numeric columns in the dataset, it can be seen that some skewedness is apparent. For example, the `creatinine_phosphokinase` and `serum_creatinine` features are severely right skewed. The `serum_sodium` feature also looks approximately normal but is not centered. Skewedness indicates the possibility of outliers and lack of normality. Likewise, the correlations of the plot above indicate that the numeric features are not highly correlated with one another, indicating that there is no multicollinearity between the numeric features. Highly correlated features would have correlation values of approximate 0.7 or higher. The lack of collinearity is an assumption in multiple machine learning models and will be needed to be assumed in order to build the logistic regression model later on.

Separating into training and test data

For each model, the holdout method is incorporated by partitioning the dataset into training and test data. Before partitioning, the data will be randomized to ensure proper random sampling. After partitioning, the training data will be use to train the model and the test data will be used to validate the model for evaluation.

```
Proportions of class in target variable in training and test data:

test_labels
        0         1
0.6777778 0.3222222

train_labels
        0         1
0.6794258 0.3205742
```

The dataset was partitioned using a 70:30 split. This split proportion was chosen to make sure variance on parameter estimates and performance statistics would be kept low. Furthermore, the split proportion is appropriate as the dataset is small, a 80:20 split would be too unequal. Leaving 30% of the original data as testing data provides ample validation for the trained model. The proportions of the class of the

target variable in the training and test data are shown above to showcase the results of randomization and partitioning.

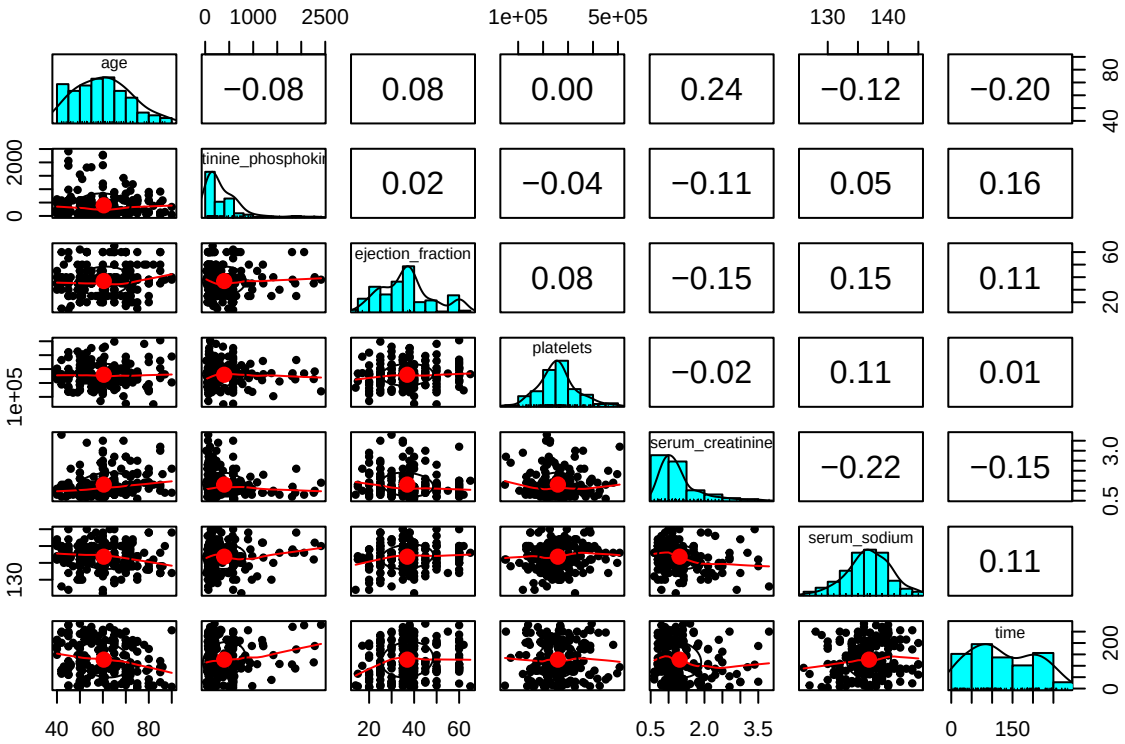
Data pre-processing

Handling missing values

Before any modeling can be done, missing values have to be taken care of. Fortunately, there are no missing values in this dataset. Missing values were randomly added to the dataset to illustrate proper data preparation procedures. This resulted in 16 outliers in the dataset, resulting in 7.66% of the rows of the dataset containing a missing value. Typically, if the dataset has 5% or less in missing values, observations with missing values would be removed. However, since this percentage is greater than 5%, missing values were imputed instead. Missing values were handled using mean imputation for numerical variables and mode imputation for categorical variables.

Outlier detection

The data was checked to see if there were any outliers using the z score approach. Any column that had an observation that had a z-score greater than 2.5 standard deviations was considered an outlier. There were 23 outliers found that constitute approximately 11.0048% of the dataset. Outliers were replaced and imputed using the mean of their respective columns.



The plot above shows the histograms of the numeric features after outliers were imputed. Some features are more centered and look approximately normal such as the `serum_sodium` feature and `platelets` feature. Previously skewed features are still skewed but are less severe. Correlations have slightly increased but are still not high enough to be of concern. Scatterplots have points that are more concentrated together, there are less single data points that are far away from the main bulk of the data distribution.

Encoding categorical features

The categorical variables are already encoded upon downloading the dataset. Since all categorical features only had two levels, they were binary encoded so that they would have a value of 0 or 1. Likewise the categorical features were converted to numeric data type for data modeling.

Feature engineering

Regarding feature engineering, I wanted to see if there were any potential synergistic effects between hypertension and diabetes. To showcase the interaction, a new feature was created by multiplying the binary variables representing each condition. For example, if the patient were to have both a positive hypertension value and positive diabetes value, they would be positive for the interaction variable. But if they were negative for one condition or both conditions, the patient would have a negative interaction value.

`Summary statistics of hbp_diabetes feature:`

```
  0   1
129  80
```

Subsequently, the new interaction variable named `hbp_diabetes` is a binary categorical variable with 0 for no interaction and 1 for interaction. The summary statistics of the new variable for the `train` data are showcased above.

At this point of data pre-processing, training and test datasets for decision tree modeling were saved as `dt_train` and `dt_test`. The pre-processing steps needed for decision tree modeling are missing value and outlier imputation, encoding of categorical variables, and the included feature engineering. Standardization, feature selection, and feature transformation are not needed for decision tree modeling.

Standardization of data

`Summary statistics of training data:`

```
      age            anaemia       creatinine_phosphokinase    diabetes
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00000          Min.   :0.000
 1st Qu.:0.2000   1st Qu.:0.0000   1st Qu.:0.03640          1st Qu.:0.000
 Median :0.4000   Median :0.0000   Median :0.08243          Median :0.000
 Mean   :0.4077   Mean   :0.4163   Mean   :0.15633          Mean   :0.378
 3rd Qu.:0.6000   3rd Qu.:1.0000   3rd Qu.:0.23389          3rd Qu.:1.000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.00000          Max.   :1.000
 ejection_fraction high_blood_pressure   platelets      serum_creatinine
 Min.   :0.0000    Min.   :0.0000      Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.3137    1st Qu.:0.0000      1st Qu.:0.3652   1st Qu.:0.09375
 Median :0.4118    Median :0.0000      Median :0.4674   Median :0.15625
 Mean   :0.4499    Mean   :0.3684      Mean   :0.4613   Mean   :0.22116
 3rd Qu.:0.5098    3rd Qu.:1.0000      3rd Qu.:0.5457   3rd Qu.:0.25644
 Max.   :1.0000    Max.   :1.0000      Max.   :1.0000   Max.   :1.00000
  serum_sodium         sex            smoking            time
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.4737   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.2206
 Median :0.5789   Median :1.0000   Median :0.0000   Median :0.3772
 Mean   :0.5703   Mean   :0.6555   Mean   :0.3349   Mean   :0.4378
 3rd Qu.:0.6842   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.6904
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
  hbp_diabetes
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
```

```
 Mean   :0.3828
 3rd Qu.:1.0000
 Max.   :1.0000
```

Above is the summary statistics of the standardized training dataset. Using min-max normalization, the numeric columns were normalized to ensure that the features are rescaled to a standard range. This also ensures that each feature contributes the same approximate amount to the target variable. If standardization or scaling were not incorporated, each variable would contribute a different amount to the model, resulting in biased results.

