edx Capstone Project 2: Heart Failure Project submission

Abideen Sadiq

09/04/2021

Abstract

Heart failure means that the heart is unable to pump blood around the body properly. It usually occurs because the heart has become too weak or stiff. Heart failure is a long-term condition that tends to get gradually worse over time. If we can identify the key metrics that lead to heart failure, then we can potentially provide healthcare and solutions proactively before the event occurs.

Introduction

This report aims to construct a Machine Learning model which will predict whether a person will suffer fatal heart failure upon their next hospital visit. The data being used here is the Heart Failure Dataset on Kaggle.

For this report, the Heart Failure dataset is split into a training and a validation set (`train` and `validation` respectively). Only the `train` data set is used for model construction. The `validation` data set is used only for assessing the performance of the *final* model. `train` is split into `heart_train` and `heart_test`. Various models are constructed using `heart_train` and their performances are assessed using `heart_test`. The best performing model is then retrained using `train` and assessed using `validation`. This way, `validation` has no effect on which model is selected to be the final model. The R code used to construct these data sets, models and plots is available in this GitHub repo.

`validation` is 20% of the entire data set and `heart_test` is 20% of `train`. The reason 20% is used for testing and validating in this report is because the data set is quite small. Using 20% instead of 10% for example gives more data to assess the performance of the models.

Data Exploration

The structure of the `train` dataset is shown below. “PD” is the predictor variable: “Yes” indicates the patient has passed away.

The features are made up of age, gender (biological sex), a selection of conditions (including anaemia, high blood pressure and diabetes), as well as measurements of fluids such as creatinine phosphokinase, sodium and creatinine. The data contains observations from 299 patients, 32% of which have passed. It’s hard to tell whether this is reflective of the true mortality rate for heart failure as the rate differs over time.

```
## 'data.frame':    299 obs. of  12 variables:
##  $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ anaemia                 : Factor w/ 2 levels "Yes","No": 2 2 2 1 1 1 1 1 2 1 ...
##  $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
##  $ diabetes                : Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 1 2 2 ...
##  $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "Yes","No": 1 2 2 2 2 1 2 2 2 1 ...
##  $ platelets               : num  265000 263358 162000 210000 327000 ...
##  $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
##  $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
##  $ sex                     : Factor w/ 2 levels "Male","Female": 1 1 1 1 2 1 1 1 2 1 ...
##  $ smoking                 : Factor w/ 2 levels "Yes","No": 2 2 1 2 2 1 2 1 2 1 ...
##  $ PD                      : Factor w/ 2 levels "Yes","No": 1 1 1 1 1 1 1 1 1 1 ...
```

The images below show a correlation plot of the factor and continuous features respectively. The non-continuous correlations are left blank:

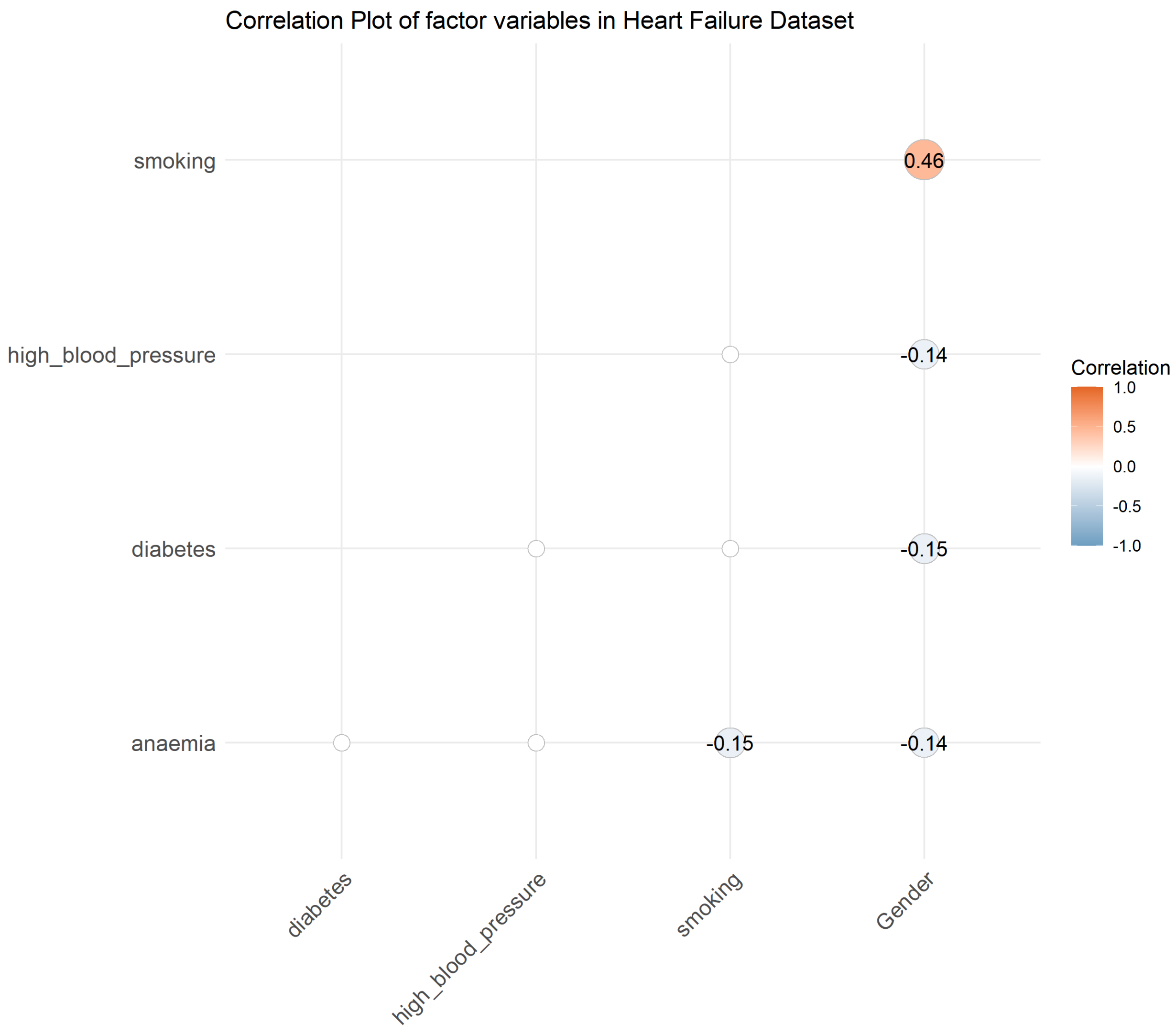

Figure 1: Correlation plot of factor variables in the Heart Failure dataset.

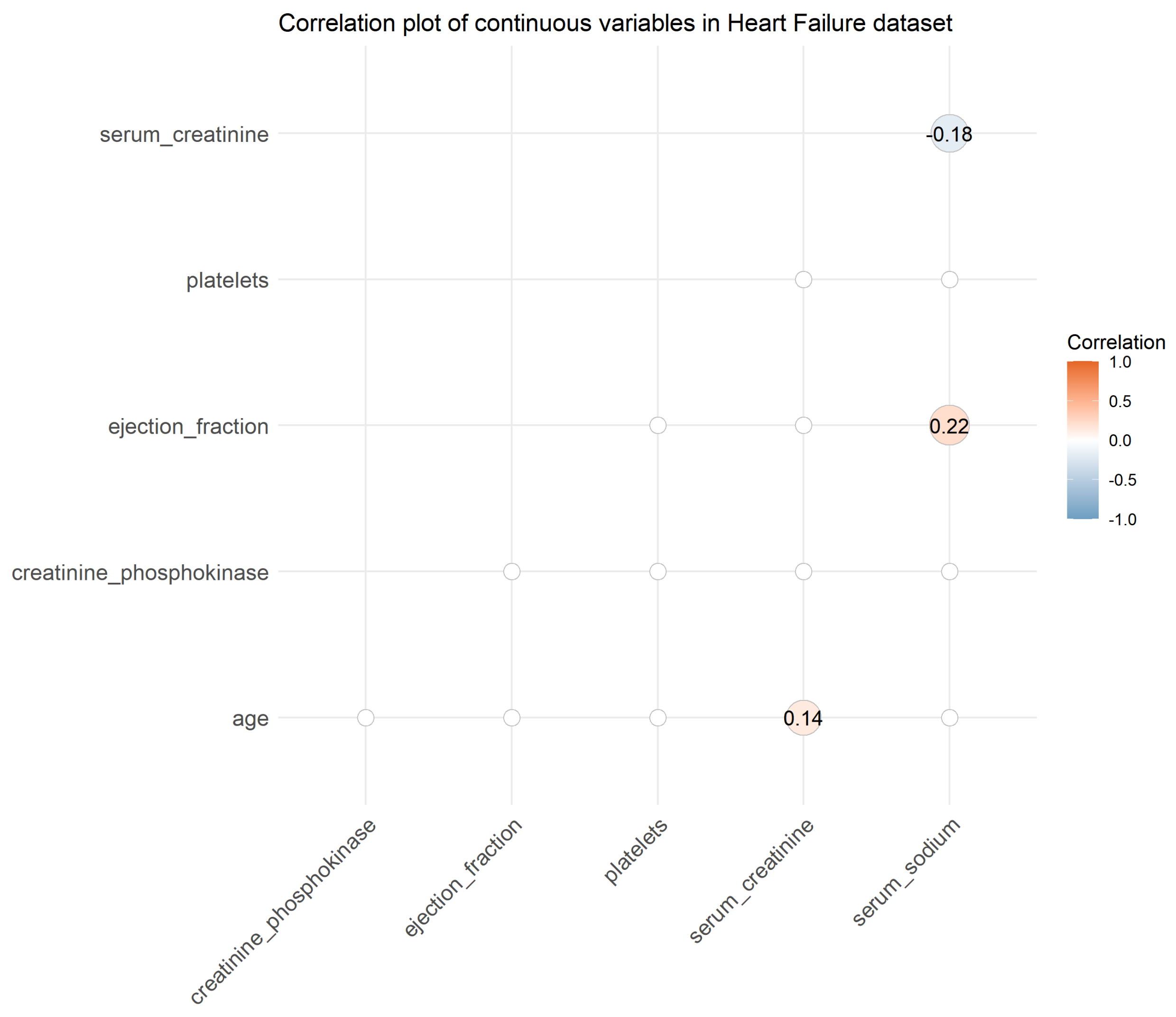

Figure 2: Correlation plot of factor variables in the Heart Failure dataset.

We will now look at each variable in turn then look at the significant relationships between some of the variables as shown in the correlation plots.

