DSCI 445 Project Paper

Paige Galvan, Neha Deshpande, & Witlie Leslie

2024-11-25

Motivation

The goal of our project is to predict mortality from heart failure using behavioral risk factor data. Heart failure is a disease that affects millions of people yearly. Although modern medicine has improved, it can be hard to determine causes of heart failure due to how many variables can affect it. The Heart Failure Clinical Records Dataset provides a collection of medical indicators such as age, ejection fraction, serum creatinine, and co-existing conditions like diabetes and high blood pressure. By analyzing this data, researchers can uncover patterns that contribute to better understanding the progression of heart failure.

The main motivation for our group to study this dataset is to dive a little bit deeper into which factors affect heart failure. Knowing that heart failure is a leading cause of death around the world, finding meaningful patterns can inform public health strategies, such as targeted lifestyle modifications or health care campaigns. The main objective is to transform this raw data into meaningful conclusions on heart disease.

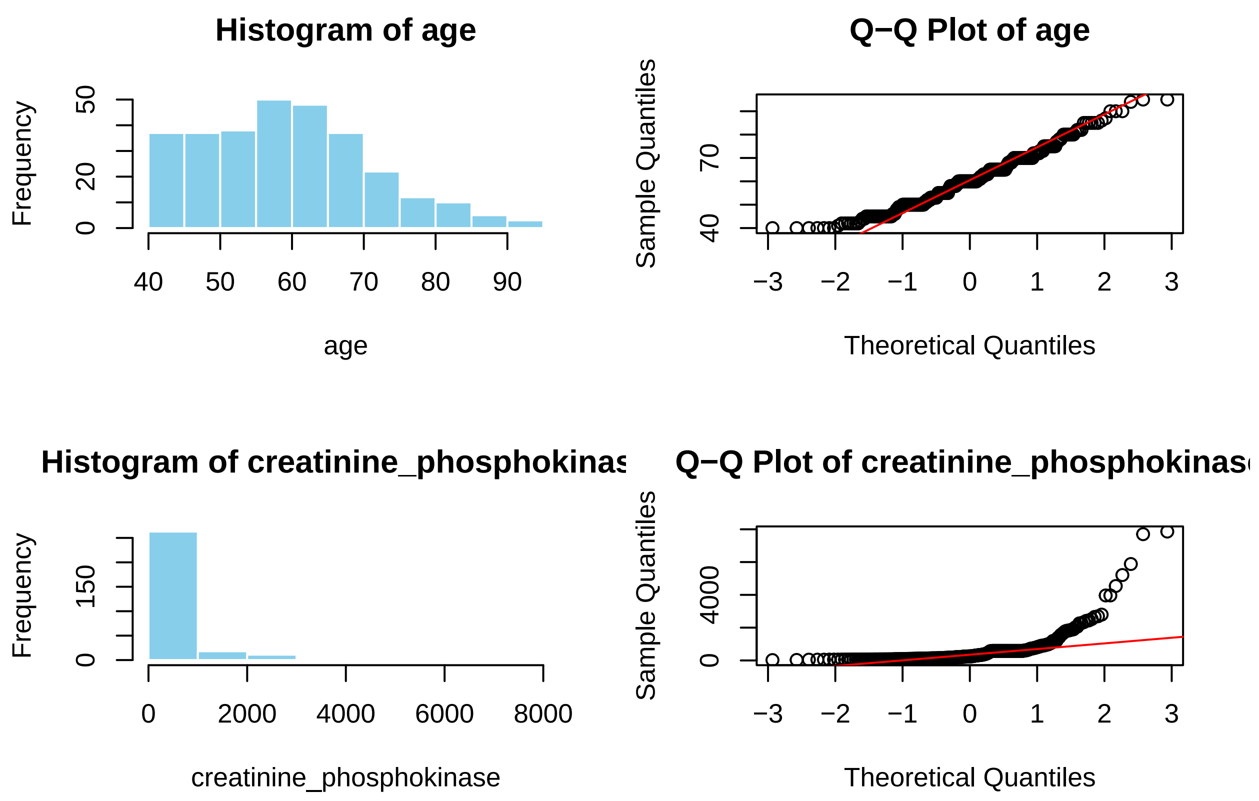
Methodology

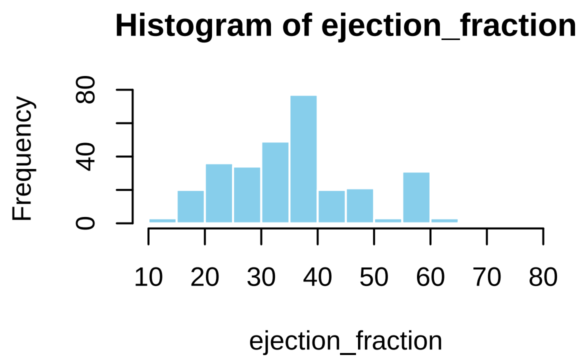
In our project, we aimed to develop a predictive model to determine the likelihood of death resulting from heart failure using clinical patient data. The binary response variable, "DEATH_EVENT," indicates whether a patient has died (value of 1) or survived (value of 0). The dataset includes 12 risk factor variables, comprising 5 binary variables—anemia status, diabetes status, high blood pressure status, sex, and smoking status—and 7 numerical variables, including age, creatine phosphokinase level, ejection fraction, platelet concentration, serum creatinine level, serum sodium level, and the length of the follow-up period.