However, please note that standardization is only needed for kNN modeling, as decision tree and logistic regression are not affected by standardization of data. Separate training and test datasets will be used for the other models.

Feature selection with the Boruta method

Feature selection allows for the identification of important features to be identified and elimination of the unimportant features. Models that use feature selection reduce their computational times, increase prediction accuracy, and improve model interpretability. For this project, the Boruta method was utilized as the method of feature selection.

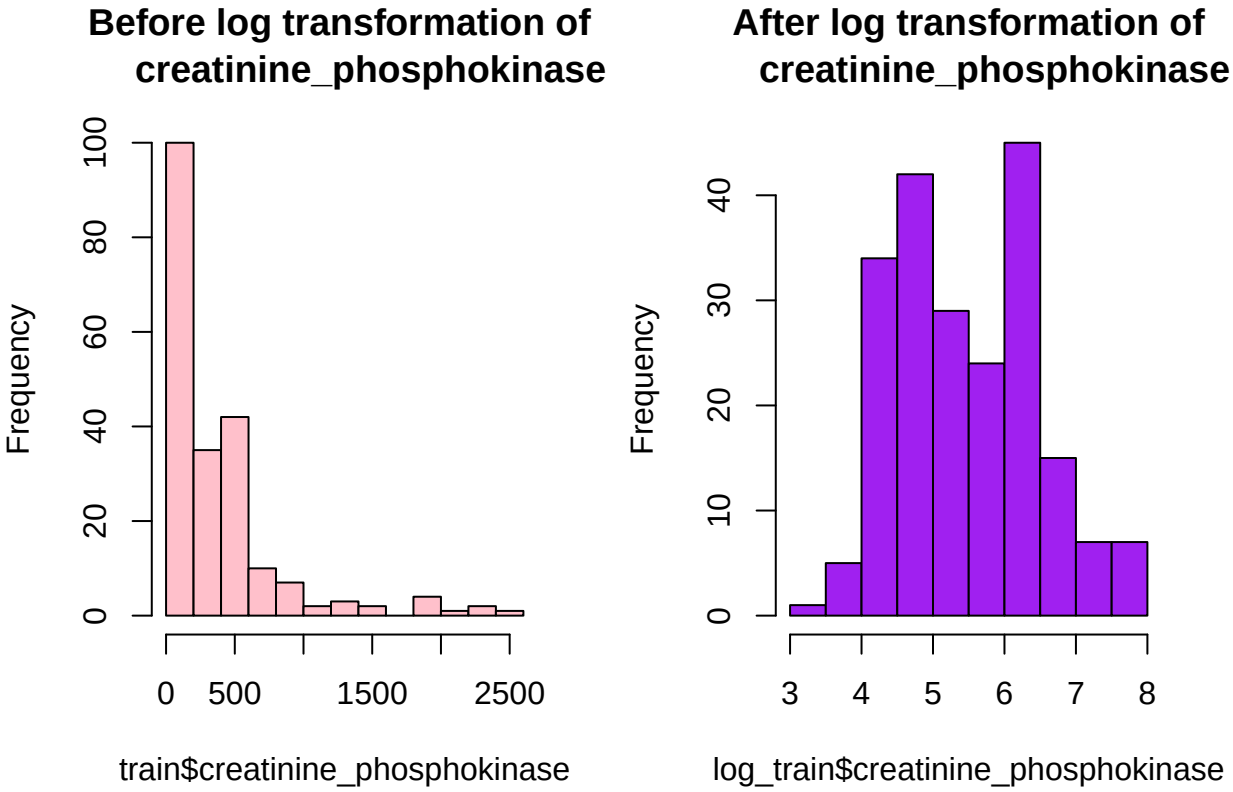
```
Boruta performed 99 iterations in 2.124593 secs.
 4 attributes confirmed important: age, ejection_fraction,
serum_creatinine, time;
 8 attributes confirmed unimportant: anaemia, diabetes, hbp_diabetes,
high_blood_pressure, platelets and 3 more;
 1 tentative attributes left: creatinine_phosphokinase;
```

From the feature selection via Boruta method, the features found to be significant are `age`, `ejection_fraction`, `serum_creatinine`, and `time`. Possibly significant features are `creatinine_phosphokinase` and `serum_sodium`. Unfortunately, it looks like the engineered feature `hbp_diabetes` was found to be non-significant. Since the Boruta method found the individual `high_blood_pressure` and `diabetes` features to be non-significant, it logically makes sense that the interaction of the two would also be non-significant.

For the kNN model, there is no embedded method of feature selection in the model. Therefore, the `knn_train` and `knn_test` has the features found to be significant and will be used to build the kNN model. That is essentially why `knn_train` and `knn_test` datasets were saved at this point of pre-processing. To reiterate, the pre-processing steps done to the `knn_train` and `knn_test` were missing value and outlier imputation, encoding, standardization, feature engineering, and feature selection.

On the other hand, the decision tree and logistic regression model already have embedded methods for feature selection. The decision tree algorithm is greedy, using forward selection to select the best feature subset. The logistic regression model will incorporate step-wise backwards elimination in order to find significant features.

Feature transformation



One requirement of logistic regression is that features have to exhibit reasonably normal distributions. Recall that the distributions of some features are severely right skewed when first looking at the data at a glance. Non-normal distributions were also confirmed by running the Shapiro-Wilk test. To combat skewedness, log transformations on the `creatinine_phosphokinase` and `serum_creatinine` were done. To showcase the effect of log transformation, histograms of the `creatinine_phosphokinase` variable before and after log transformation can be seen above. Notice how after log transformation, the distribution looks approximately normal compared to the severe right skewedness from before.

Since feature transformation is just needed for logistic regression modeling, `log_test` and `log_train` datasets will be saved here. Please note that the `log_test` and `log_train` did not undergo standardization, as standardization is not a necessary pre-processing step for logistic regression modeling. Likewise, feature selection is not needed for logistic regression as the model has an embedded method of feature selection through step-wise elimination.

Results of data pre-processing

Table 1: Pre-processing steps for each model

	kNN	Decision Tree	Logistic Regression
Missing values imputed	X	X	X
Outliers imputed	X	X	X
Encoding categorical features	X	X	X
Feature engineering	X	X	X
Standardization	X		
Feature selection	X		
Feature transformation			X

This concludes the pre-processing stage of the data. To reiterate, training and test data were partitioned after randomizing the original dataset. Then, the training and test data underwent pre-processing. The pre-processing steps for each model are summarized in the table above since each model has different required steps. Each model has their own training and test dataset pair (named after the model, i.e. `knn_train` or `log_test`) but have the same missing values/outliers imputed, the same encoding, and the same feature engineering. The training and test data pairs will be used for their respectively named models.

kNN modeling

The kNN classification algorithm was chosen due to the quick training phase and simpleness of the model. Likewise, there are no assumptions about the distributions in the numeric features. Recall that there was some initial skewedness was observed for some variables but feature transformation was not a requirement for this algorithm. Additionally, the nature of the algorithm as an instance-based learner and non-parametric model was chosen to add some variability to models in predicting the target feature (Lantz, 2023).