Starting off with Gender: The death rates between genders are equal:

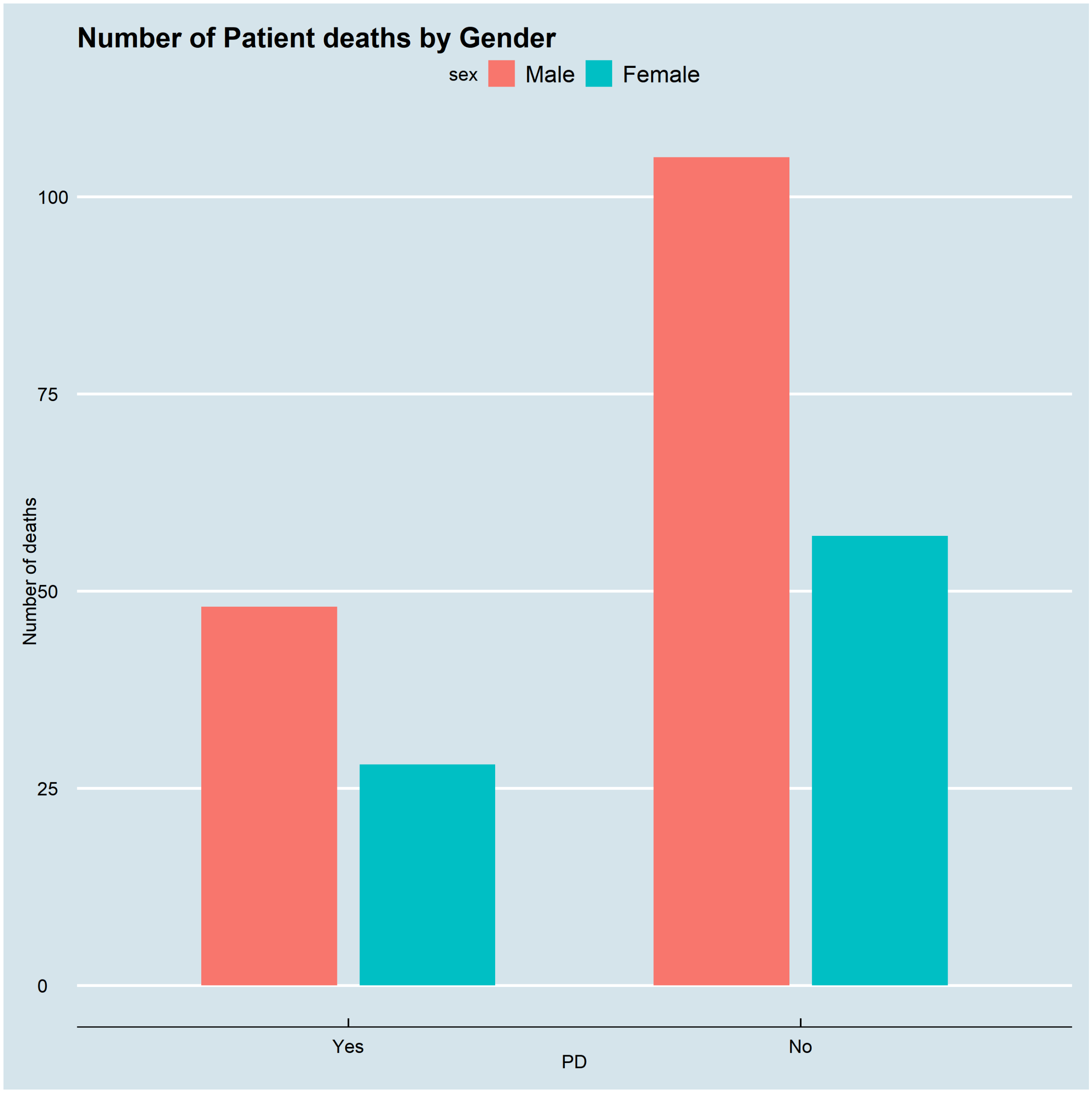

Figure 3: Number of Patient deaths by Gender

The distribution of both survivals/deaths by age seem to follow a similar pattern, with the majority of patients being 60-65 years old. However, there are more people past this age-range that have died than surived in this dataset:

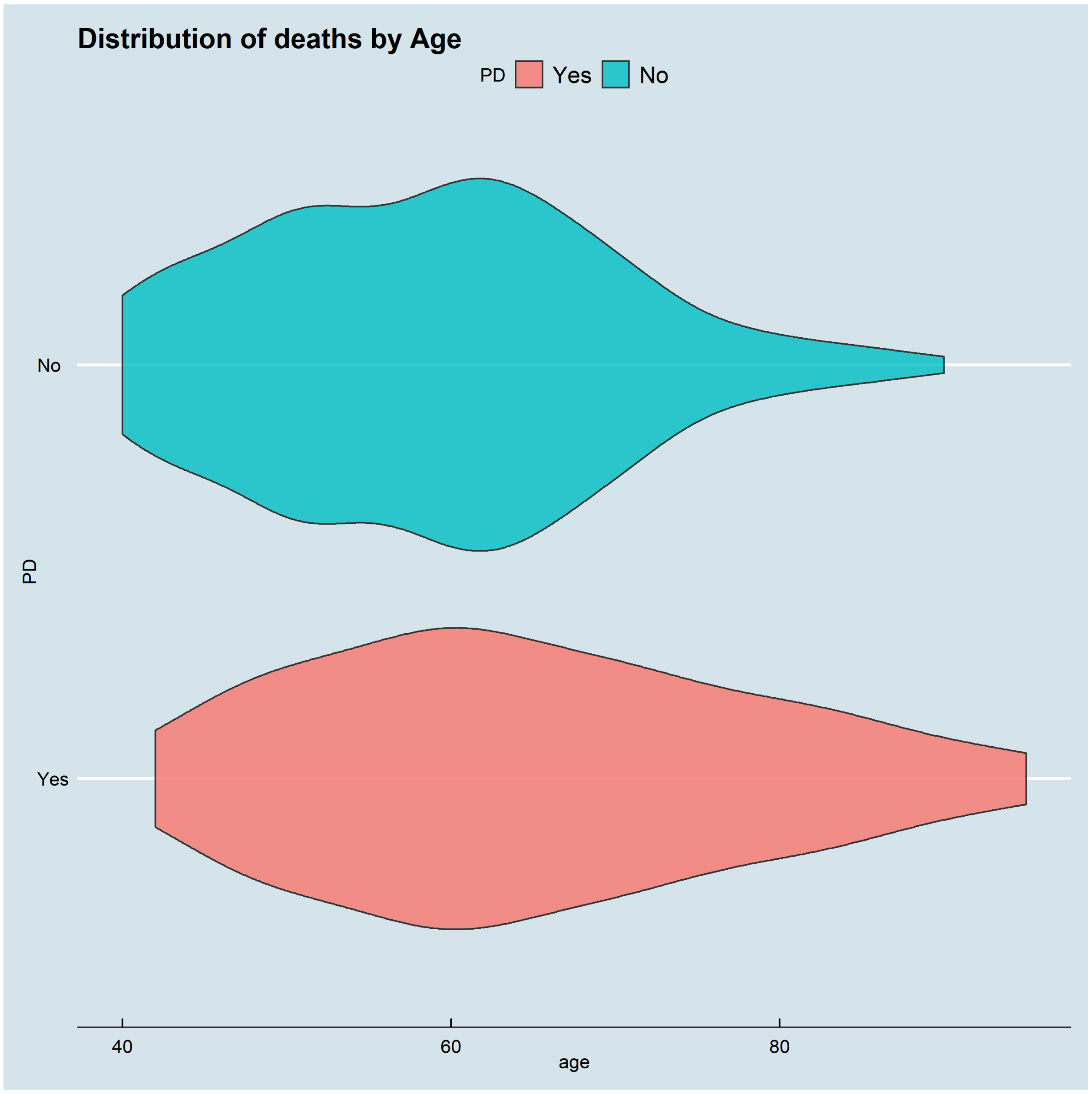

Figure 4: Distribution of deaths by Age.

There doesn’t seem to be much of a difference of creatinine phosphokinase levels between alive/dead patients:

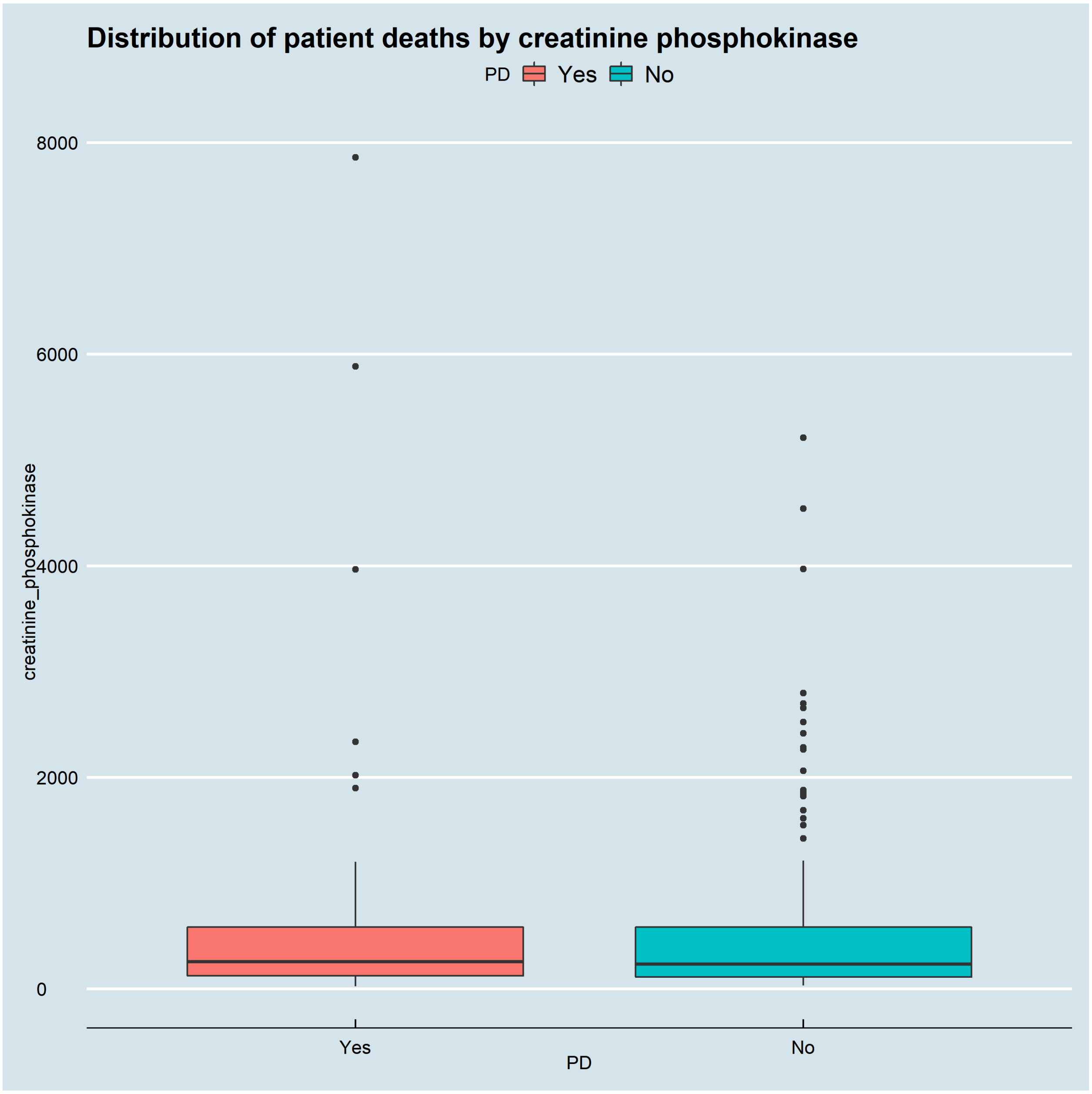

Figure 5: Distribution of patient deaths by creatinine phosphokinase.

Those who die have a lower ejection fraction on average than those who have survived:

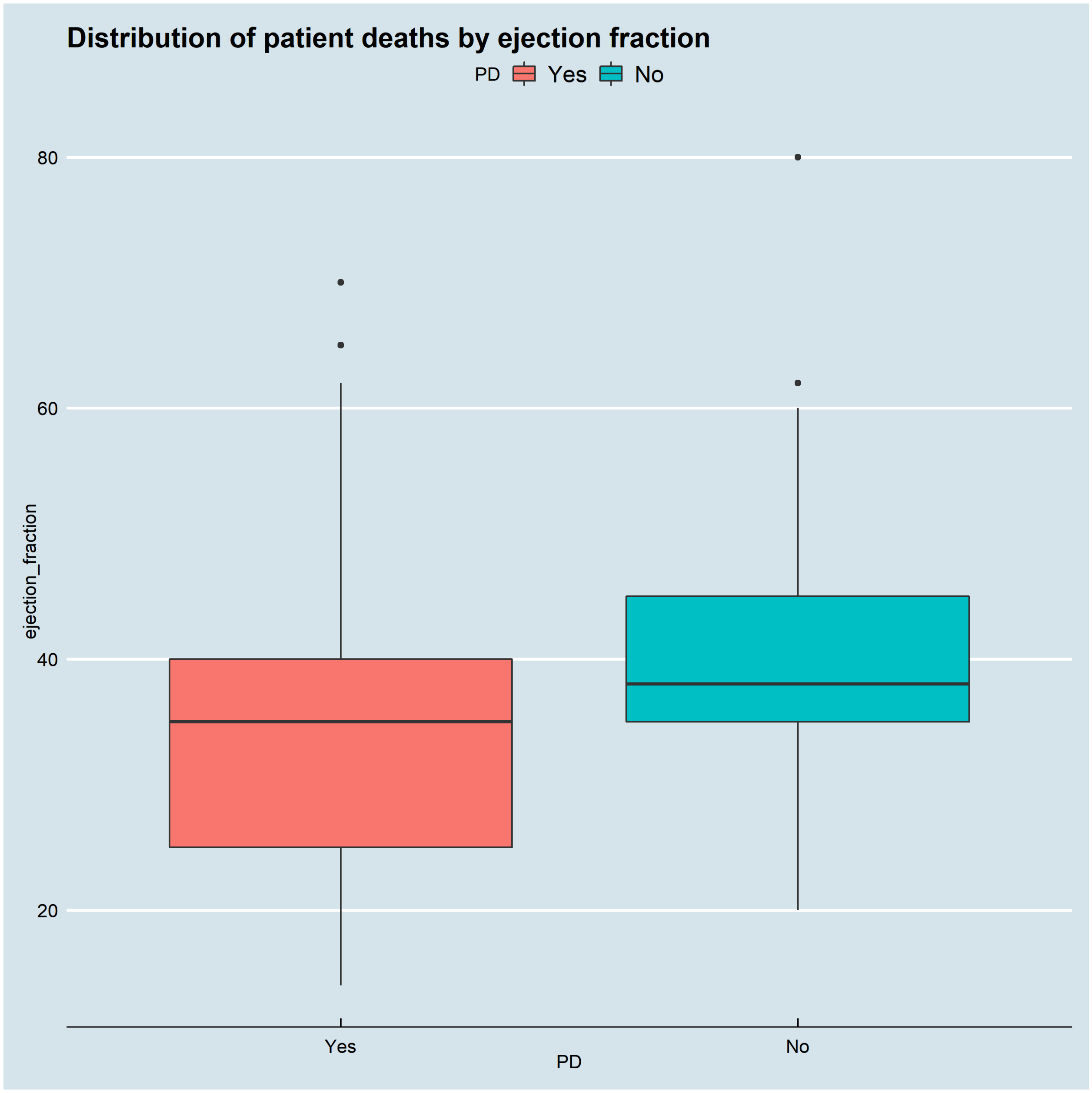

Figure 6: Distribution of patient deaths by ejection fraction.

The platelet count does not appear to be a significant factor in survival:

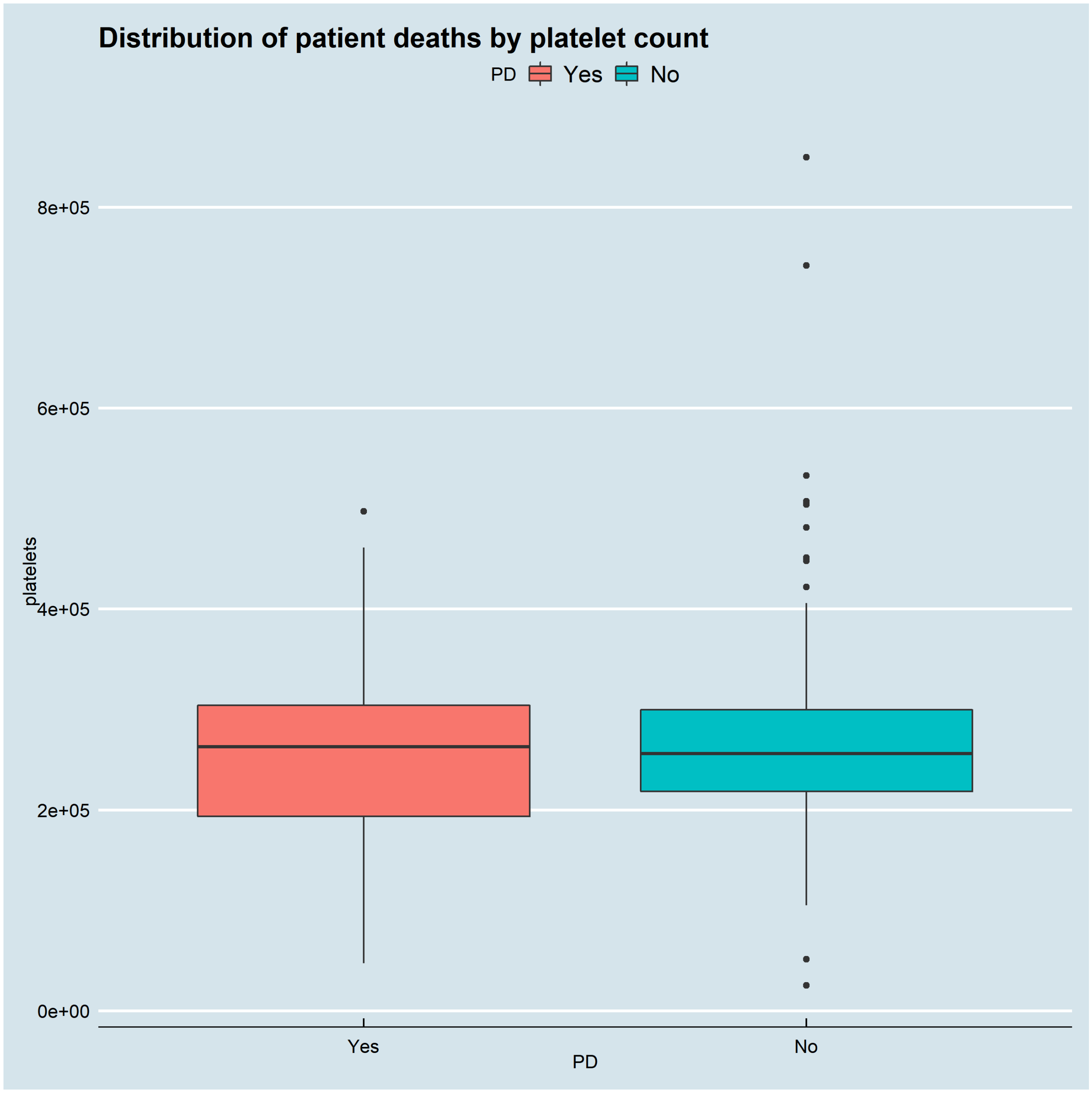

Figure 7: Distribution of patient deaths by platelet count.

The levels of creatinine are higher in those who die:

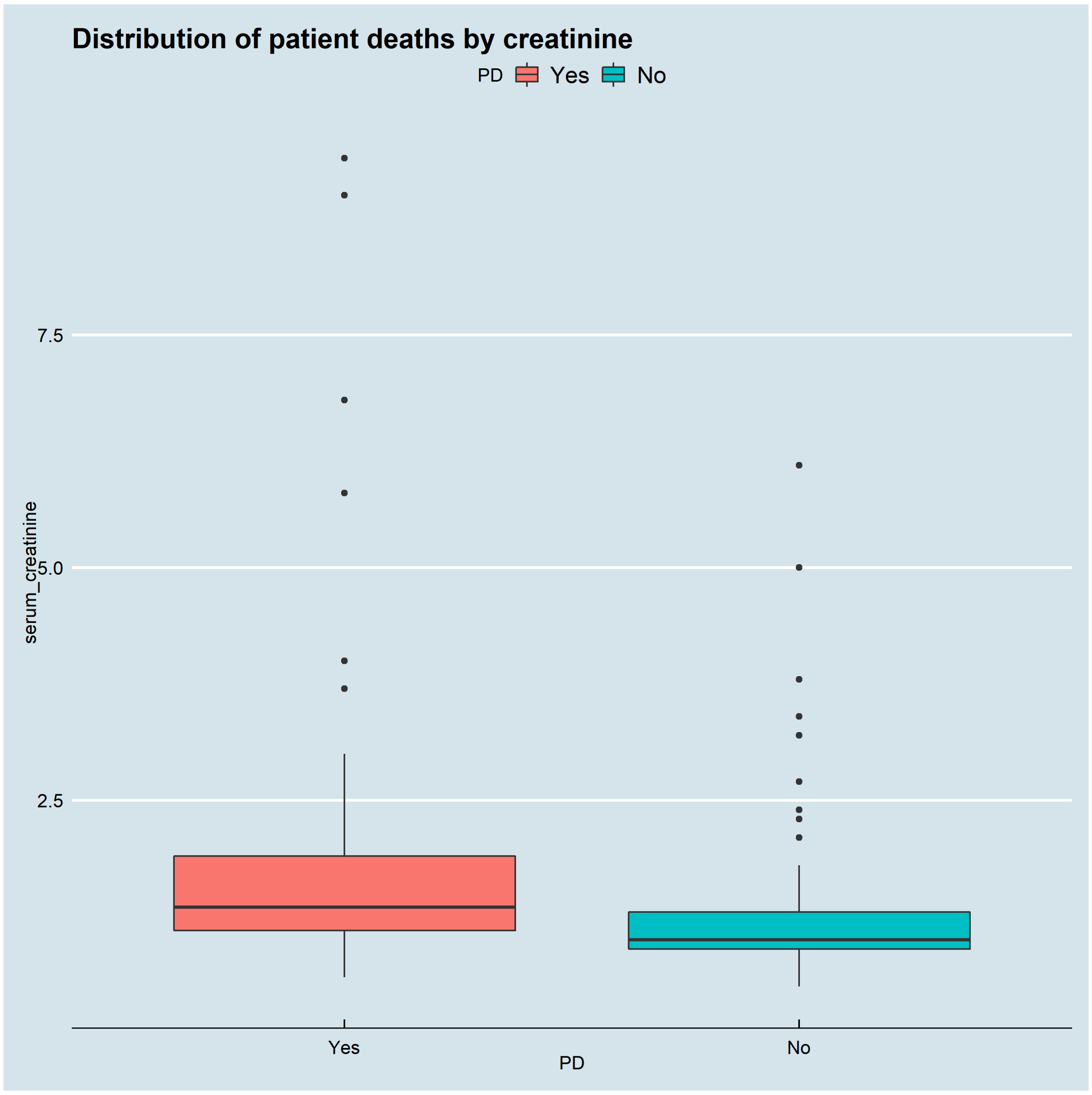

Figure 8: Distribution of patient deaths by creatinine.

The level of sodium is lower on average in those who die:

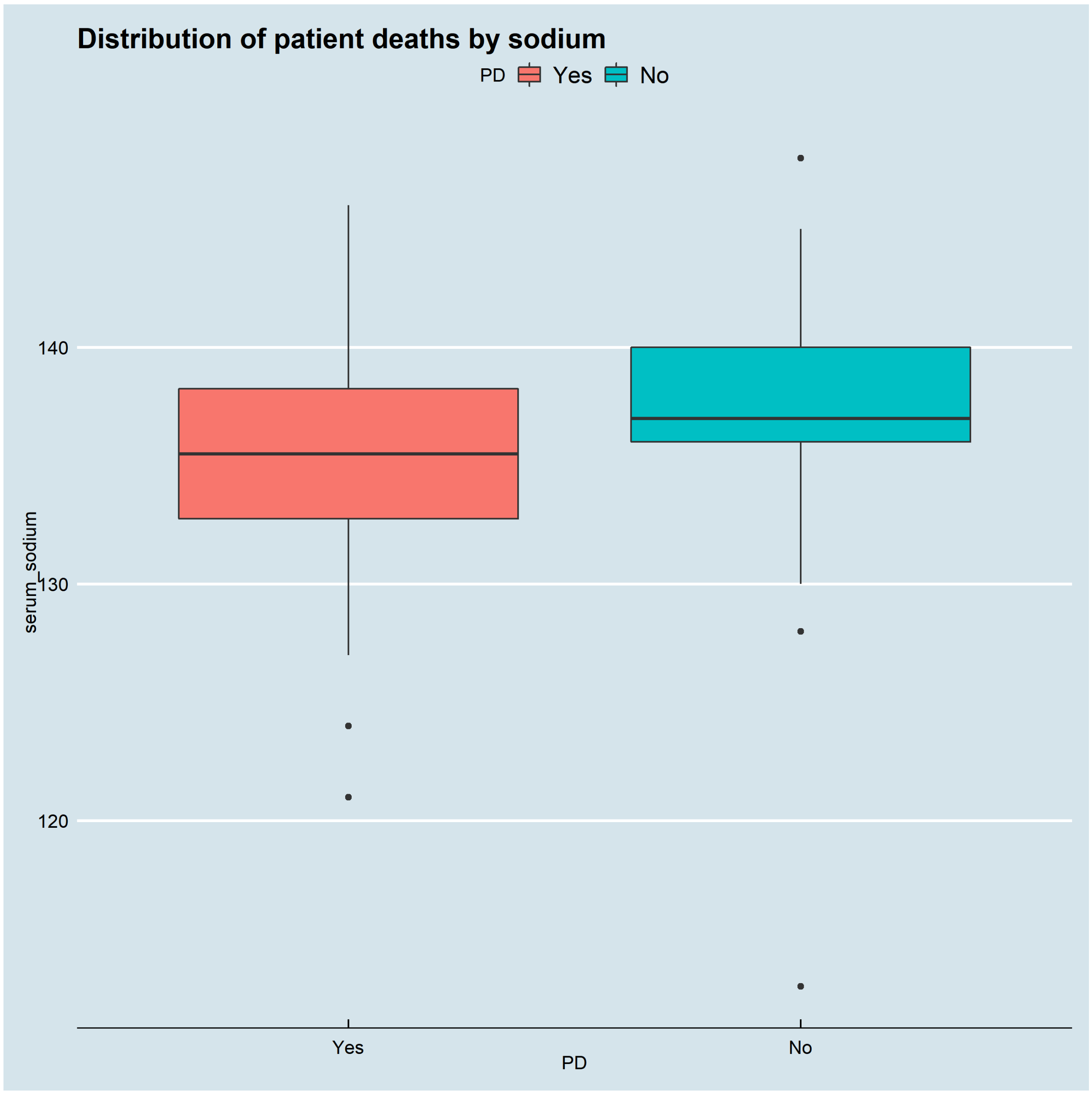

Figure 9: Distribution of patient deaths by sodium.