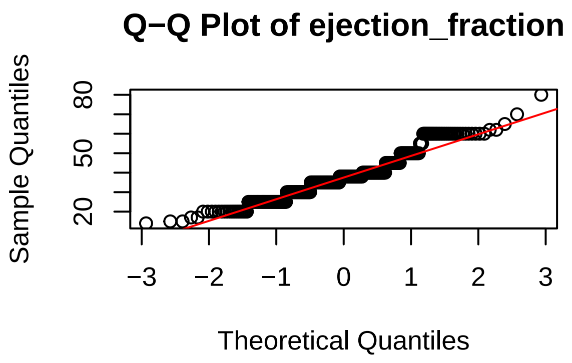
Exploratory Analysis Before applying machine learning models, we began by performing an exploratory analysis of the data. This included assessing the linearity and normality of the predictors, identifying any outliers, and exploring potential correlations among the variables. We visualized distributions using histograms and box plots to understand the spread of each feature, and scatter plots to check the relationships between the predictor variables and the target variable (mortality). This helped us determine whether the data required transformations before applying machine learning techniques.

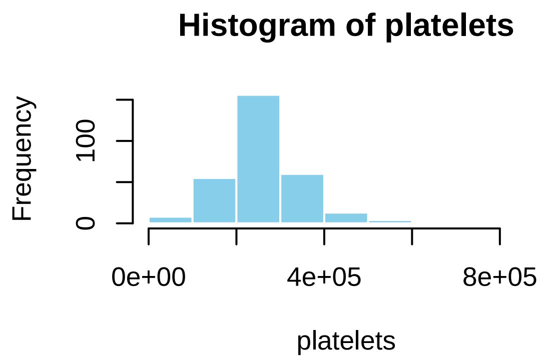
Linearity and Normality The analysis aimed to examine the linearity and normality of continuous variables and their relationships with the binary response variable, "DEATH_EVENT." Shapiro-Wilk tests indicated that none of the continuous variables followed a normal distribution, as all p-values were below 0.05. Histograms revealed that most variables were skewed, except for platelets and serum sodium, which appeared closer to normal. This was further supported by Q-Q plots, where variables like age, ejection fraction, serum sodium, and platelets exhibited patterns suggesting near-normality, while others deviated significantly

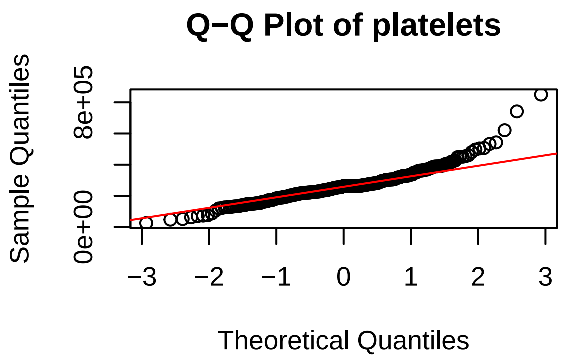
```
data <- read.csv("heart_failure_clinical_records_dataset.csv")

continuous_vars <- c("age", "creatinine_phosphokinase", "ejection_fraction",
                     "platelets", "serum_creatinine", "serum_sodium")

par(mfrow = c(2, 2)) # Set up a grid for multiple plots
for (var in continuous_vars) {

  hist(data[[var]], main = paste("Histogram of", var), xlab = var, col = "skyblue", border = "white")

  qqnorm(data[[var]], main = paste("Q-Q Plot of", var))
  qqline(data[[var]], col = "red")
}
```

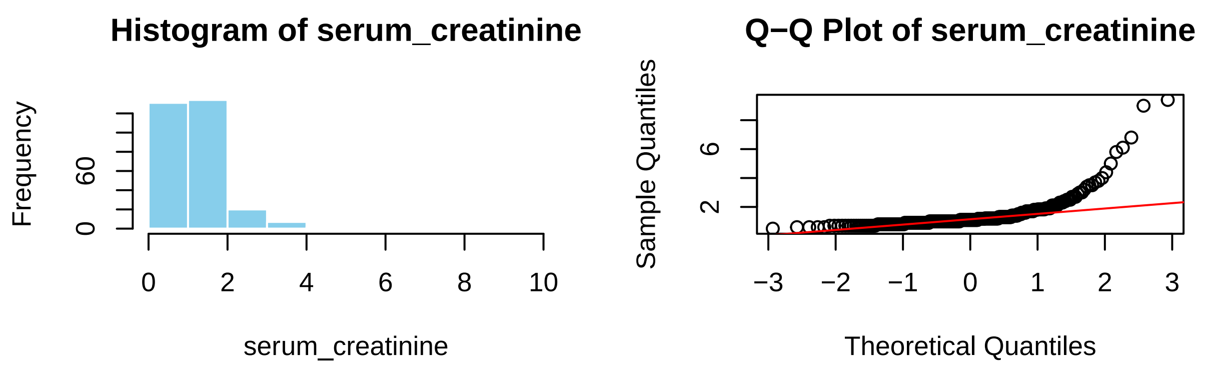


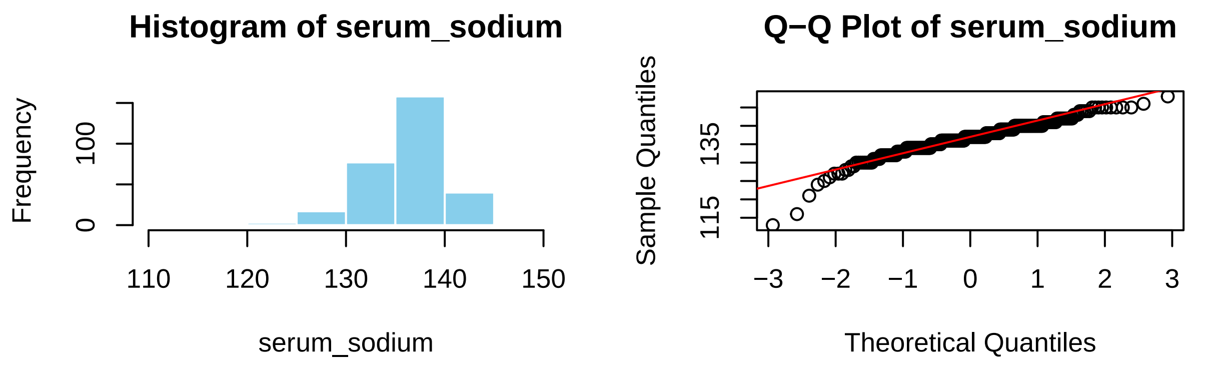

Histogram of ejection_fraction
Frequency
0
40
80
10 20 30 40 50 60 70 80
ejection_fraction