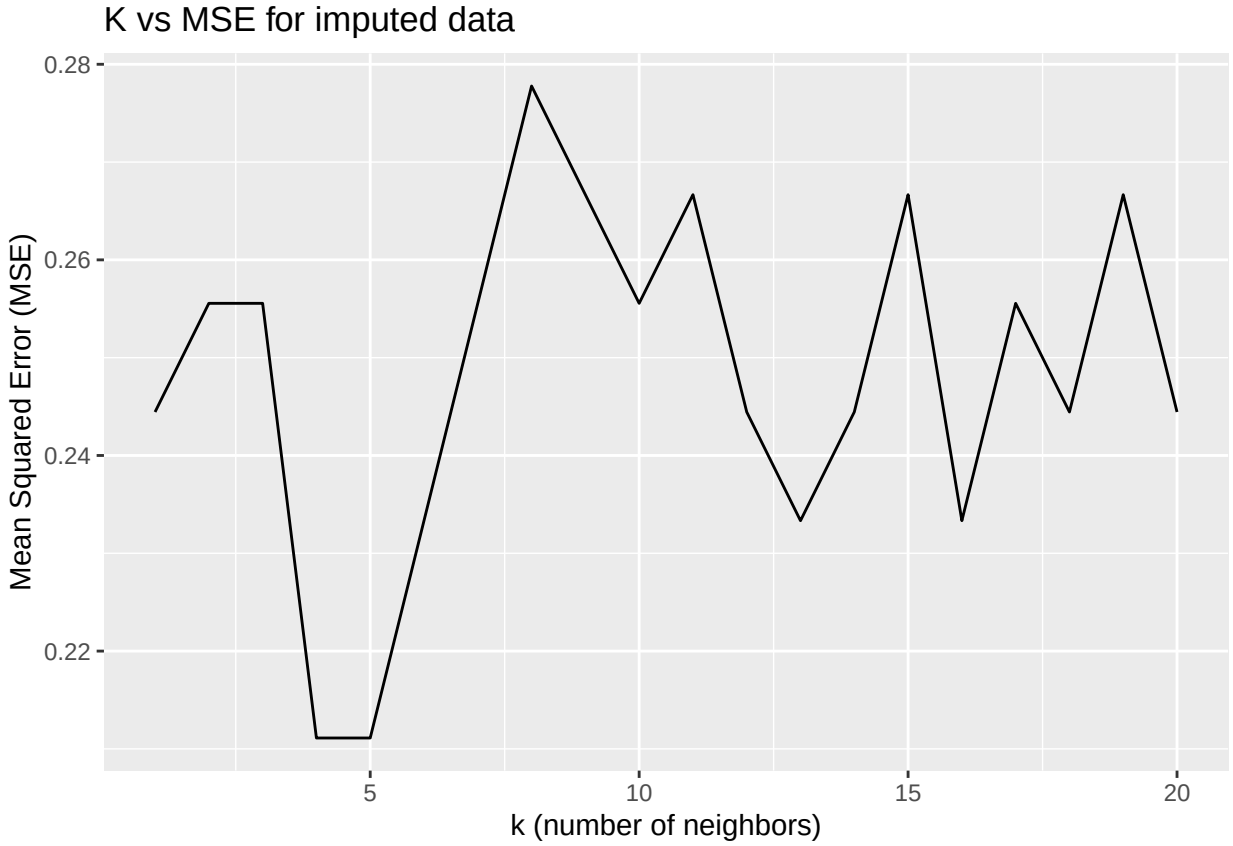
Data preparation for kNN modeling

Prior to building the kNN classification model, outliers and missing values must be imputed, numeric features need to be standardized and categorical features need to be encoded, and feature engineering were done. Likewise, feature selection is another preprocessing step performed as the kNN model does not have any embedded methods in the algorithm. These steps are necessary in order to shape the data and produce an effective model with interpretative results.

Training the kNN model

Table 2: MSE values at each K for imputed data

k	mse
1	0.2444444
2	0.2555556
3	0.2555556
4	0.2111111
5	0.2111111
6	0.2333333
7	0.2555556
8	0.2777778
9	0.2666667
10	0.2555556
11	0.2666667
12	0.2444444
13	0.2333333
14	0.2444444
15	0.2666667
16	0.2333333
17	0.2555556
18	0.2444444
19	0.2666667
20	0.2444444



Using the knn function from the class package, a loop was set up to train the model on the imputed data with various k values from 1 to 20. The mean squared error was calculated from the predictions based off each k-values. From the plot above, it can be seen that the lowest MSE was 0.2111111 with a respective k value of 4 or 5. This is surprising since the typical k-value is usually the square root of the number of observations in the training data (Lantz, 2023). For this situation, the expected k value is approximately 14 since there are 209 observations. However, the k-value with the lowest MSE will be used to construct the kNN model due to the findings show in the line plot.

Predicting with the kNN model

To look at the predictions with the k value of 4, a trained kNN model with the specified k-value of 4 was used to predict the target value on the test data. A confusion matrix was then used to compare the predicted results of the trained kNN model with the actual results.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 55 13
         1  6 16

               Accuracy : 0.7889
                 95% CI : (0.6901, 0.8679)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.01376

                  Kappa : 0.484
```

```
 Mcnemar's Test P-Value : 0.16867

            Sensitivity : 0.5517
            Specificity : 0.9016
         Pos Pred Value : 0.7273
         Neg Pred Value : 0.8088
             Prevalence : 0.3222
         Detection Rate : 0.1778
   Detection Prevalence : 0.2444
      Balanced Accuracy : 0.7267

       'Positive' Class : 1
```

Looking at the results of the confusion matrix, the model results in 55 true negatives, 16 true positives, 6 false positives, and 13 false negatives. The model boasts an accuracy of 0.7888889 accuracy. The sensitivity is 0.5517241 and the specificity is 0.9016393. Lastly, the kappa value was 0.4840072, indicating a fair agreement between the prediction and true values. The sensitivity is rather low and is concerning as it means that there are more false negatives. The presence of false negatives indicates that patients are predicted with survival when exhibiting the characteristics and symptoms that will actually result in death. A high sensitivity is very important in cases where death and survival are predicted in the health industry. Therefore to see if we can improve such statistics, cross validation will be applied in the next section.

Improving the kNN model with cross-validation

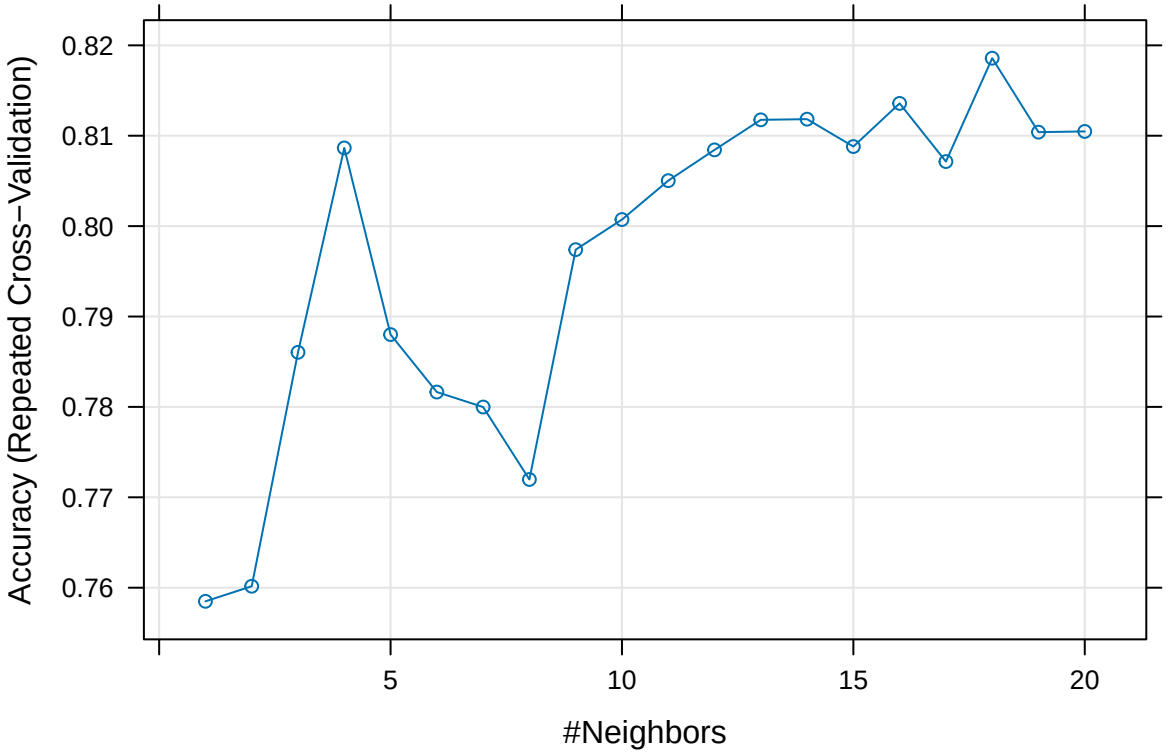
Since the dataset is small in size, k-fold cross validation was used to see if performance can be improved with more reliable results. Cross-validation is a method used to divide the dataset in k folds/subsets. The kNN model will be trained and evaluated k times with those subsets. The resulting performance statistics will be averaged to create generalized model results. Using k-fold cross validation will also prevent overfitting.

The value of k-fold was chosen to be 3 since the original dataset is so small. The training data would be split 3-fold, modeled with each subset, before averaging out the performance statistics.

Training the cross-validation kNN model