Now looking at the significant correlations between explanatory variables.

Women with diabetes are more likely to die than men with diabetes, but men are more likely to die overall:

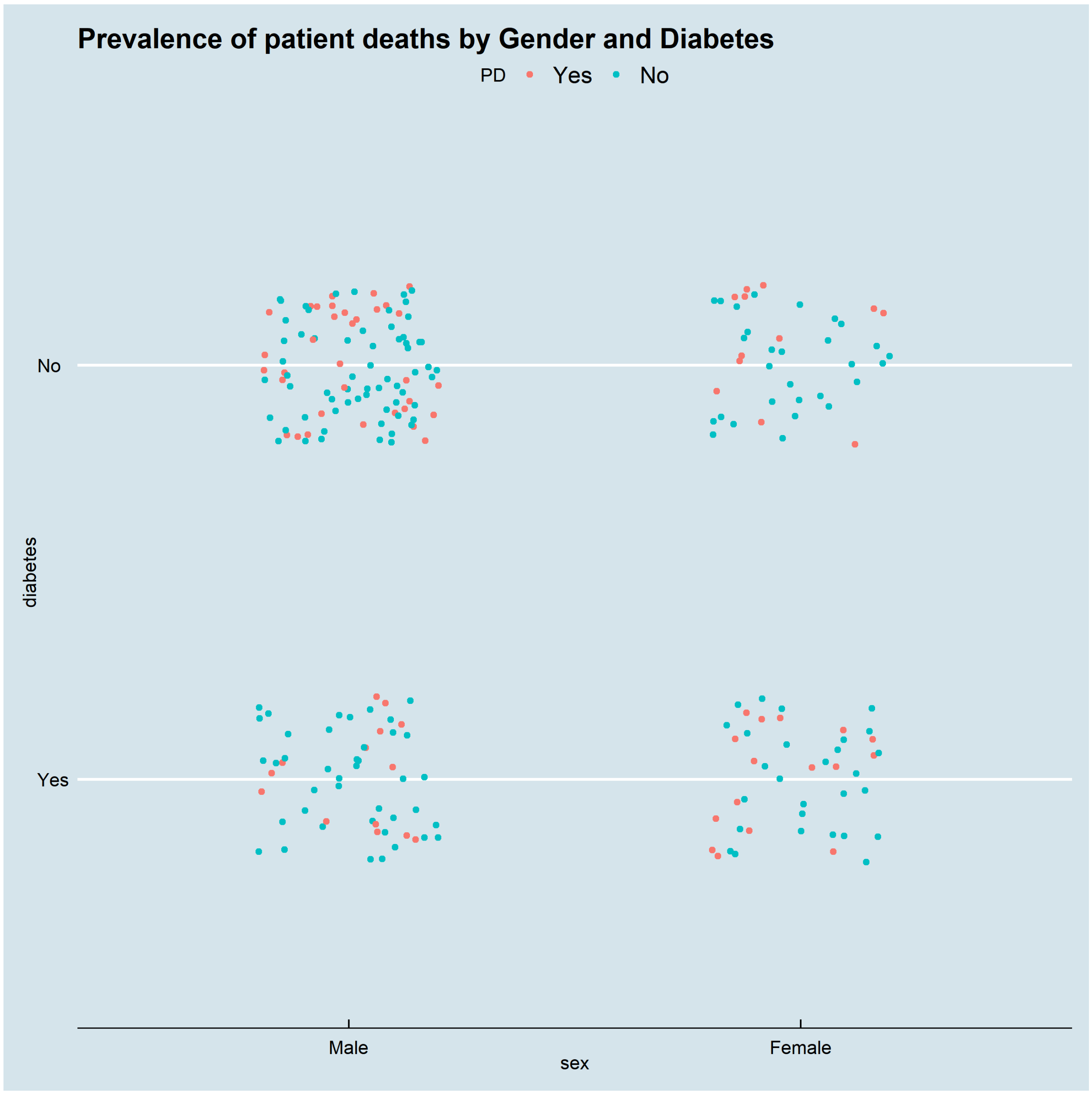

Figure 10: Gender-Diabetes relationship.

Women who smoke more likely to die than men who smoke (very small sample size, however):

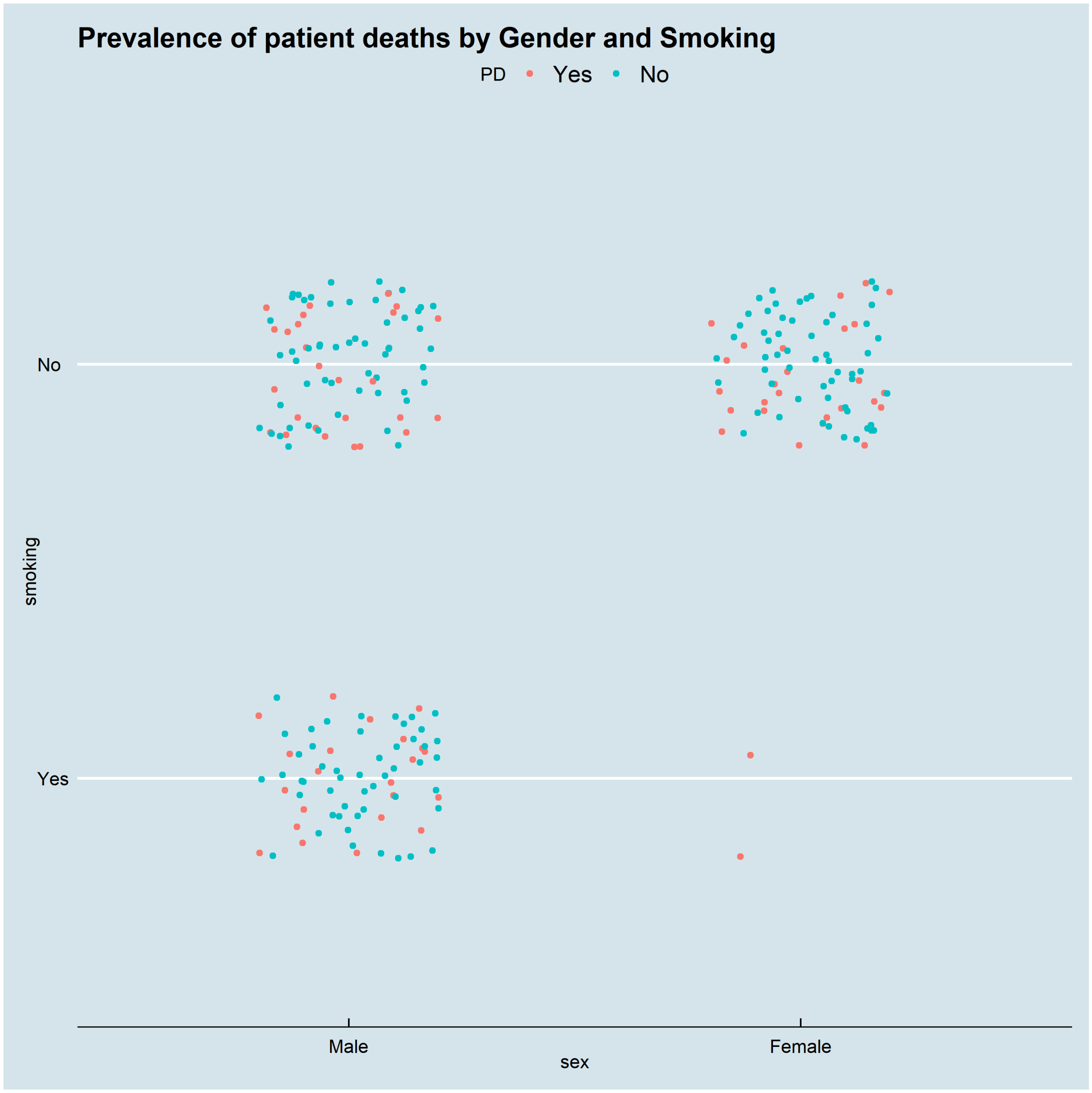

Figure 11: Gender-smoking relationship.

Those with diabetes who also smoke are at most risk:

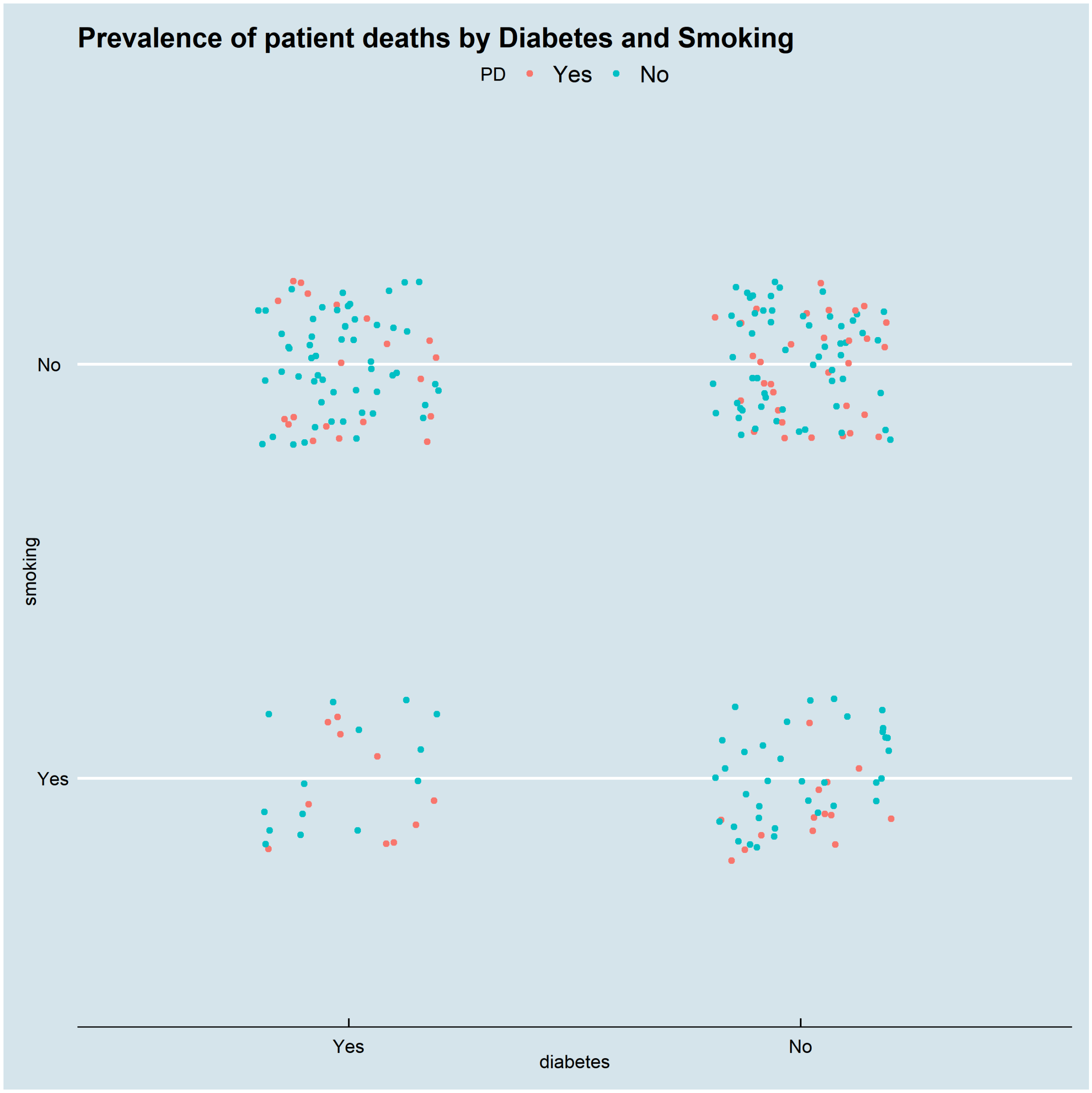

Figure 12: Smoking-Diabetes relationship.