Q−Q Plot of ejection_fraction
Sample Quantiles
20
50
80
−3 −2 −1 0 1 2 3
Theoretical Quantiles


Histogram of platelets
Frequency
0
100
0e+00 4e+05 8e+05
platelets


Q−Q Plot of platelets
Sample Quantiles
0e+00
8e+05
8e+05
−3 −2 −1 0 1 2 3
Theoretical Quantiles





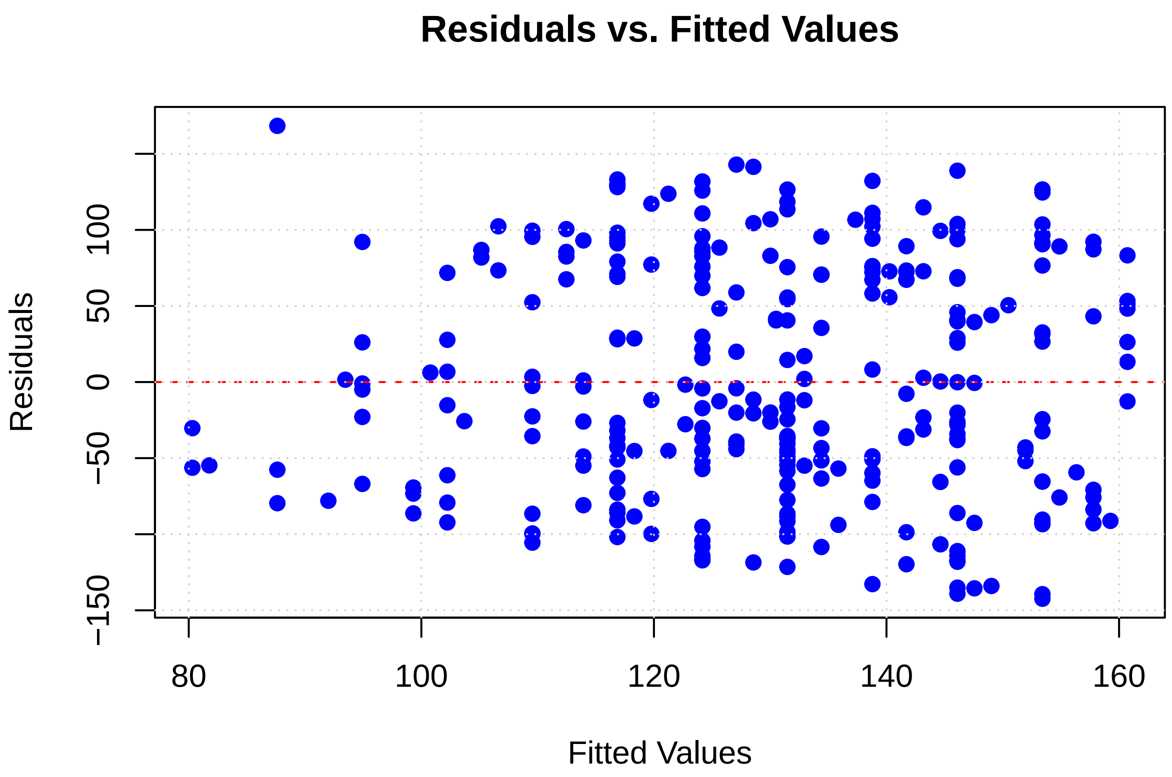
```
for (var in continuous_vars) {
  test <- shapiro.test(data[[var]])
  cat("Shapiro-Wilk test for", var, ":\n")
  print(test)
}
```

```
## Shapiro-Wilk test for age :
##
##  Shapiro-Wilk normality test
##
## data:  data[[var]]
## W = 0.97547, p-value = 5.35e-05
##
## Shapiro-Wilk test for creatinine_phosphokinase :
##
##  Shapiro-Wilk normality test
##
## data:  data[[var]]
## W = 0.51426, p-value < 2.2e-16
##
## Shapiro-Wilk test for ejection_fraction :
##
##  Shapiro-Wilk normality test
##
## data:  data[[var]]
## W = 0.94732, p-value = 7.216e-09
```

```
##
## Shapiro-Wilk test for platelets :
##
##  Shapiro-Wilk normality test
##
## data:  data[[var]]
## W = 0.91151, p-value = 2.883e-12
##
## Shapiro-Wilk test for serum_creatinine :
##
##  Shapiro-Wilk normality test
##
## data:  data[[var]]
## W = 0.55147, p-value < 2.2e-16
##
## Shapiro-Wilk test for serum_sodium :
##
##  Shapiro-Wilk normality test
##
## data:  data[[var]]
## W = 0.93903, p-value = 9.215e-10
```

Logistic regression analysis showed that predictor variables, such as age and serum creatinine, were significantly associated with the likelihood of a DEATH_EVENT. Residual diagnostics confirmed a good model fit, with minimal issues in linearity or outliers. Testing quadratic terms for the predictors showed no substantial improvement, indicating that the relationships between the variables and DEATH_EVENT were primarily linear.

```r
data <- read.csv("heart_failure_clinical_records_dataset.csv")

model <- lm(time ~ age, data = data)

fitted_values <- fitted(model)
residuals <- resid(model)

plot(fitted_values, residuals,
     main = "Residuals vs. Fitted Values",
     xlab = "Fitted Values",
     ylab = "Residuals",
     pch = 19, col = "blue")
abline(h = 0, col = "red", lty = 2)
grid()
```



Logistic Regression with Regularization Logistic regression is a go-to method for binary classification, and we explored three versions to analyze predictive performance. First, we fit a basic logistic regression model without regularization as a baseline. While simple, it doesn't handle collinearity or irrelevant predictors. Next, we applied Ridge regression regularization, which penalizes large coefficients to stabilize the model, though it doesn't eliminate predictors, making it less interpretable than Lasso. Finally, we used Lasso regularization, which not only penalizes coefficients, but also performs feature selection by shrinking some to zero, improving interpretability. Comparing their predictive power helps determine which approach balances accuracy and simplicity best.