```
k-Nearest Neighbors

209 samples
  6 predictor
  2 classes: '0', '1'

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 189, 188, 187, 189, 188, 188, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.7584921  0.4259360
   2  0.7601587  0.4402803
   3  0.7860390  0.5039281
   4  0.8086508  0.5465835
   5  0.7880014  0.4966091
   6  0.7816450  0.4752438
   7  0.7799856  0.4758397
   8  0.7719697  0.4610723
```

```
 9  0.7973954  0.5145964
10  0.8007287  0.5162162
11  0.8050289  0.5247384
12  0.8084343  0.5305608
13  0.8117605  0.5345201
14  0.8118326  0.5315961
15  0.8088023  0.5150972
16  0.8135786  0.5364899
17  0.8071429  0.5189559
18  0.8185786  0.5494607
19  0.8103968  0.5261457
20  0.8104762  0.5242608

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 18.
```



Cross validation results indicated that the most optimal k-value was 18 with an accuracy of 0.8185786 and kappa of 0.5494607. The cross-validation kNN model with a k-value of 18 is then used to make predictions on the test data. Already, the main difference between the regular kNN model and cross-validated kNN model can be seen. The difference in k-values indicate that cross-validation found that the highest accuracy would result from a different k-value than the initial model. It can also be noted that the k-value of 4 also had a smaller peak in accuracy. However, accuracy values can be shown to be relatively high as k equals 13 or more. Using the most optimal k-value of 18, the kNN cross-validationn model was used to predict the target variable in the next section.

Predicting with the cross-validation kNN model

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction  0  1
         0 56 18
         1  5 11

               Accuracy : 0.7444
                 95% CI : (0.6416, 0.8306)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.10582

                  Kappa : 0.337

 Mcnemar's Test P-Value : 0.01234

            Sensitivity : 0.3793
            Specificity : 0.9180
         Pos Pred Value : 0.6875
         Neg Pred Value : 0.7568
             Prevalence : 0.3222
         Detection Rate : 0.1222
   Detection Prevalence : 0.1778
      Balanced Accuracy : 0.6487

       'Positive' Class : 1
```

The resulting confusionMatrix from the cross validation kNN model had 56 true negatives, 11 true positives, 5 false positives, and 18 false negatives. This model had an accuracy of 0.7444444 accuracy. The sensitivity is 0.3793103 and the specificity is 0.9180328. The kappa value is 0.3369635, indicating a fair agreement. The sensitivity is significantly lower than the previous kNN model, allowing us to conclude the kNN cross-validation model has worst performance out of the two models. Likewise, accuracy and kappa values are lower for the cross-validated model further supporting this decision.

kNN model evaluation

With the kNN models, the accuracies of both the regular and boosted models are decently high but not high enough to be implemented in a real-world setting. Specifically, for a situation where the diagnosis and survivability of a patient depends on a model, 70% accuracies are not enough. Likewise, the cross validation kNN model had the lower sensitivity of 0.3793103, already making it a model that should be ruled out. A highly sensitive model is the preferred as a model that can have the least amount of false negatives is essential. Less false negatives ensures that less fewer cases of predicted death are missed for heart failure cases. This is very important for a model in a clinical setting.

Possible reasons why the kNN models did not perform as well as expected may be due to the quick training phase on the already small dataset. The quick training phase does not allow for the algorithm to learn the subtle patterns in the data. Instead, the fast training phase may generalize and cause overfitting. Likewise the difference in k-values between the kNN model and kNN cross-validation model may indicate underlying issues. Choosing a k-value is important as the wrong value can also lead to overfitting and high variance.

Decision tree modeling

A decision tree model was chosen to test the dataset as the flowchart tree structure in an easy to read and understand format. Likewise, decision trees can be used on small datasets and works well as a classifier that can handle multiple types of problems. This algorithm is easy to use in certain situation like credit score modeling or market studies. Similarly, decision trees can be used for diagnosis medical conditions given laboratory test results, patient demographics, and symptoms. Given that the goal of this project is to predict

survival status of heart failure patients, the decision tree algorithm's use in diagnosing medical conditions is fitting.

A decision tree model was constructed using the C50 package. With recursive partitioning, otherwise known as the divide and conquer method, the dataset is continually split into subsets that are then split and so on. The partitioning stops when the decision tree algorithm determines the optimum stopping criteria to predict the target variable (Lantz, 2023). Afterwards a boosted decision tree model will be constructed to see if performance can be improved from the base decision model.

In regards to pruning, post-pruning was chosen over pre-pruning as the chosen method for generalization. Pre-puning was a method that avoids extra work at the cost of missing patterns that may occur if the tree were allowed to grown. However, the dataset is rather small and therefore not computationally extensive. Post-pruning would allow the decision tree model to grow largely, allowing the model to find any possible pattern that may be important given the opportunity. Post-pruning is incorporated in the C5.0 algorithm package.

Data preparation

For decision tree modeling, missing value and outlier handling, binary encoding, and feature engineering were the main pre-processing steps. The decision tree algorithm is based on dividing and partitioning the data in order to make predictions. Therefore, the decision tree model is not sensitive to feature transformation or feature scaling because the partitioning does not change with the transformation monotonically. Therefore, the `dt_train` and `dt_test` dataset will be used to train and test the decision tree model.