Not a strong relationship between sodium levels and age - correlation likely due to outliers:

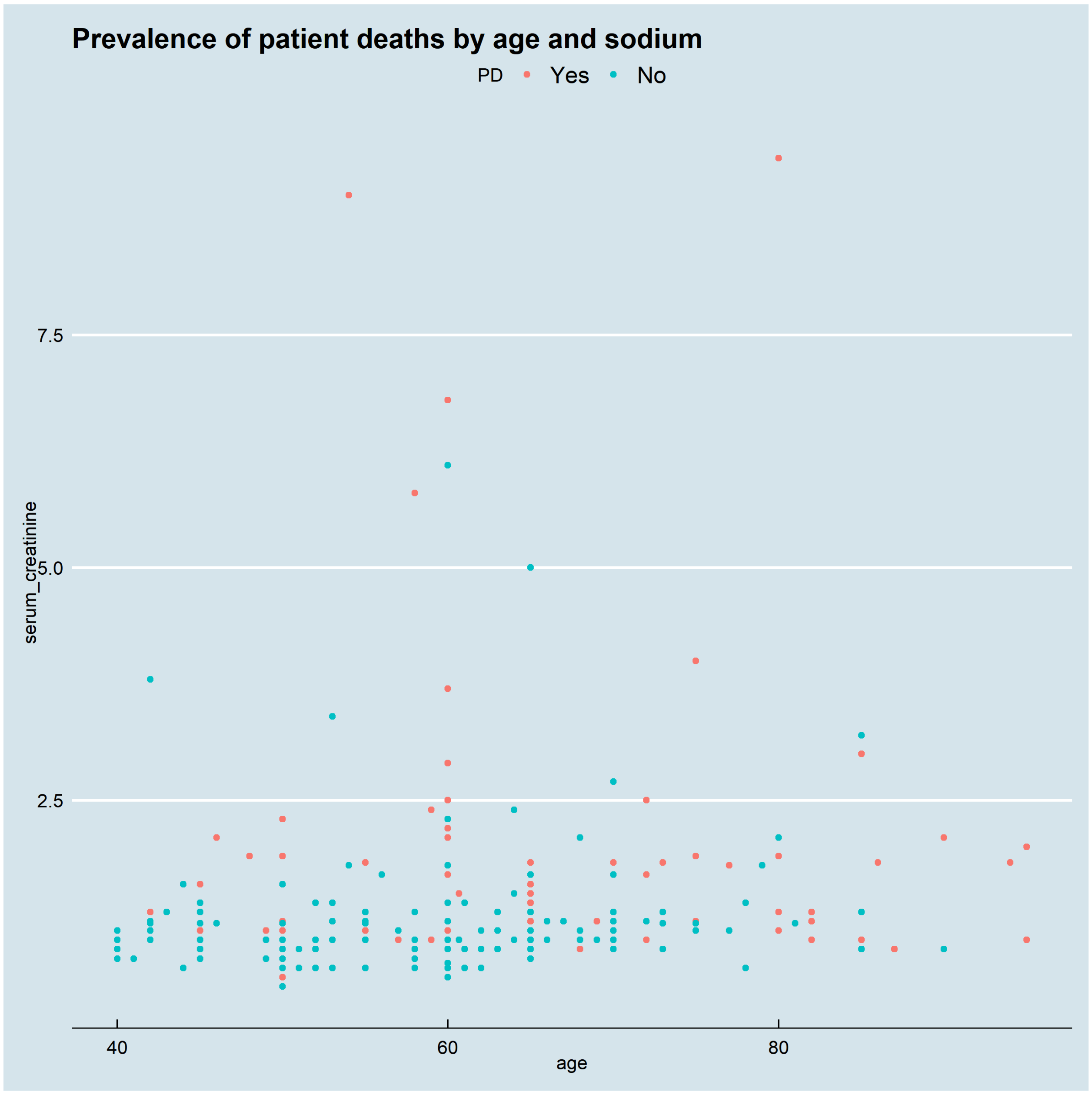

Figure 13: Creatinine-Age relationship.

Also a weak correlation between levels of sodium and creatinine:

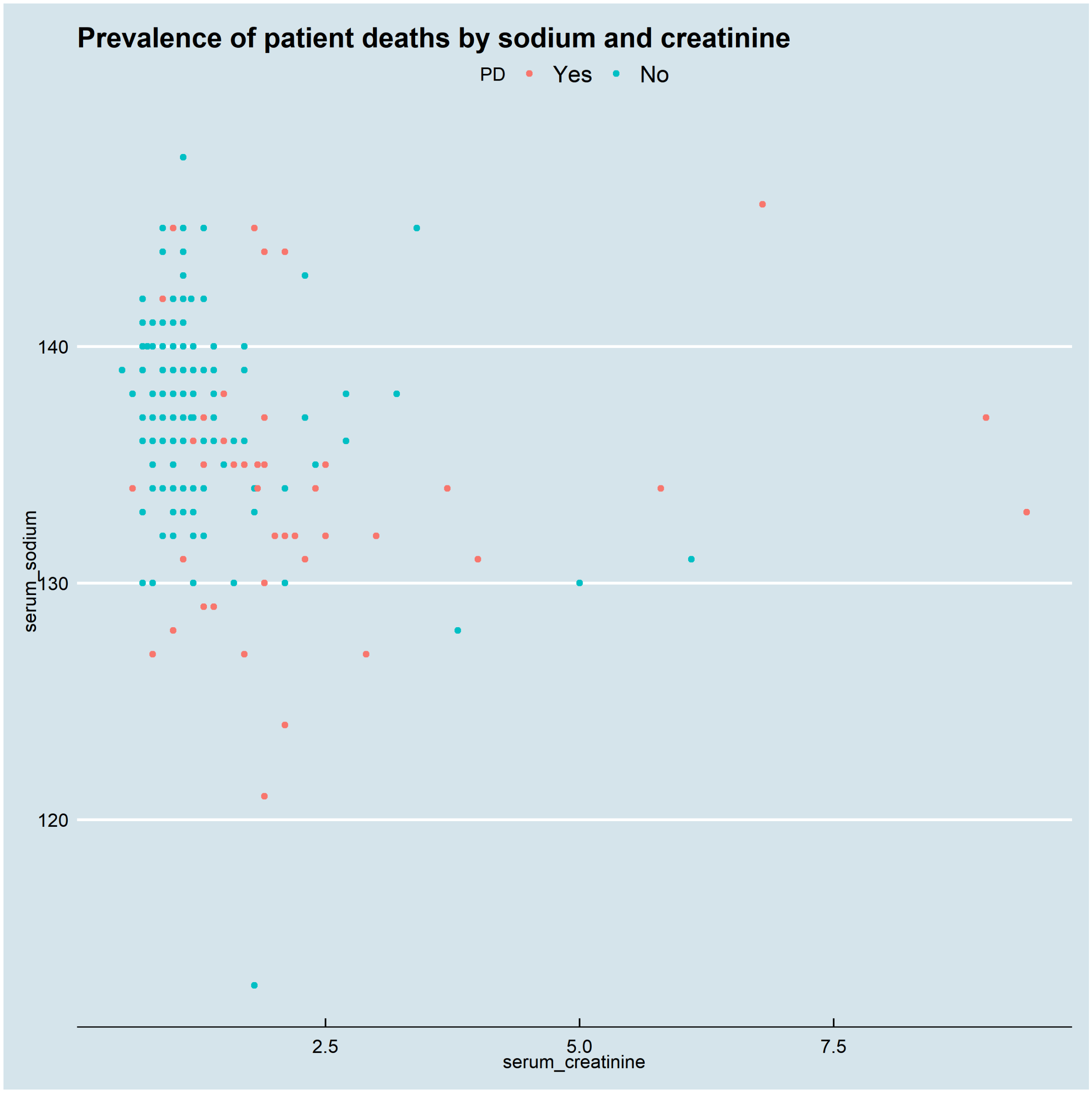

Figure 14: Sodium-Creatinine relationship

There is less sodium in the blood of those who die, this could be due to a weak heart not being able to pump blood efficiently around the body, as the plot below suggests:

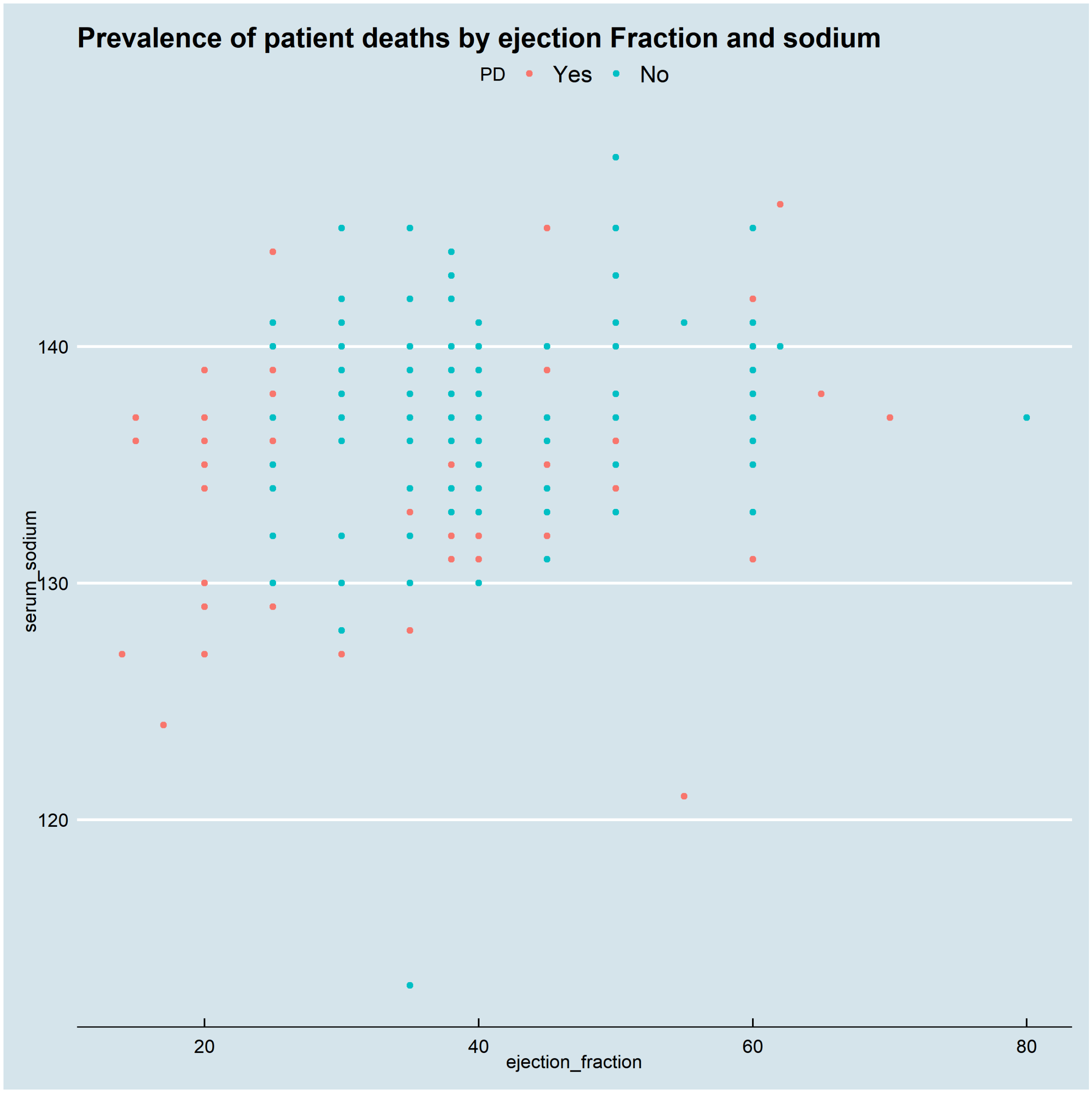

Figure 15: Sodium-Ejection fraction relationship.