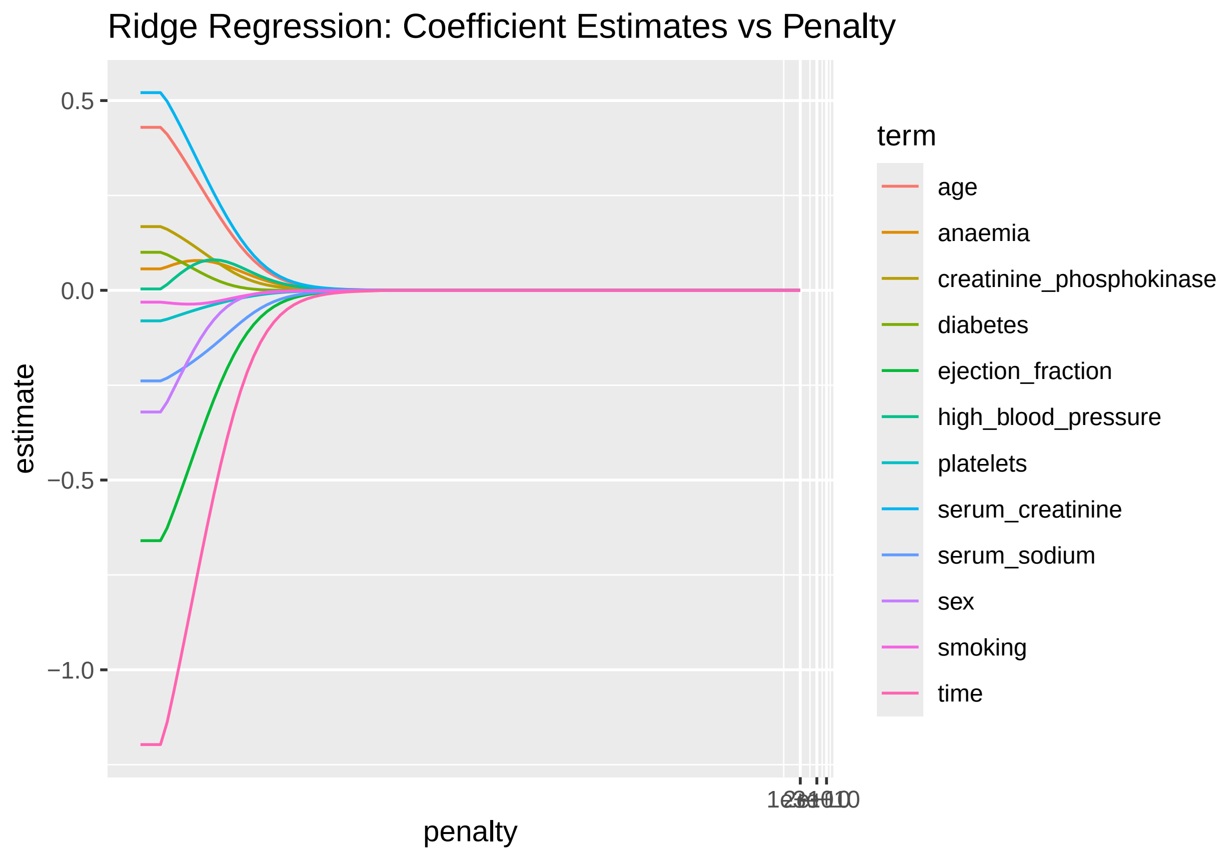
Because the predictor variables are of varying ranges and units, we began by scaling all continuous features to prevent our regularization techniques from over-penalizing variables with larger ranges. Next, we split our data into a training set (containing 80% of the data) and a test set (containing 20%) so that we could assess the performance of our logistic regression models using cross validation.

Once this setup was complete, we created our first logistic regression model with no regularization. The metrics we used to assess the performance of these models are roc_auc, which is the area under the receiver-operating characteristic (ROC) curve that represents the probability that the model will correctly rank a randomly selected positive example higher than a negative one, as well as accuracy, which is the proportion of correct predictions out of all total predictions. Below are the roc_auc and accuracy values of the first logistic regression model with no regularization:

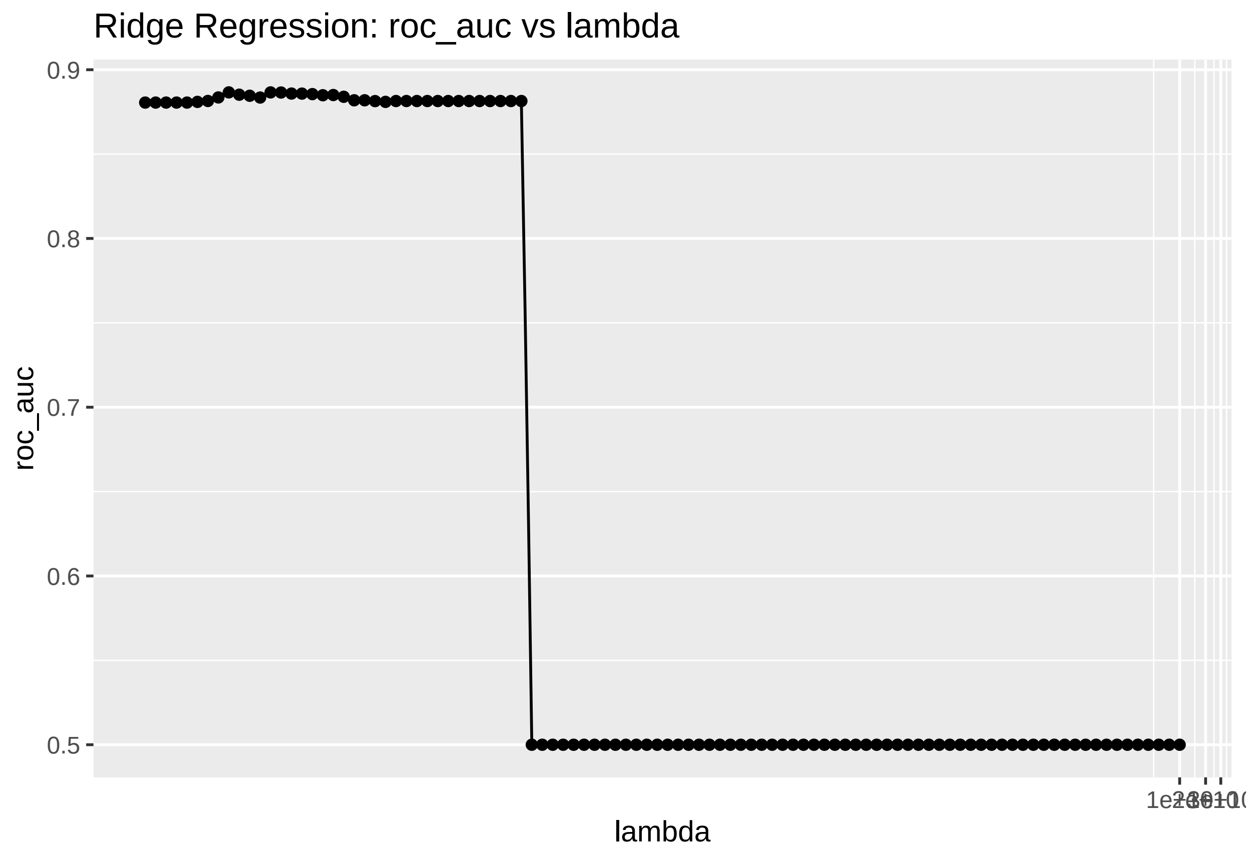
```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 roc_auc  binary         0.145
## 2 accuracy binary         0.833
```

The logistic regression model with no regularization gave an roc_auc value of 0.1325 and an accuracy value of 0.8333. This accuracy value is fairly high, but the roc_auc value is very low, which indicates poor performance.

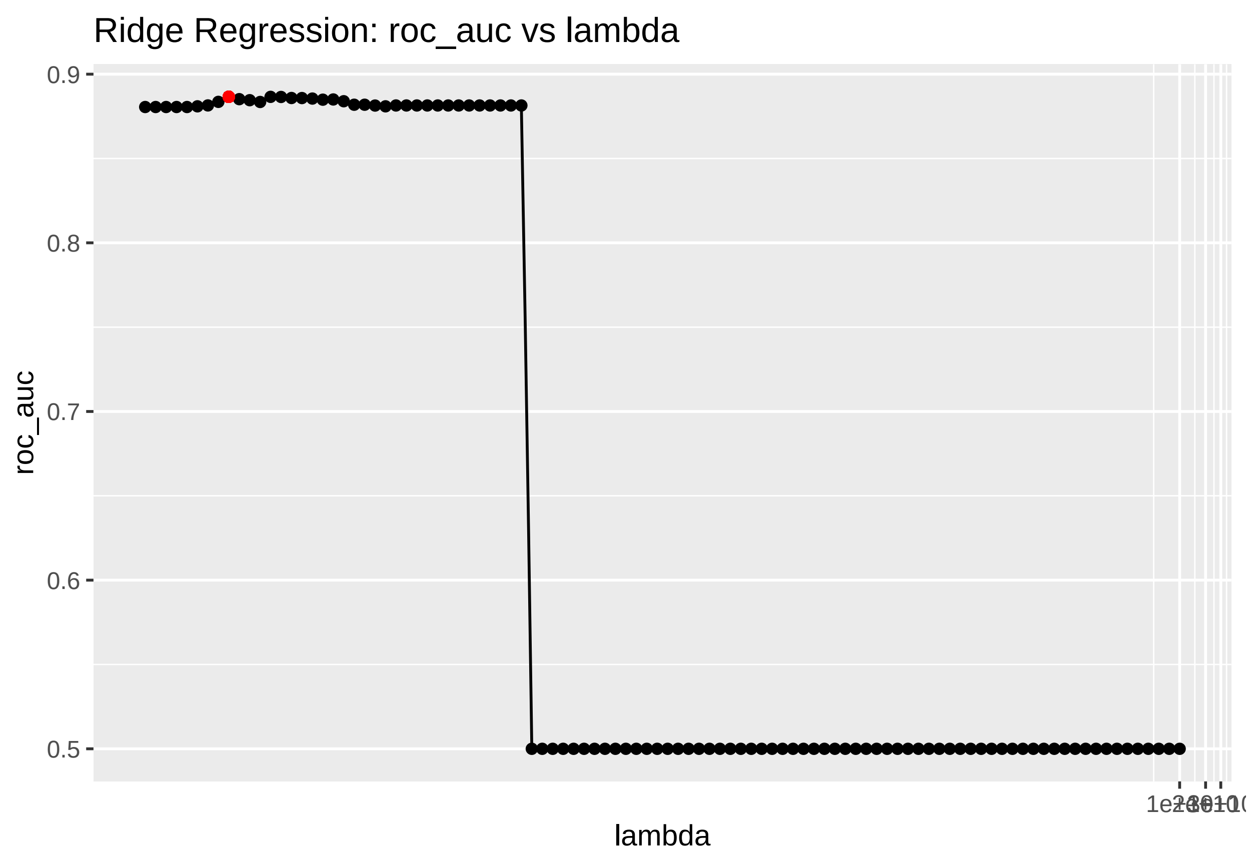
Next, we created a logistic regression model including ridge regression regularization. We determined the optimal lambda penalty value using cross validation.