Training the C5.0 decision tree model

```
Call:
C5.0.formula(formula = death_event ~ ., data = tree_train)

Classification Tree
Number of samples: 209
Number of predictors: 13

Tree size: 13

Non-standard options: attempt to group attributes


Call:
C5.0.formula(formula = death_event ~ ., data = tree_train)


C5.0 [Release 2.07 GPL Edition]     Mon Apr 15 13:11:07 2024
-------------------------------

Class specified by attribute `outcome'

Read 209 cases (14 attributes) from undefined.data

Decision tree:

time <= 73: 1 (57/10)
time > 73:
:...ejection_fraction <= 25:
    :...diabetes > 0:
```

```
:   :...age <= 53: 0 (4/1)
:   :   age > 53: 1 (7/1)
:   diabetes <= 0:
:   :...sex <= 0: 0 (2)
:       sex > 0:
:       :...platelets <= 198000: 0 (5)
:           platelets > 198000:
:           :...serum_creatinine <= 1.9: 1 (7/2)
:               serum_creatinine > 1.9: 0 (3)
ejection_fraction > 25:
:...age <= 69: 0 (96/1)
    age > 69:
    :...time > 162: 0 (13)
        time <= 162:
        :...serum_creatinine <= 1.18: 0 (5)
            serum_creatinine > 1.18:
            :...anaemia <= 0: 1 (5)
                anaemia > 0:
                :...age <= 75: 1 (2)
                    age > 75: 0 (3)


Evaluation on training data (209 cases):

        Decision Tree
      ----------------
      Size      Errors

        13   15( 7.2%)   <<


       (a)   (b)    <-classified as
      ----  ----
       129    13    (a): class 0
         2    65    (b): class 1


    Attribute usage:

    100.00% time
     72.73% ejection_fraction
     64.59% age
     13.40% diabetes
     11.96% serum_creatinine
      8.13% sex
      7.18% platelets
      4.78% anaemia


Time: 0.0 secs
```

Using the C5.0 package, the decision tree model was trained. The resulting model had a tree size of 13 based off of 13 predictors. Looking at the summary statistics, the attribute usage mainly used three features; `time`, `ejection_fraction` and `age`. Automatically, the C5.0 decision tree model incorporates post-pruning when

building the model.

Predicting with the C5.0 decision tree model

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 53 10
         1  8 19

               Accuracy : 0.8
                 95% CI : (0.7025, 0.8769)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.00716

                  Kappa : 0.5337

 Mcnemar's Test P-Value : 0.81366

            Sensitivity : 0.6552
            Specificity : 0.8689
         Pos Pred Value : 0.7037
         Neg Pred Value : 0.8413
             Prevalence : 0.3222
         Detection Rate : 0.2111
   Detection Prevalence : 0.3000
      Balanced Accuracy : 0.7620

       'Positive' Class : 1
```

From the decision tree model, the predictions resulted in a confusionMatrix with 53 true negatives, 19 true positives, 8 false positives, and 10 false negatives. This model had an accuracy of 0.8. The sensitivity is 0.6551724 and the specificity is 0.8688525. The kappa was 0.5336788, indicating a moderate agreement between predictions and actual values. Comparing to the previous kNN models, the C5.0 decision tree model has higher accuracy and sensitivity values, but a lower specificity. Specifically, the decision tree model showcased the highest sensitivity seen so far, but is still not high enough to be considered for real-life applications. However, it can be considered the best performing model constructed so far. To see if the performance statistics could be further improved, a boosted model with 10 trials was constructed.

Building the boosted decision tree model

There were 10 trials conducted with the boosted C50 model. The features `age`, `creatinine_phosphokinase`, `ejection_fraction`, `serum_creatinine`, and `time` had attribute usage of 90% or more to construct the model. Notably, four out of five of those features were chosen in the Boruta method feature selection, reinforcing that those features are significant predictors for the target variable. The boosted model was then used to predict the target variable with the test data.

Predicting with the boosted decision tree model

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
```

```
         0 57 16
         1  4 13

               Accuracy : 0.7778
                 95% CI : (0.6779, 0.8587)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.02489

                  Kappa : 0.4293

 Mcnemar's Test P-Value : 0.01391

            Sensitivity : 0.4483
            Specificity : 0.9344
         Pos Pred Value : 0.7647
         Neg Pred Value : 0.7808
             Prevalence : 0.3222
         Detection Rate : 0.1444
   Detection Prevalence : 0.1889
      Balanced Accuracy : 0.6914

       'Positive' Class : 1
```

The predictions from the boosted tree model resulted in a confusion matrix with 57 true negatives, 13 true positives, 4 false positives, and 16 false negatives. The boosted tree model had an accuracy of 0.7777778 accuracy. The sensitivity is 0.4482759 and the specificity is 0.9344262. The kappa value is 0.4292961, indicating a moderate agreement. Boosting did not improve the decision tree model, as the boosted model had lower performance statistics compared to the original decision tree model. One possible reason for this is that boosting can cause overfitting, resulting it harder for the model to accurately predict test data.

Decision tree model evaluation

Comparing decision tree models, it can be the seen that the regular C5.0 decision tree model overall has a higher accuracy of 0.8 and higher sensitivity of 0.6551724. However, the boosted decision has a lower specificity of 0.8688525. The sensitivity is important as incorrectly predicting survival as the outcome for heart failure patient when it is actually death should be avoided. Therefore, though the accuracy values are comparable, the better model would be the one with the higher sensitivity. In this case, the regular non-boosted decision tree model is the better model.

Comparing to the kNN models, both the decision tree model had the highest accuracy, sensitivity, kappa values so far out of all the models. This indicates that the decision tree algorithm may be one of the better models to predict heart failure for patients. However, though performance statistics are good in comparison to other models, it does not meet industry standards for real-world application.

Possible reasons for the performance of this model in comparison to previous models is that decision trees are prone to underfitting or overfitting. Likewise, one weakness of the C5.0 algorithm is that it is sensitive to small changes in training data therefore making the decision logic also sensitive. Further improvements can be explored by incorporating the decision tree algorithm in an ensemble model.

Logistic regression modeling

Binomial logistic regression modeling is another method that can be used for classification. It differs from kNN modeling and decision tree modeling as it is a parametric statistical method that fits a regression curve on a categorical predictor variable (Lantz, 2023). Logistic modeling can handle continuous and categorical

independent variables and requires normal distributions that can be achieved by data transformation. The binomial logistic regression model predicts the probability of an observation being in one class of the target variable. This model was chosen due to its ability to provide information regarding effectiveness of predictor variables in terms of size of coefficients and positive or negative associations with the target variable. Likewise the heart failure data set is not highly dimensional meaning logistic regression should be less prone to overfitting.

Data preparation

To construct the logistic regression model for classification, steps must be taken in order to prepare the data for the model. Encoding and handling missing values and outliers are necessary steps prior to model building. Feature engineering was incorporated as well. Logistic regression modeling does need to have reasonably normal distributions and therefore feature transformation needs to be done to offset skewedness in data distributions. Standardization or scaling does not need to be done as the model performance is unaffected by it. Feature selection does also not need to be done as a pre-processing step as the model has an embedded method of stepwise elimination in the model itself.