Machine Learning Overview: Performance Measurement

We will focus on two metrics to assess the performance of our machine learning models:

1) **Accuracy**, which is the proportion of correctly classified events.
2) **Sensitivity**, which is the proportion of 'positive events' (i.e. the event of importance) that are correctly classified.In this exercise, the positive event is a patient passing away (PD = 1).

This report aims to maximise the mean of the accuracy and the sensitivity. 10-fold cross-validation is used to choose the ideal hyperparameters (where applicable) which maximise the mean of the accuracy and the sensitivity.

Approach 1: Logistic Regression

The first model we'll build will be a logistic regression model. Logistic Regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable Y. For our dataset, the dependent variable is PD.

We assume a linear relationship between the explanatory variables and the log-odds (also called logit) of the event that $Y = 1$. This linear relationship can be written in the following mathematical form (where l is the log-odds, b is the base of the logarithm, and β_i are parameters of the model):

$$l = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{1}$$

One choice that has to be made when constructing a logistic regression model is what cutoff to use. The cutoff p is such that $\hat{\pi}_i > p \Rightarrow$ observation i is classed as positive. A typical choice is 0.5, however the context of this problem gives reason to consider a value lower than 0.5. Not identifying a person at risk of heart failure is more costly than incorrectly classifying someone who is not at risk.

10-fold cross-validation is used to find the optimal p. The figure below shows the results for various cutoffs:

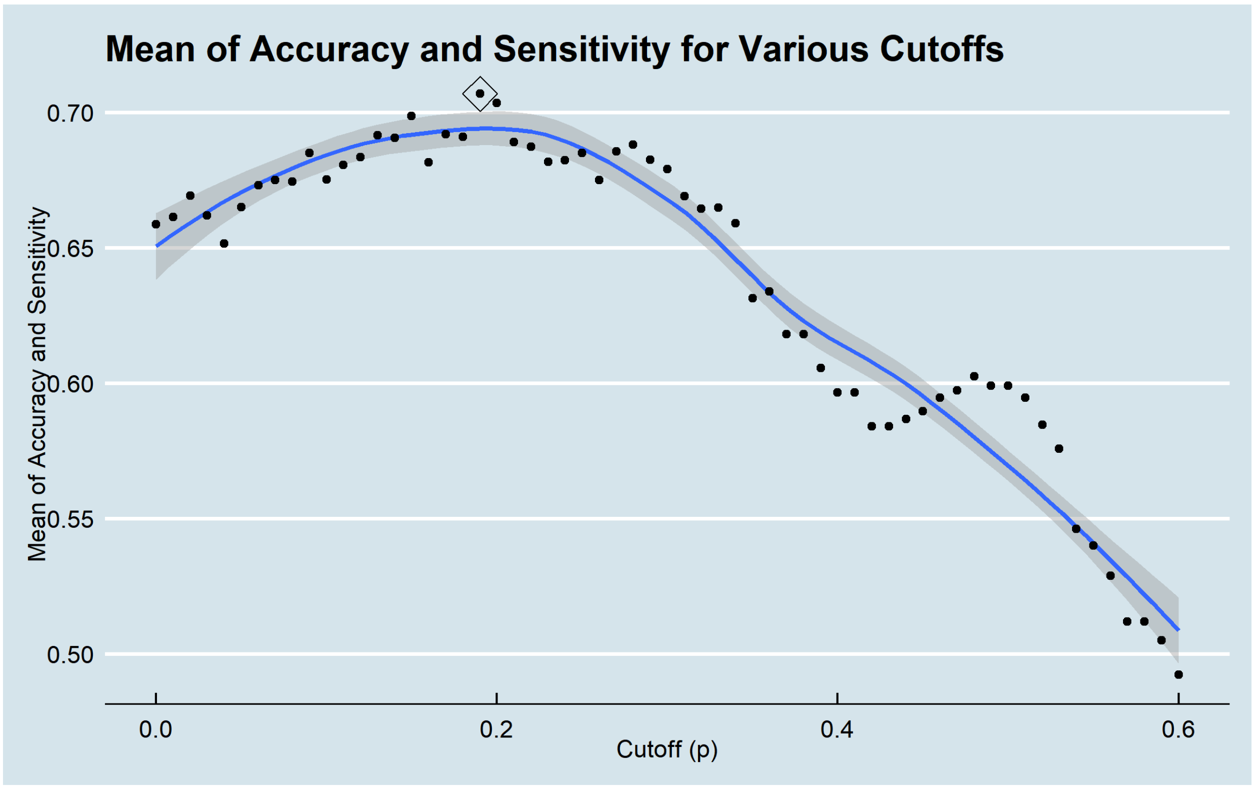

Figure 16: Cross validation results for logistic regression model. Optimal p is 0.19.

The summary of the model, trained on `heart_train` is shown below:

```
##
## Call:
## glm(formula = as.numeric(PD == "Yes") ~ ., family = binomial(logit),
##     data = heart_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1843  -0.7686  -0.5043   0.8175   2.4026
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               2.857e+00  5.648e+00   0.506 0.613031
## age                       6.465e-02  1.652e-02   3.914 9.08e-05 ***
## anaemiaNo                -3.765e-01  3.807e-01  -0.989 0.322673
## creatinine_phosphokinase  1.662e-04  1.873e-04   0.887 0.374876
## diabetesNo               -1.930e-01  3.726e-01  -0.518 0.604505
## ejection_fraction        -6.255e-02  1.863e-02  -3.357 0.000788 ***
## high_blood_pressureNo    -1.995e-01  3.923e-01  -0.508 0.611155
## platelets                -1.508e-06  1.920e-06  -0.785 0.432161
## serum_creatinine          6.013e-01  1.837e-01   3.274 0.001061 **
## serum_sodium             -3.827e-02  4.125e-02  -0.928 0.353574
## sexFemale                 4.733e-01  4.289e-01   1.103 0.269837
## smokingNo                -5.938e-01  4.515e-01  -1.315 0.188452
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 236.23  on 188  degrees of freedom
## Residual deviance: 190.43  on 177  degrees of freedom
## AIC: 214.43
##
## Number of Fisher Scoring iterations: 5
```

The summary indicates that a lot of the features aren't statistically significant (p value > 0.05). We will try training the model again, but only using features that are deemed statistically significant:

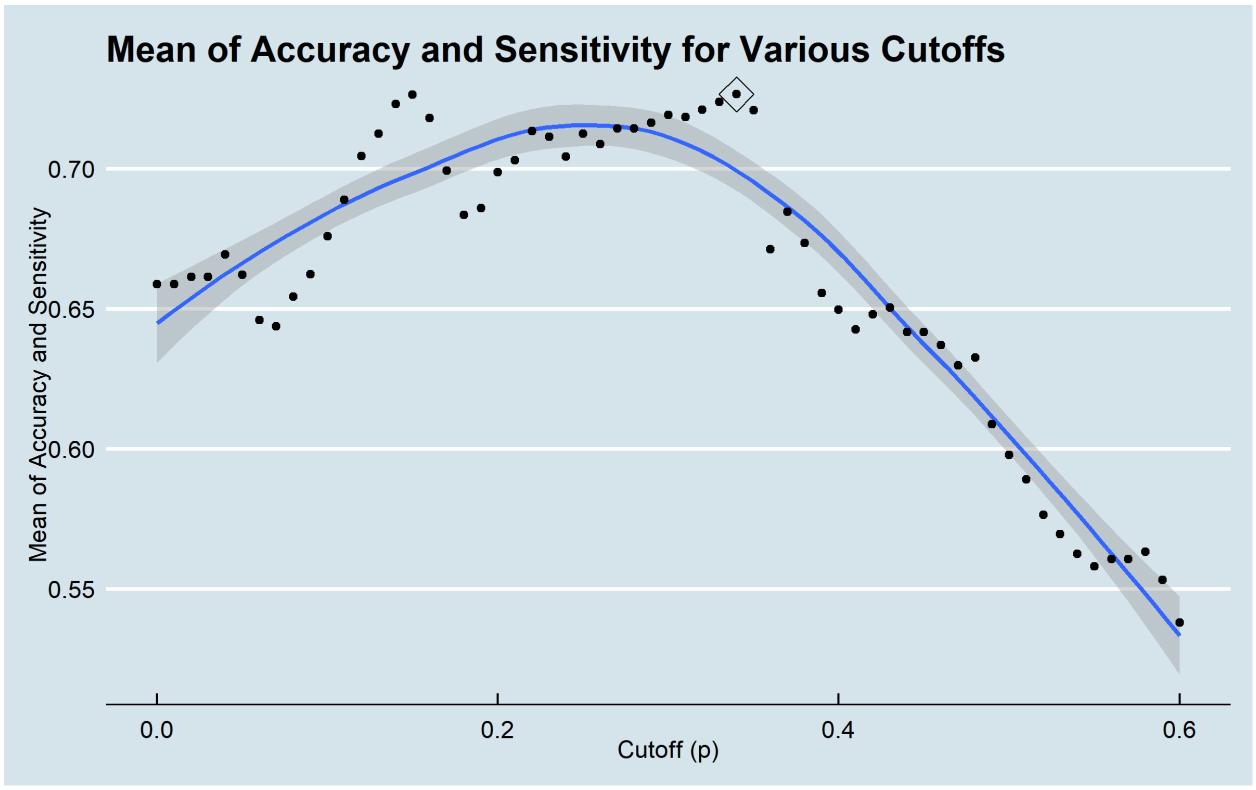

Figure 17: Second cross validation results for logistic regression model. Optimal p is 0.34.