This graph depicts the estimated value of each coefficient corresponding to each feature for each lambda value tested. As the penalty lambda value increases, the coefficient estimates diminish towards zero without being removed entirely.



This graph depicts the roc_auc value for each lambda tested. A higher roc_auc value indicates better performance, and we can see there is a significant and sudden drop in roc_auc value after approximately the first third of lambda values. After completing cross validation, we can see which lambda value demonstrates the best performance:

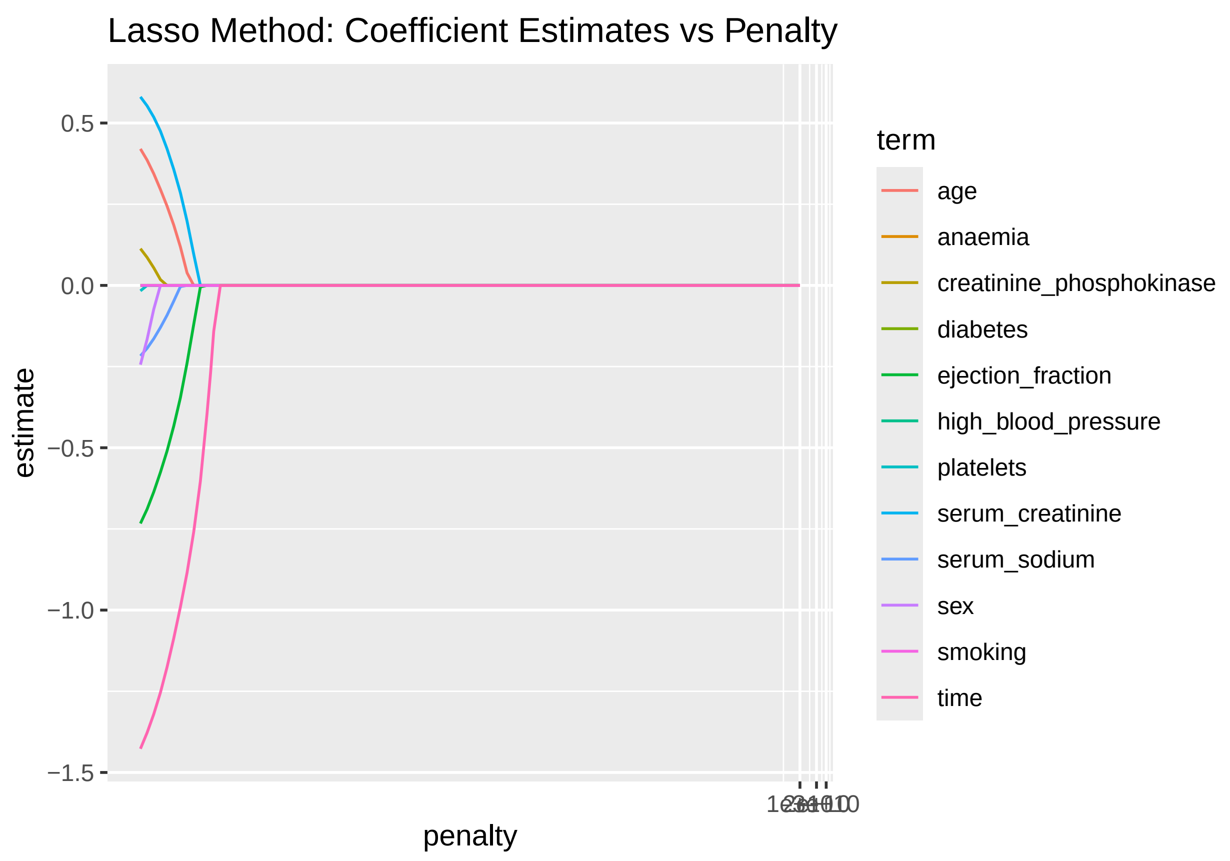


Once we have chosen the optimal penalty, we can fit the ridge regression model with this lambda value and assess its performance with roc_auc and accuracy metrics:

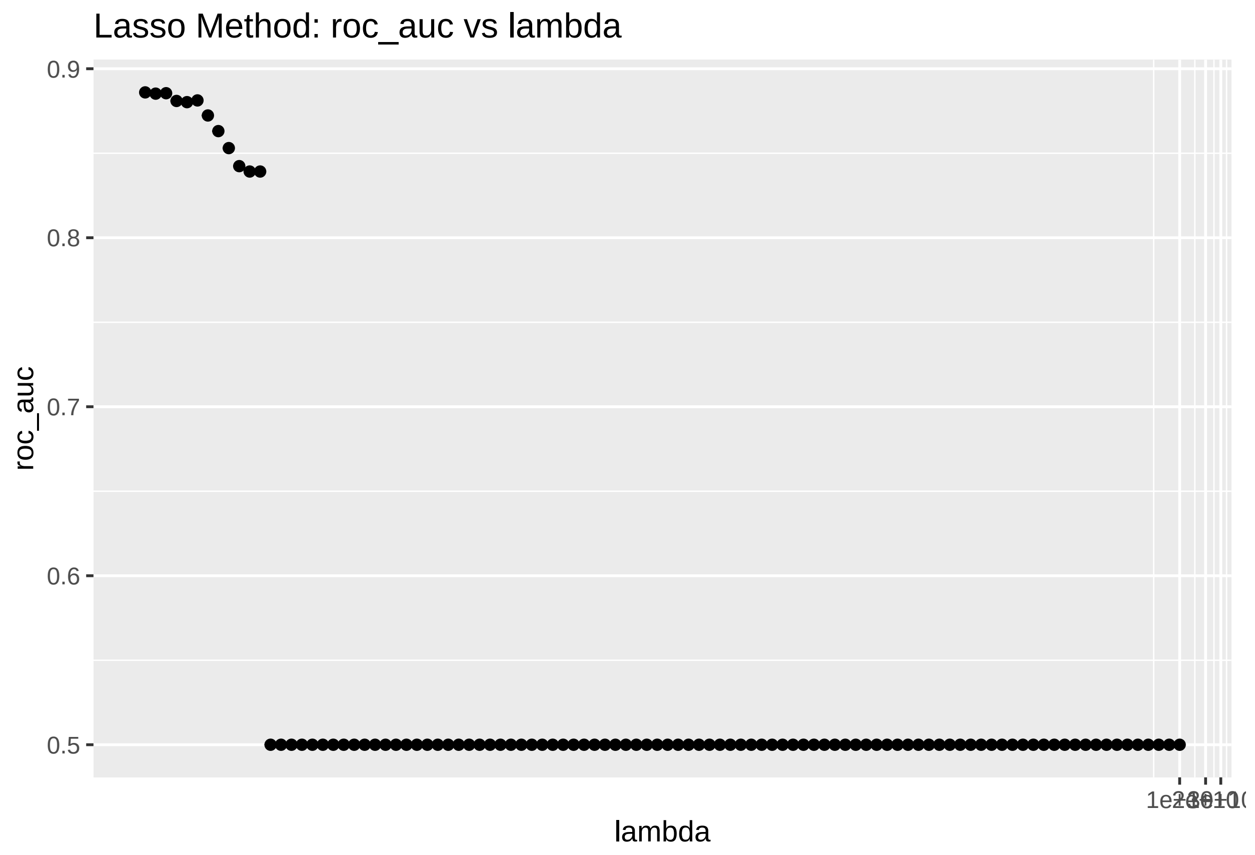
```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 roc_auc  binary         0.144
## 2 accuracy binary         0.8
```

The logistic regression model with ridge regression regularization gave an roc_auc value of 0.1374 and an accuracy value of 0.8333. These values are nearly identical to those of the logistic model with no regularization.

For our final logistic regression model, we will implement lasso, once again using cross validation to determine the optimal lambda penalty value.

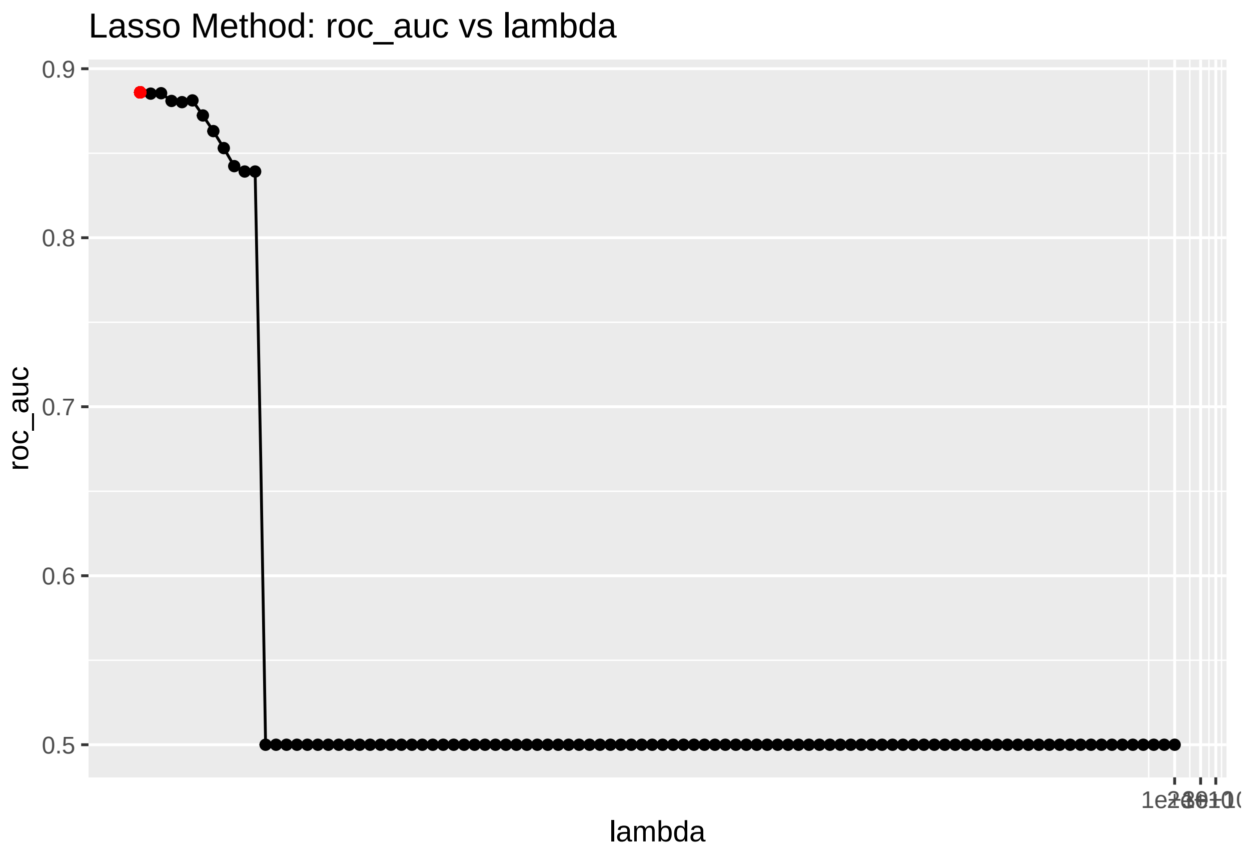


Here we can see the coefficient estimates reduce towards zero as lambda increases. The reduction in coefficient estimates is notably steeper for lasso than with ridge regression. Unlike ridge regression, lasso performs feature selection by driving some coefficients to equal zero, thus eliminating them.