Training the logistic regression model

```
Start:  AIC=153.16
death_event ~ age + anaemia + creatinine_phosphokinase + diabetes +
    ejection_fraction + high_blood_pressure + platelets + serum_creatinine +
    serum_sodium + sex + smoking + time + hbp_diabetes

                          Df Deviance    AIC
- high_blood_pressure      1   125.17 151.16
- hbp_diabetes             1   125.28 151.28
- sex                      1   125.36 151.36
- platelets                1   125.37 151.37
- diabetes                 1   125.49 151.49
- smoking                  1   125.54 151.54
- anaemia                  1   125.65 151.65
- creatinine_phosphokinase 1   126.09 152.09
- serum_sodium             1   126.85 152.85
<none>                         125.16 153.16
- age                      1   134.35 160.35
- serum_creatinine         1   134.48 160.48
- ejection_fraction        1   151.92 177.92
- time                     1   190.97 216.97

Step:  AIC=151.16
death_event ~ age + anaemia + creatinine_phosphokinase + diabetes +
    ejection_fraction + platelets + serum_creatinine + serum_sodium +
    sex + smoking + time + hbp_diabetes

                          Df Deviance    AIC
- sex                      1   125.36 149.36
- platelets                1   125.37 149.37
- smoking                  1   125.54 149.54
- hbp_diabetes             1   125.56 149.56
- anaemia                  1   125.66 149.66
- diabetes                 1   125.76 149.76
- creatinine_phosphokinase 1   126.11 150.11
- serum_sodium             1   126.89 150.90
```

```
<none>                          125.17 151.16
- age                       1   134.70 158.70
- serum_creatinine          1   135.62 159.62
- ejection_fraction         1   152.00 176.00
- time                      1   193.04 217.04

Step:  AIC=149.36
death_event ~ age + anaemia + creatinine_phosphokinase + diabetes +
    ejection_fraction + platelets + serum_creatinine + serum_sodium +
    smoking + time + hbp_diabetes

                          Df Deviance    AIC
- smoking                  1   125.60 147.60
- platelets                1   125.62 147.62
- hbp_diabetes             1   125.69 147.69
- anaemia                  1   125.88 147.88
- diabetes                 1   125.93 147.93
- creatinine_phosphokinase 1   126.20 148.20
- serum_sodium             1   127.01 149.01
<none>                         125.36 149.36
- age                      1   134.84 156.84
- serum_creatinine         1   135.90 157.90
- ejection_fraction        1   152.00 174.00
- time                     1   193.14 215.14

Step:  AIC=147.6
death_event ~ age + anaemia + creatinine_phosphokinase + diabetes +
    ejection_fraction + platelets + serum_creatinine + serum_sodium +
    time + hbp_diabetes

                          Df Deviance    AIC
- platelets                1   125.84 145.84
- hbp_diabetes             1   125.97 145.97
- diabetes                 1   126.24 146.24
- anaemia                  1   126.24 146.24
- creatinine_phosphokinase 1   126.36 146.36
- serum_sodium             1   127.20 147.20
<none>                         125.60 147.60
- age                      1   135.78 155.78
- serum_creatinine         1   135.93 155.93
- ejection_fraction        1   152.82 172.82
- time                     1   193.65 213.65

Step:  AIC=145.84
death_event ~ age + anaemia + creatinine_phosphokinase + diabetes +
    ejection_fraction + serum_creatinine + serum_sodium + time +
    hbp_diabetes

                          Df Deviance    AIC
- hbp_diabetes             1   126.18 144.18
- anaemia                  1   126.48 144.48
- diabetes                 1   126.53 144.53
- creatinine_phosphokinase 1   126.68 144.68
- serum_sodium             1   127.31 145.31
```

```
<none>                          125.84 145.84
- age                       1   135.99 153.99
- serum_creatinine          1   136.14 154.14
- ejection_fraction         1   152.84 170.84
- time                      1   193.91 211.91

Step:  AIC=144.18
death_event ~ age + anaemia + creatinine_phosphokinase + diabetes +
    ejection_fraction + serum_creatinine + serum_sodium + time

                          Df Deviance    AIC
- diabetes                 1   126.55 142.55
- anaemia                  1   126.86 142.85
- creatinine_phosphokinase 1   127.07 143.07
- serum_sodium             1   127.65 143.65
<none>                         126.18 144.18
- serum_creatinine         1   136.35 152.35
- age                      1   136.64 152.64
- ejection_fraction        1   153.09 169.09
- time                     1   194.33 210.33

Step:  AIC=142.55
death_event ~ age + anaemia + creatinine_phosphokinase + ejection_fraction +
    serum_creatinine + serum_sodium + time

                          Df Deviance    AIC
- anaemia                  1   127.31 141.31
- creatinine_phosphokinase 1   127.61 141.61
- serum_sodium             1   128.09 142.09
<none>                         126.55 142.55
- age                      1   136.64 150.64
- serum_creatinine         1   137.03 151.03
- ejection_fraction        1   153.74 167.74
- time                     1   194.36 208.36

Step:  AIC=141.31
death_event ~ age + creatinine_phosphokinase + ejection_fraction +
    serum_creatinine + serum_sodium + time

                          Df Deviance    AIC
- creatinine_phosphokinase 1   128.75 140.75
- serum_sodium             1   128.82 140.82
<none>                         127.31 141.31
- age                      1   137.05 149.05
- serum_creatinine         1   138.24 150.24
- ejection_fraction        1   154.14 166.14
- time                     1   194.68 206.68

Step:  AIC=140.75
death_event ~ age + ejection_fraction + serum_creatinine + serum_sodium +
    time

                   Df Deviance    AIC
- serum_sodium      1   130.41 140.41
```

```
<none>                    128.75 140.75
- age                 1   138.49 148.49
- serum_creatinine    1   138.85 148.85
- ejection_fraction   1   156.41 166.41
- time                1   194.72 204.72

Step:  AIC=140.41
death_event ~ age + ejection_fraction + serum_creatinine + time

                     Df Deviance    AIC
<none>                    130.41 140.41
- age                 1   141.07 149.07
- serum_creatinine    1   141.47 149.47
- ejection_fraction   1   158.73 166.73
- time                1   195.79 203.79


Call:
glm(formula = death_event ~ age + ejection_fraction + serum_creatinine +
    time, family = binomial, data = log_train)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        1.171329   1.332453   0.879  0.37936
age                0.066837   0.021940   3.046  0.00232 **
ejection_fraction -0.108185   0.023840  -4.538 5.68e-06 ***
serum_creatinine   1.813597   0.568209   3.192  0.00141 **
time              -0.025075   0.004123  -6.082 1.19e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 262.21  on 208  degrees of freedom
Residual deviance: 130.41  on 204  degrees of freedom
AIC: 140.41

Number of Fisher Scoring iterations: 6
```

The procedural stepwise backwards elimination of the binomial logistic regression is shown above. The significant variables were found to be `age`, `ejection_fraction`, `serum_creatinine` and `time`. The final model had the lowest AIC of 140.412826. In comparison, the starting model in the stepwise backwards elimination was 153.16. The model with the lowest AIC value is considered the better fitting model.

The logistic regression model equation is Y = 1.1713293 + 0.0668369(age) + -0.1081846(ejection_fraction) + 1.813597(serum_creatinine) + -0.0250752(time).

One thing I'd like to note was that the stepwise backwards elimination resulted in the finding the same significant features that were found earlier in the final project with the Boruta method. This reinforces the results of feature selection and signifies that the tentative features that may have been significant with Boruta method might not actually be significant.

Predicting with the logistic regression model

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction  0  1
         0 55 17
         1  6 12

               Accuracy : 0.7444
                 95% CI : (0.6416, 0.8306)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.10582

                  Kappa : 0.3503

 Mcnemar's Test P-Value : 0.03706

            Sensitivity : 0.4138
            Specificity : 0.9016
         Pos Pred Value : 0.6667
         Neg Pred Value : 0.7639
             Prevalence : 0.3222
         Detection Rate : 0.1333
   Detection Prevalence : 0.2000
      Balanced Accuracy : 0.6577

       'Positive' Class : 1
```

The predictions from the logistic regression model resulted in a confusionMatrix with 55 true negatives, 12 true positives, 6 false positives, and 17 false negatives. The logistic regression model had an accuracy of 0.7444444 accuracy. The sensitivity is 0.4137931 and the specificity is 0.9016393. The kappa value is 0.3502825, indicating a moderate agreement between predictions and true target values.

Compared to the kNN and decision tree models, the logistic regression model has the comparable accuracy, specificity, and sensitivity values. None of the performance statistics of the logistic regression model are significantly high. Sensitivity is only 0.4137931, indicating that patients are still getting predicted survival when its actually death less than half the time. To see if the model can be further improved I can look towards tuning the logistic regression model in the next section.

Improving the logistic regression model

```
# A tibble: 1 x 3
       penalty mixture .config
         <dbl>   <dbl> <chr>
1 0.0000000001       0 Preprocessor1_Model01
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 56 20
         1  5  9