The summary of the refined model is shown below:

```
##
## Call:
## glm(formula = as.numeric(PD == "Yes") ~ age + ejection_fraction +
##     serum_creatinine, family = binomial(logit), data = heart_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1744  -0.7833  -0.5360   0.8805   2.3799
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.94135    1.04687  -2.810 0.004960 **
## age                0.05830    0.01519   3.838 0.000124 ***
## ejection_fraction -0.06202    0.01776  -3.492 0.000479 ***
## serum_creatinine   0.59358    0.17127   3.466 0.000529 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 236.23  on 188  degrees of freedom
## Residual deviance: 195.59  on 185  degrees of freedom
## AIC: 203.59
```

```
##
## Number of Fisher Scoring iterations: 4
```

The confusion matrix below indicates that the glm model performs significantly better than our original model:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes   8  3
##        No    8 30
##
##                Accuracy : 0.7755
##                  95% CI : (0.6338, 0.8823)
##     No Information Rate : 0.6735
##     P-Value [Acc > NIR] : 0.0822
##
##                   Kappa : 0.4449
##
##  Mcnemar's Test P-Value : 0.2278
##
##             Sensitivity : 0.5000
##             Specificity : 0.9091
##          Pos Pred Value : 0.7273
##          Neg Pred Value : 0.7895
##              Prevalence : 0.3265
##          Detection Rate : 0.1633
##    Detection Prevalence : 0.2245
##       Balanced Accuracy : 0.7045
##
##        'Positive' Class : Yes
##
```

The accuracy & sensitivity of our first model is shown in the table below:

Table 1: Results after construction of the first model.

Method	Accuracy	Sensitivity	Mean
Logistic Regression	0.7755102	0.5	0.6377551

Approach 2: Naive Bayes

Naive Bayes is a simple technique for constructing classifiers: Classifiers are models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.

There is not a single algorithm for training such classifiers, but a family of algorithms are based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

Abstractly, Naïve Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities:

$$p(C_k \mid x_1, \dots, x_n) \tag{2}$$

for each of k possible outcomes or classes C_k.The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \tag{3}$$

10-fold cross-validation is used to assess which naive bayes model provides the highest accuracy. The figure below shows that the non-parametric model performs much better than the Gaussian Naive Bayes model:

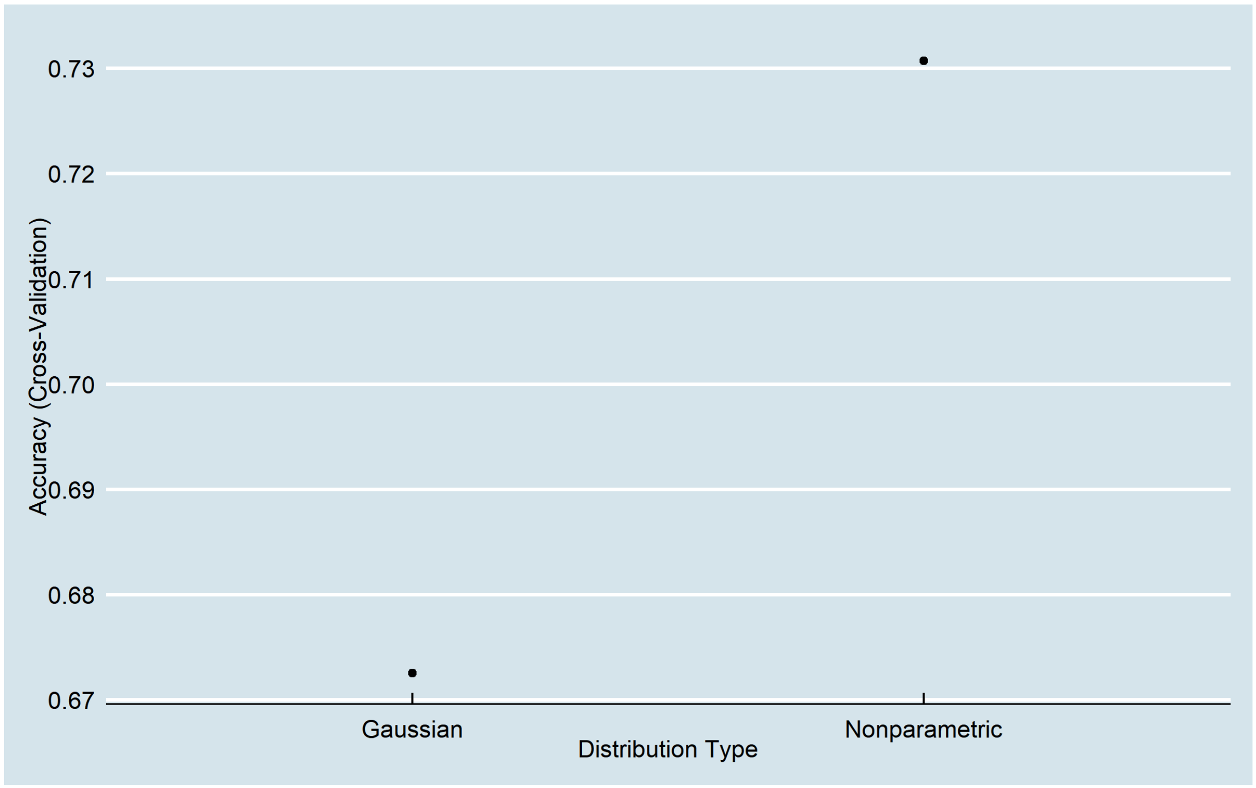

Figure 18: Cross validation results for Naive Bayes model. Optimal model is the Nonparametric model.

The confusion matrix for this model is shown below:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes   8  3
##          No    8 30
##
##                Accuracy : 0.7755
##                  95% CI : (0.6338, 0.8823)
##     No Information Rate : 0.6735
##     P-Value [Acc > NIR] : 0.0822
##
##                   Kappa : 0.4449
##
##  Mcnemar's Test P-Value : 0.2278
##
##             Sensitivity : 0.5000
##             Specificity : 0.9091
##          Pos Pred Value : 0.7273
##          Neg Pred Value : 0.7895
##              Prevalence : 0.3265
##          Detection Rate : 0.1633
##    Detection Prevalence : 0.2245
##       Balanced Accuracy : 0.7045
##
##        'Positive' Class : Yes
##
```

The accuracy & sensitivity of the second model is shown in the table below:

Table 2: Results after construction of the second model

Method	Accuracy	Sensitivity	Mean
Logistic Regression	0.7755102	0.5	0.6377551
Naive Bayes	0.7755102	0.5	0.6377551

Approach 3: Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes. Decision trees are commonly used in operations research, specifically in decision analysis. One advantage of decision trees is that they are highly interpretable. The way in which decision trees make classifications is in line with how many people would expect doctors to predict the class of a potentially at-risk patient. The rpart package [6] is used to construct the decision tree.

The decision that has to be made when constructing a decision tree model is what complexity parameter (the factor by which the model's performance needs to improve by to warrant another split) to use. Bootstrap (25 samples of 25% of the data set) is used to select the optimal complexity parameter.

The default approach taken by the `train` function in the caret package is to use a `minsplit` of 20 and `minbucket` of 7. The results of the bootstrap are shown in Figure 10, indicating that 0.005 is the optimal choice.

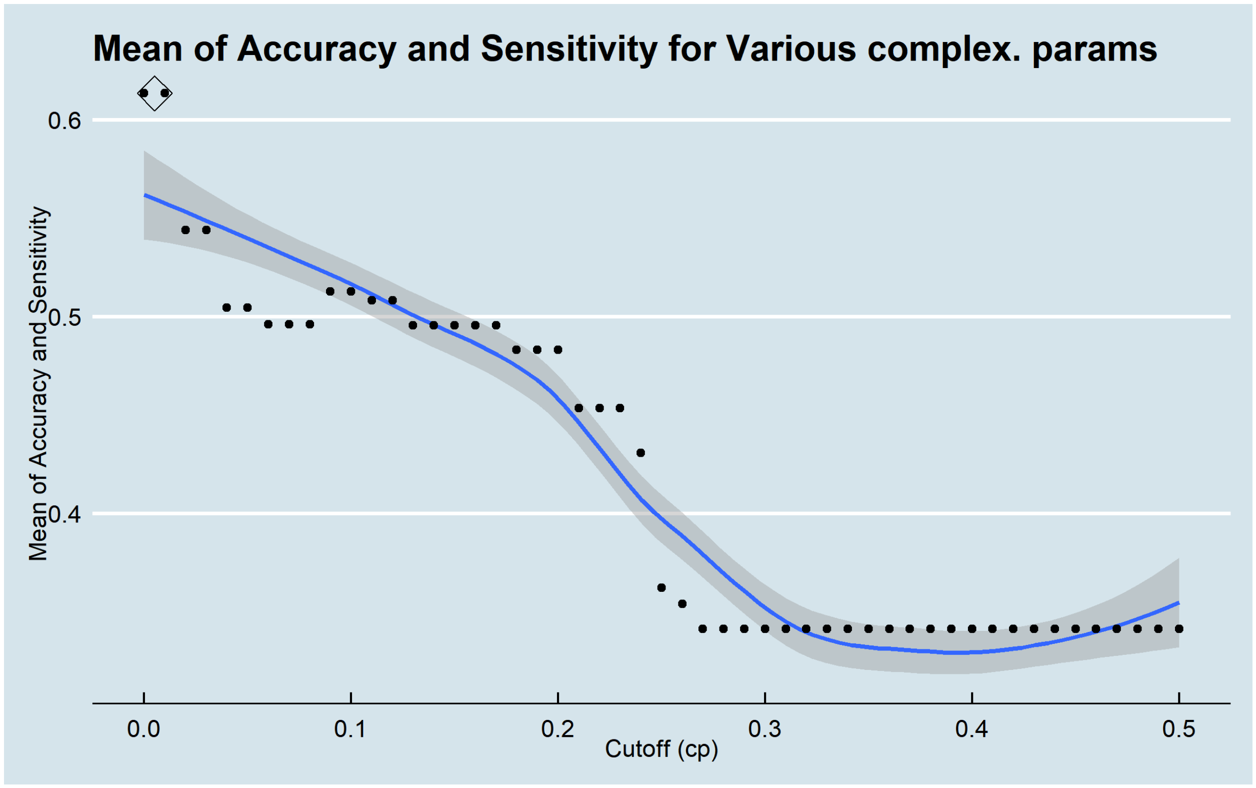

Figure 19: Bootstrap (25 samples of 25% of the data) results from decision tree. Optimal cp is 0.005.

The image below illustrates exactly how the tree makes decisions. The root node makes the first split based on the patient's serum creatinine levels. If it's about 1, they are classed as being at risk. If not, a further split is made based on the patient's ejection_fraction, and so on. The percentage at the bottom of each leaf is the proportion of observations in `heart_train` that lie in that leaf. The decimal above the percentage is the proportion of observations in that leaf that survived.

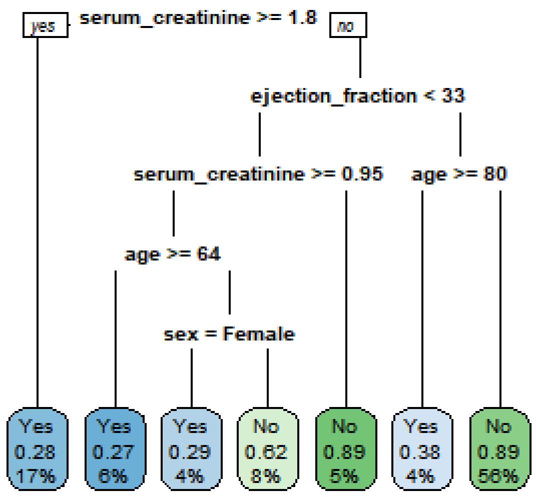

Figure 20: Decision tree.