               Accuracy : 0.7222
                 95% CI : (0.6178, 0.8115)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.21648
```

```
                 Kappa : 0.2642

 Mcnemar's Test P-Value : 0.00511

            Sensitivity : 0.3103
            Specificity : 0.9180
         Pos Pred Value : 0.6429
         Neg Pred Value : 0.7368
             Prevalence : 0.3222
         Detection Rate : 0.1000
   Detection Prevalence : 0.1556
      Balanced Accuracy : 0.6142

       'Positive' Class : 1
```

An attempt to improve the logistic regression model was done by hyperparameter tuning. Specifically, the mixing and penalty arguments were set to be tuned with a grid to find the most optimal values (Chugh, 2023). The predictions from the tuned logistic regression model resulted in a confusionMatrix with 56 true negatives, 9 true positives, 5 false positives, and 20 false negatives. The tuned model had an accuracy of 0.7222222 accuracy. The sensitivity is 0.3103448 and the specificity is 0.9180328. The kappa value is 0.264225 considered a fair agreement between predictions and actual values.

Logistic regression model evaluation

Comparing results from the confusion matrices of the untuned and tuned logistic regression models, the former model performed slightly better. Accuracy values differed by a small margin, but the untuned model had a higher sensitivity of 0.4137931. This sensitivity is still rather dismal overall, especially since previous models have had higher sensitivity values. Specifically, the tuned model had the lowest sensitivity model out of all models so far, making it eliminated from being the best performing model. In the next section, ensemble modeling will be used to see if a combination of multiple models can prove to be the ideal model.

Possible reasons for the performance may be that the logistic regression is that there are numerous assumptions that need to be satisfied. However, real-world data may not be able to satisfy such conditions. Likewise, logistic regression works well with larger datasets. Performance statistics could be improved if more observations were obtained.

Ensemble model

Ensemble models were incorporated to showcase a method of meta-learning (Lantz, 2023). Ideally, the combination of multiple, diverse models in an ensemble method would result in improved performance statistics.

Simple ensemble model with majority voting

The ensemble model was built using a combination of three different machine learning classification models: kNN model, C50 boosted decision tree model, and stepwise backwards elimination logistic regression model. These models were chosen to add tuning to the simple ensemble model. The function was constructed so that the a training data, test data, training labels, and test labels could be inputted. The models are then individually trained and will then make predictions for the test data. The predictions are stored in a data frame. At the end of the ensemble function, the code is written to look at the 3 different predictions from each model for each observation, decide through majority vote for the final predictions, and then output a final prediction that goes into a confusion matrix.

Predicting with the simple ensemble model

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 57  4
         1 20  9

               Accuracy : 0.7333
                 95% CI : (0.6297, 0.8211)
    No Information Rate : 0.8556
    P-Value [Acc > NIR] : 0.9992

                  Kappa : 0.2862

 Mcnemar's Test P-Value : 0.0022

            Sensitivity : 0.6923
            Specificity : 0.7403
         Pos Pred Value : 0.3103
         Neg Pred Value : 0.9344
             Prevalence : 0.1444
         Detection Rate : 0.1000
   Detection Prevalence : 0.3222
      Balanced Accuracy : 0.7163

       'Positive' Class : 1
```

The predictions from the ensemble model resulted in a confusionMatrix with 57 true negatives, 9 true positives, 20 false positives, and 4 false negatives. The simple ensemble model had an accuracy of 0.7333333 accuracy. The sensitivity is 0.6923077 and the specificity is 0.7402597. The kappa value is 0.2861864, indicating a moderate agreement.

Overall, the simple ensemble model performs rather averagely. None of the performance statistics were the highest or lowest but were rather in the middle compared to the other models. This is rather logical since the simple ensemble is basically averaging the results of previously constructed models. To compare to another ensemble method, a random forest model will be constructed in the next section.

RandomForest model

A random forest model was also chosen to showcase a homogenous ensemble model that incorporated bagging. A random forest algorithm can handle both categorical and continous features and has incorporated feature selection (Lantz, 2023). This was chosen to compare and contrast against the simple ensemble method with majority voting model constructed previously. In this model, trees are generated based on random subsets of features. This is one advantage of the random forest model as it prevents the greedy behavior of most decision tree algorithms (Lantz, 2023).

```
# train rf model
rf_model <- randomForest(death_event ~ ., data = tree_train, mtry = sqrt(14))
rf_model
```

```
Call:
 randomForest(formula = death_event ~ ., data = tree_train, mtry = sqrt(14))
```

```
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 15.79%
Confusion matrix:
    0  1 class.error
0 128 14  0.09859155
1  19 48  0.28358209
```

Predicting with the random forest model

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 58 14
         1  3 15

               Accuracy : 0.8111
                 95% CI : (0.7149, 0.8859)
    No Information Rate : 0.6778
    P-Value [Acc > NIR] : 0.003498

                  Kappa : 0.5198

 Mcnemar's Test P-Value : 0.015293

            Sensitivity : 0.5172
            Specificity : 0.9508
         Pos Pred Value : 0.8333
         Neg Pred Value : 0.8056
             Prevalence : 0.3222
         Detection Rate : 0.1667
   Detection Prevalence : 0.2000
      Balanced Accuracy : 0.7340

       'Positive' Class : 1
```

Using the training and test data from the decision tree model, the confusion matrix above outputs the results of the trained random forest model on test data. There are 58 true negatives, 15 true positives, 3 false positives, and 14 false negatives. The random forest model had an accuracy of 0.8111111 accuracy. The sensitivity is 0.5172414 and the specificity is 0.9508197. The kappa value is 0.519774, which is a moderate agreement. In all performance statistics except for sensitivity, the random forest model performed better than the simple ensemble model.

Ensemble model evaluation

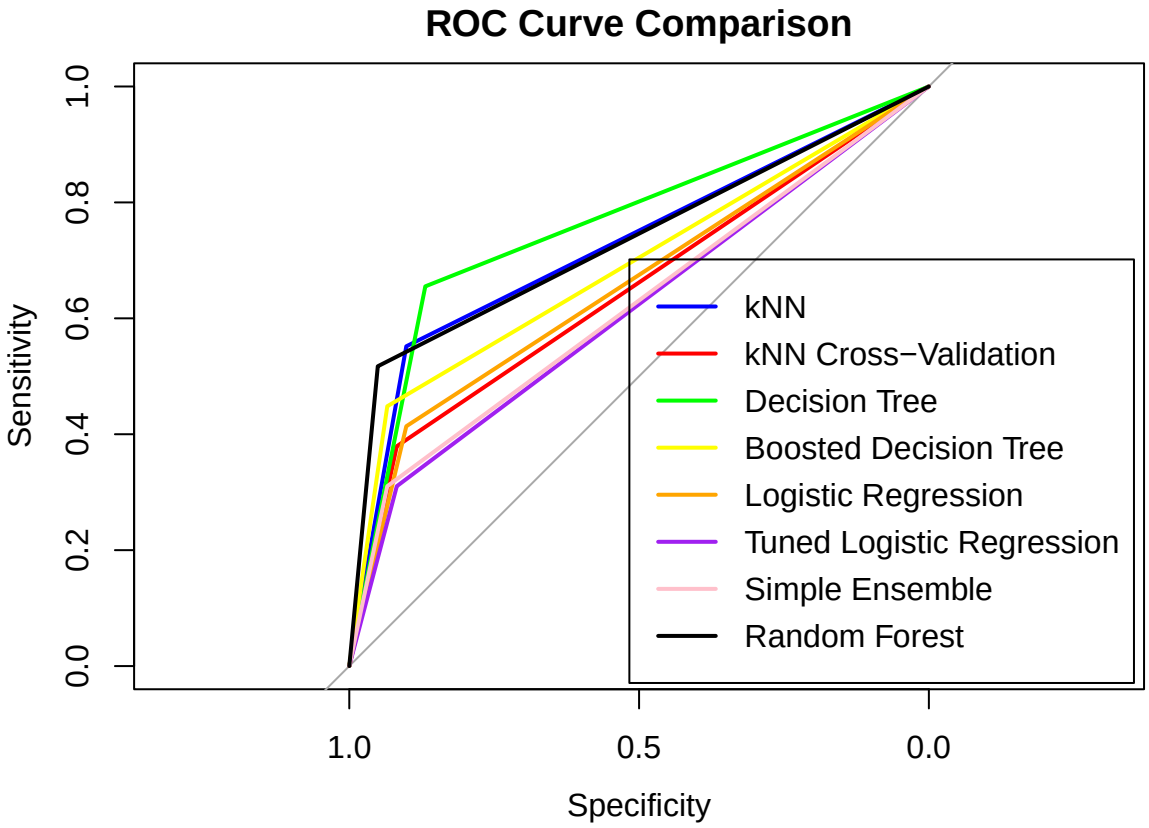
The simple ensemble model's performance is rather subpar. Though some may say an accuracy of 0.7333333 is still considered high, previous models were shown to have higher accuracy values and higher sensitivity values. Specifically, the sensitivity of the ensemble model is 0.6923077. This is in line with previous models and even triumphs the decision tree sensitivity of 0.6551724. Therefore, the ensemble model was an improvement in that it had higher accuracy and sensitivity compared to the logistic regression model and the kNN cross-validation model. However, it falls short as the best performing model when compared to the accuracy and specificity

values of the decision tree or kNN model. One thing to note is that the simple ensemble model had the lowest number of false negatives seen out of all models, but the highest number of false positives. This may indicate the classifier's tendency to overly predict positive cases.

On the other hand, the random forest model is a contender for one of the best performing models. It has the highest accuracy of 0.8111111 and second highest kappa value of 0.519774. Furthermore, this model has the least number of false negatives and higher F1 values seen out of all the models. Since sensitivity is one of the most important statistics for a model in predicting diagnoses, unfortunately this model's sensitivity of 0.5172414 is still not the most ideal despite performing well in comparison to the other models. Likewise, even if the random forest model has one of the higher kappa values, the kappa value indicates only a moderate agreement between predictions and actual values. A moderate agreement, sub-60% sensitivity, and only 80% accuracy are not sufficient enough for implementing the model in the real world.

Conclusion

Comparing models



The plot above shows the ROC curves of each model. An ideal model would have a ROC curve closer to the top left corner of the plot. The most curved lines are the the C5.0 decision tree, random forest, and kNN lines, indicating that these algorithms may be the best performing algorithms. However, looking at the curves themselves, all models showcase a slight curve instead of a more rounded one. This indicates that they are not close to the ideal as there is still quite a distance between the top corner and the ROC curves. Overall, the ROC curves support the conclusion that none of the models meet industry standards for implementation.

Table 3: Predictions of classification models

	TN	TP	FN	FP
Actual	61	29	-	-
kNN	55	16	13	6
kNN Cross Validation	56	11	18	5
C50 Decision Tree	53	19	10	8
C50 Boosted Decision Tree	57	13	16	4
Logistic Regression	55	12	17	6
Tuned Logisitic Regression	56	9	20	5
Simple Ensemble	57	9	4	20
Random Forest	58	15	14	3

Table 4: Performance statistics of classification models

	Accuracy	Sensitivity	Specificity	Kappa	Precision	F1
kNN	0.7888889	0.5517241	0.9016393	0.4840072	0.7272727	0.6274510
kNN Cross Validation	0.7444444	0.3793103	0.9180328	0.3369635	0.6875000	0.4888889
C50 Decision Tree	0.8000000	0.6551724	0.8688525	0.5336788	0.7037037	0.6785714
C50 Boosted Decision Tree	0.7777778	0.4482759	0.9344262	0.4292961	0.7647059	0.5652174
Logistic Regression	0.7444444	0.4137931	0.9016393	0.3502825	0.6666667	0.5106383
Tuned Logisitic Regression	0.7222222	0.3103448	0.9180328	0.2642250	0.6428571	0.4186047
Simple Ensemble	0.7333333	0.6923077	0.7402597	0.2861864	0.3103448	0.4285714
Random Forest	0.8111111	0.5172414	0.9508197	0.5197740	0.8333333	0.6382979

When comparing classification counts, it can be seen that the models were able to predict the majority of true negatives. This is reflected in their decently high specificity values as they range from 0.74 to 0.95. On the other hand, it can be seen that the models had more difficulty predicting the positive class as the number of true positive predictions were barely half the number of actual positives. This is supported by the lower range of sensitivity values ranging from 0.31 to 0.69.

Comparing total sum counts of false positives and false negatives, the C5.0 decision tree, kNN, and simple ensemble model had the lowest counts out of all the models. The C5.0 decision tree model had the highest number of true negative predictions as reflected in the high accuracy value. However, despite having lower counts of false positives and negatives, the 10 false negatives are still of a concern. Similarly, the kNN and simple ensemble models had lower counts of false negatives but still a significant amount of false positives that offset them. False negatives and false positives have to be minimized, especially in a model that is used to predict major health events in a real world setting.

Concerning performance statistics, the ideal model has accuracy, sensitivity, specificity, and kappa values close to 1. Looking at the results, the C5.0 decision tree model or the random forest model come closest to the ideal. The decision tree and random forest model had the higher accuracy values of 0.8 and 0.8111111 respectively and the higher sensitivity values of 0.6551724 and 0.5172414 respectively. Likewise, they also have the highest F-scores with a value of 0.6785714 and 0.6382979 for the decision tree and random forest models respectively. A higher F-score indicates that the model is more precise and robust.

Additionally, the kappa values can also be interpreted to determine the effectiveness of the classification models. Cohen's kappa is more applicable in interpreting the results of the data due to the large unbalanced classes of the target variable (Lantz, 2023). Kappa values decreased when improvement in terms of cross-validation, boosting, or hyperparameter tuning were made to the 3 main models. The range of kappa values from the various models result in "fair to moderate agreement" according to common interpretations, with the decision tree and random forest having the highest kappa values. However, a "fair to moderate" agreement does not suffice when the purpose of the model is predicting the survival of a heart failure patient.

If one model were to be chosen as the best performing algorithm, the C5.0 decision tree model would be the best model. In concern to performance statistics, the decision tree model ranks consistently high in accuracy, sensitivity, kappa, and F1 values. The decision tree model performs decently well in regards to specificity and precision. Other models fall short in certain statistic and are offset by some improvements in other areas. However, other models have statistics that are not consistently high enough to prove merit as the best model.

Real-world application

In a real world setting, the performance of these various models are not enough for implementation in hospital. Overall accuracy is decently high, but not near 100%. Additionally, low sensitivity values and only fair to moderate kappa values are not standard for being the ideal model. Likewise, false negatives are a major concern. Lives are on the line so sensitivity rates must be high enough in order to lower the number of false negatives. A false negative would be detrimental as it results in misdiagnosing a patient as healthy when in reality the patient is not. Nonetheless, the models previously demonstrated can be utilized as a good starting point for constructing a more accurate and ideal model.

Some potential avenues of exploration in order to improve performance statistics would be to gather a large data set. With more data, a model could be able to learn the more subtle patterns and make more accurate predictions. Likewise, other classification algorithms such as k-means clustering or support vector machines could result in better performance statistics.

References

Chugh, V. (2023, March 17). Logistic regression in R tutorial. DataCamp. https://www.datacamp.com/tutorial/logistic-regression-R

Heart Failure Clinical Records. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5Z89R.

Lantz, B. (2023). Machine Learning with R - Fourth Edition. Packt Publishing.