The confusion matrix for this model is shown below:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  10  8
##        No    6 25
##
##                Accuracy : 0.7143
##                  95% CI : (0.5674, 0.8342)
##     No Information Rate : 0.6735
##     P-Value [Acc > NIR] : 0.3292
##
##                   Kappa : 0.3706
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.6250
##             Specificity : 0.7576
##          Pos Pred Value : 0.5556
##          Neg Pred Value : 0.8065
##              Prevalence : 0.3265
##          Detection Rate : 0.2041
##    Detection Prevalence : 0.3673
##       Balanced Accuracy : 0.6913
##
##        'Positive' Class : Yes
##
```

The accuracy & sensitivity of the model is shown in the table below:

Table 3: Results after construction of the third model

Method	Accuracy	Sensitivity	Mean
Logistic Regression	0.7755102	0.500	0.6377551
Naive Bayes	0.7755102	0.500	0.6377551
Decision Tree	0.7142857	0.625	0.6696429

Approach 4: Random Forest

This model is an extension of the decision tree - a random forest is a collection of decision trees. The way the random forest makes predictions is by some form of majority vote among all of the trees. Trees are constructed in a similar way as the previous section, however at each node a random subset of features is chosen to make the split. This increases the independence between the trees (this parameter is `mtry` in the randomForest package [8]).

Again, bootstrap (25 samples of 25%) is used to choose an optimal `mtry`. The randomForest package takes the default `nodesize` (minimum size of terminal nodes) to be 1 and the default `ntree` (number of decision trees in the forest) to be 500. 10-fold cross validation is used to calculate the optimal `mtry` that maximises the accuracy.The results are shown below in Figure 21:

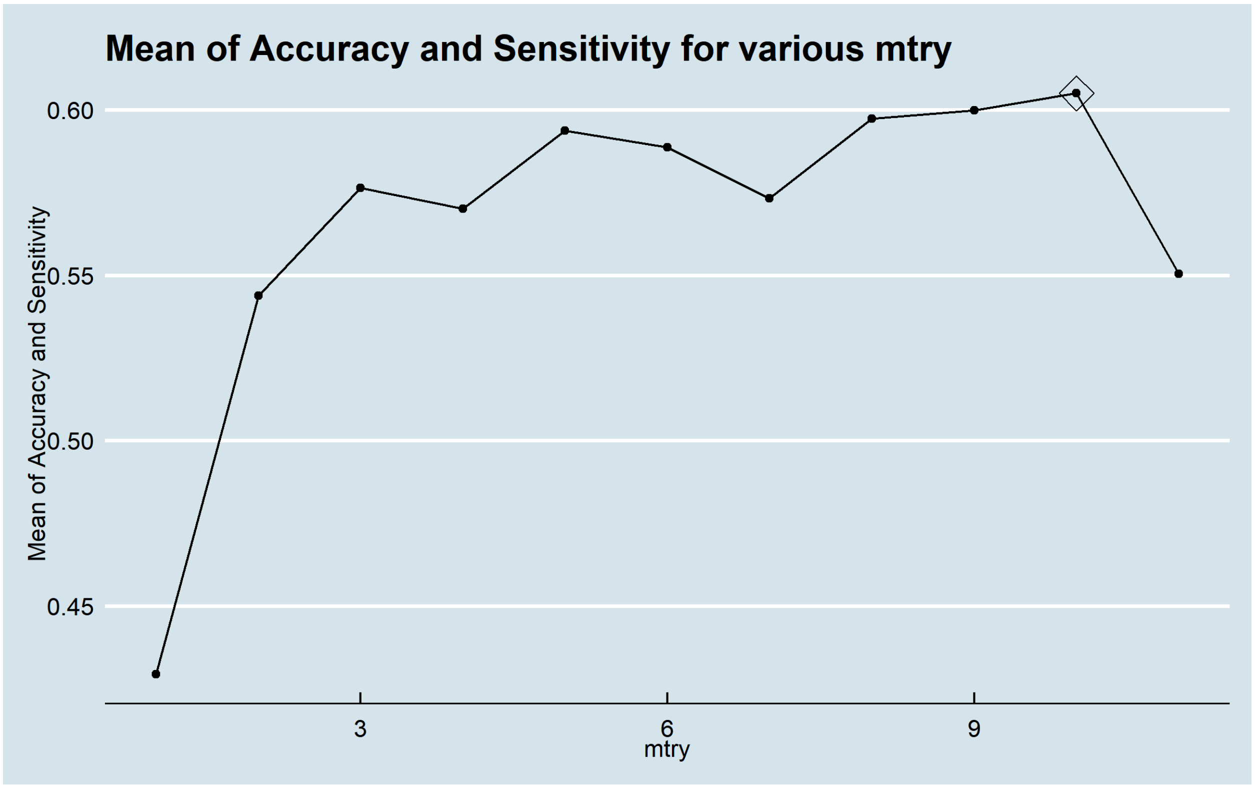

Figure 21: Bootstrap results for various values of mtry. The optimal mtry is 10

The confusion matrix below indicates that the random forest performs very well in comparison to the previous models. it achieves an accuracy of 0.79.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  11  5
##        No    5 28
##
##                Accuracy : 0.7959
##                  95% CI : (0.6566, 0.8976)
##     No Information Rate : 0.6735
##     P-Value [Acc > NIR] : 0.04297
##
##                   Kappa : 0.536
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.6875
##             Specificity : 0.8485
##          Pos Pred Value : 0.6875
##          Neg Pred Value : 0.8485
##              Prevalence : 0.3265
##          Detection Rate : 0.2245
##    Detection Prevalence : 0.3265
##       Balanced Accuracy : 0.7680
##
##        'Positive' Class : Yes
##
```

Table 4 shows the performances of the four models:

Table 4: Results after construction of the fourth model

Method	Accuracy	Sensitivity	Mean
Logistic Regression	0.7755102	0.5000	0.6377551
Naive Bayes	0.7755102	0.5000	0.6377551
Decision Tree	0.7142857	0.6250	0.6696429
Random Forest	0.7959184	0.6875	0.7417092

Approach 5: Support-Vector Machine

A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Given a training set of n points of the form

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n), \tag{4}$$

where the y_i are either 1 or -1, each indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\mathbf{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$ (i.e. separate the groups of patients that died or surived), which is defined so that the distance between the hyperplane and the nearest point $\mathbf{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying:

$$\mathbf{w}^T\mathbf{x} - b = 0, \tag{5}$$

where $\mathbf{w}$ is the (not necessarily normalized) normal vector to the hyperplane.

The goal of the optimization then is to minimize

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)\right)\right] + \lambda\|\mathbf{w}\|^2, \tag{6}$$

where the parameter λ (the cost parameter) determines the trade-off between increasing the margin size and ensuring that the x_i lie on the correct side of the margin. 10-fold cross-validation is used to choose the cost value that maximises the accuracy of the model. The results are shown below:

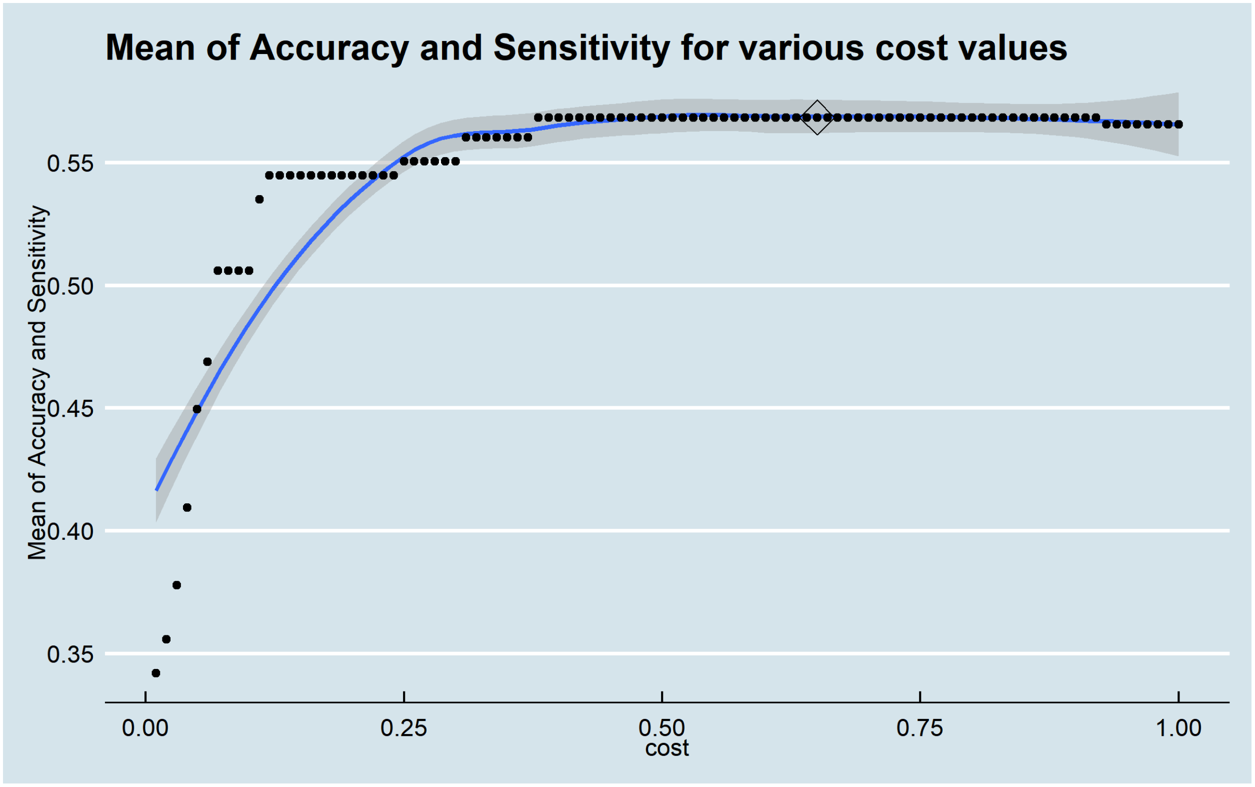

Figure 22: CV results for various cost values. The optimal cost is 0.65

The results of the Support-Vector Machine are shown below:

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction Yes No
##        Yes   8  4
##        No    8 29
##
##                Accuracy : 0.7551
##                  95% CI : (0.6113, 0.8666)
##     No Information Rate : 0.6735
##     P-Value [Acc > NIR] : 0.1424
##
##                   Kappa : 0.4049
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.5000
##             Specificity : 0.8788
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7838
##              Prevalence : 0.3265
##          Detection Rate : 0.1633
##    Detection Prevalence : 0.2449
##       Balanced Accuracy : 0.6894
##
##        'Positive' Class : Yes
##
```

Table 5 shows the performances of the five models:

Table 5: Results after construction of the fifth model

Method	Accuracy	Sensitivity	Mean
Logistic Regression	0.7755102	0.5000	0.6377551
Naive Bayes	0.7755102	0.5000	0.6377551
Decision Tree	0.7142857	0.6250	0.6696429
Random Forest	0.7959184	0.6875	0.7417092
SVM	0.7551020	0.5000	0.6275510

Approach 6: Ensemble

The final model is an ensemble of the three best performing models. Since the Logistic Regression and Support-Vector machine models performed the worst, they are omitted from the ensemble.

The ensemble takes a majority vote for each observation from the three models (Naive Bayes, decision tree and random forest) and uses that as its prediction. By dropping two models ties are avoided. Ensembling machine learning models is a great strategy to improve accuracy on test sets - it reduces the reliability on the performance of only one algorithm.

The confusion matrix below shows that the ensemble does indeed provide the best average accuracy and senstivity:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  11  5
##        No    5 28
##
##                Accuracy : 0.7959
##                  95% CI : (0.6566, 0.8976)
##     No Information Rate : 0.6735
##     P-Value [Acc > NIR] : 0.04297
##
##                   Kappa : 0.536
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.6875
##             Specificity : 0.8485
##          Pos Pred Value : 0.6875
##          Neg Pred Value : 0.8485
##              Prevalence : 0.3265
##          Detection Rate : 0.2245
##    Detection Prevalence : 0.3265
##       Balanced Accuracy : 0.7680
##
##        'Positive' Class : Yes
##
```

Table 6: Results after construction of all the models

Method	Accuracy	Sensitivity	Mean
Logistic Regression	0.7755102	0.5000	0.6377551
Naive Bayes	0.7755102	0.5000	0.6377551
Decision Tree	0.7142857	0.6250	0.6696429
Random Forest	0.7959184	0.6875	0.7417092
SVM	0.7551020	0.5000	0.6275510
Ensemble	0.7959184	0.6875	0.7417092

Final Model (Results)

In typical data science projects, it is usually the case that the ensemble achieves the best results. Thus, it is selected to be the final model.

The entire `train` data set is now used to construct a random forest. The Naive Bayes, Decision tree and Random Forest models are constructed in the same manner they were constructed on the `heart_train` dataset.

The confusion matrix indicates that the ensemble achieves 77% accuracy and 50% Sensitivity:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes No
##        Yes  10  4
##        No   10 37
```

```
##
##                Accuracy : 0.7705
##                  95% CI : (0.645, 0.8685)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 0.06369
##
##                   Kappa : 0.4359
##
##  Mcnemar's Test P-Value : 0.18145
##
##             Sensitivity : 0.5000
##             Specificity : 0.9024
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.7872
##              Prevalence : 0.3279
##          Detection Rate : 0.1639
##    Detection Prevalence : 0.2295
##       Balanced Accuracy : 0.7012
##
##        'Positive' Class : Yes
##
```

Conclusion

Though we've made a model that performs 27% better than guessing, there are ways in which we can achieve a greater level of accuracy and specificity:

Sample Size

The `heart_train` data set only has 299 observations. Furthermore, the final model is only tested on 78 observations. With a much larger dataset, we would be able to calculate more accurate hyperparamters and thus achieve higher accuracy.

More relevant features

We saw in our various models that a lot of the features in our dataset do not appear to be significant risks towards heart failure. If we had more variables to assess (such as height, BMI, location) we may be able to construct a more powerful model.