We can see in the graph above a significant and sudden drop in roc_auc value after the first dozen lambda values. After completing cross validation, we can see which lambda value demonstrates the best performance:

```
## # A tibble: 1 x 7
##   lambda .metric .estimator  mean     n std_err .config
##    <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1   0.01 roc_auc binary     0.886    10  0.0211 Preprocessor1_Model001
```



Again, once we have chosen the optimal penalty, we can fit the lasso model with this lambda value and assess its performance with roc_auc and accuracy metrics:

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 roc_auc  binary         0.134
## 2 accuracy binary         0.833
```

The logistic regression model with ridge regression regularization gave an roc_auc value of 0.1027 and an accuracy value of 0.850. These values are again nearly identical to those of the logistic model with no regularization and the logisitic model with ridge regression. The lasso model provided the highest accuracy of the three models, but also an even lower roc_auc value.

Linearity and Normality

In this analysis, we begin by testing the normality of several continuous variables in the dataset, including age, creatinine phosphokinase, ejection fraction, platelets, serum creatinine, and serum sodium. First, we used histograms and Q-Q plots to visually inspect the distribution of these variables. The histograms for most variables indicated skewness, either to the right or left, suggesting that these variables do not follow a normal distribution. Platelets and serum sodium appeared to be the most normally distributed, but even they showed some deviation from normality. The Q-Q plots confirmed these observations, showing that age, ejection fraction, serum sodium, and platelets were closer to a normal distribution, while other variables exhibited greater deviations.

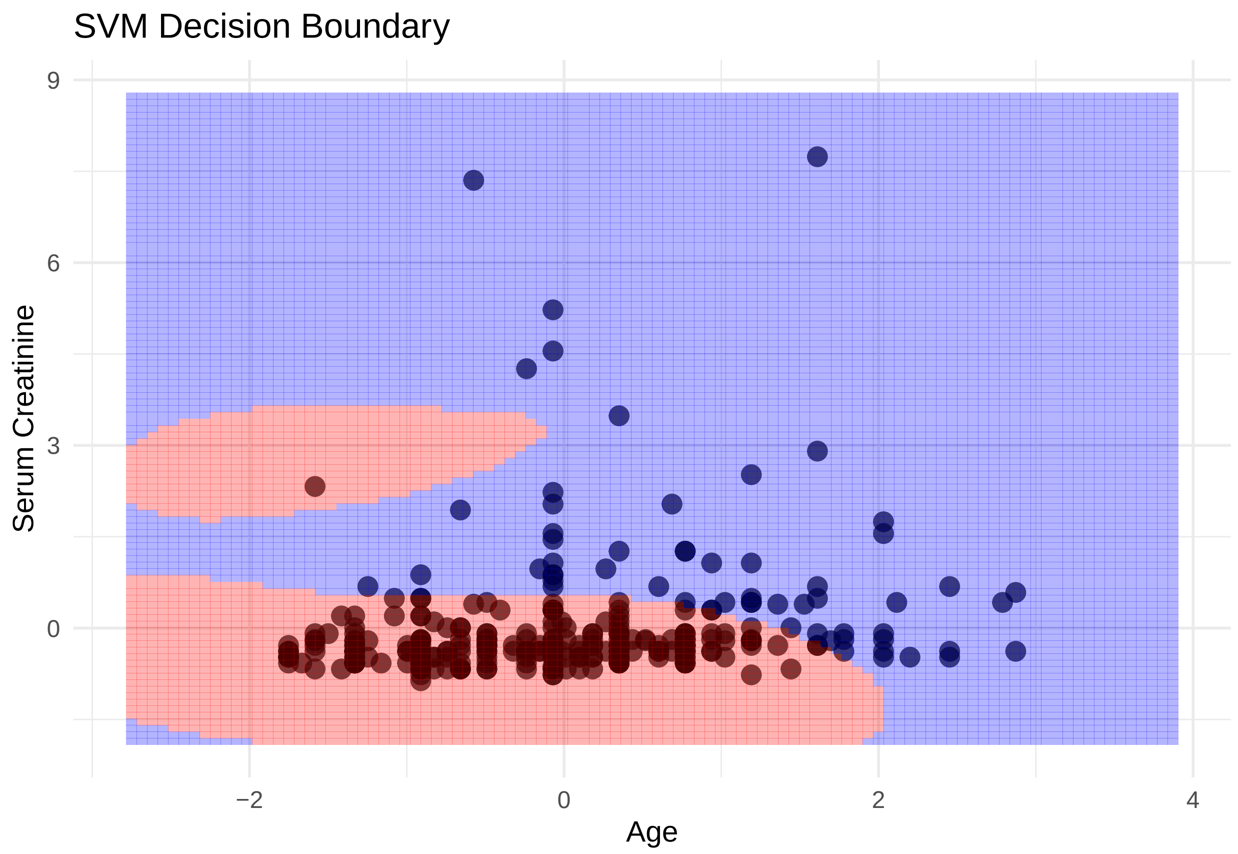
To check the normality of the continuous variables, we performed the Shapiro-Wilk test, and all p-values were below 0.05, indicating that none of the variables followed a normal distribution.

We then examined the relationship between these variables and the binary outcome, DEATH_EVENT, using scatter plots with linear regression lines. These plots showed that variables like age and serum creatinine were somewhat associated with the likelihood of death, showing linear trends in most cases. Logistic regression models confirmed that many of the variables, such as age and serum creatinine, were significantly related to the risk of death.

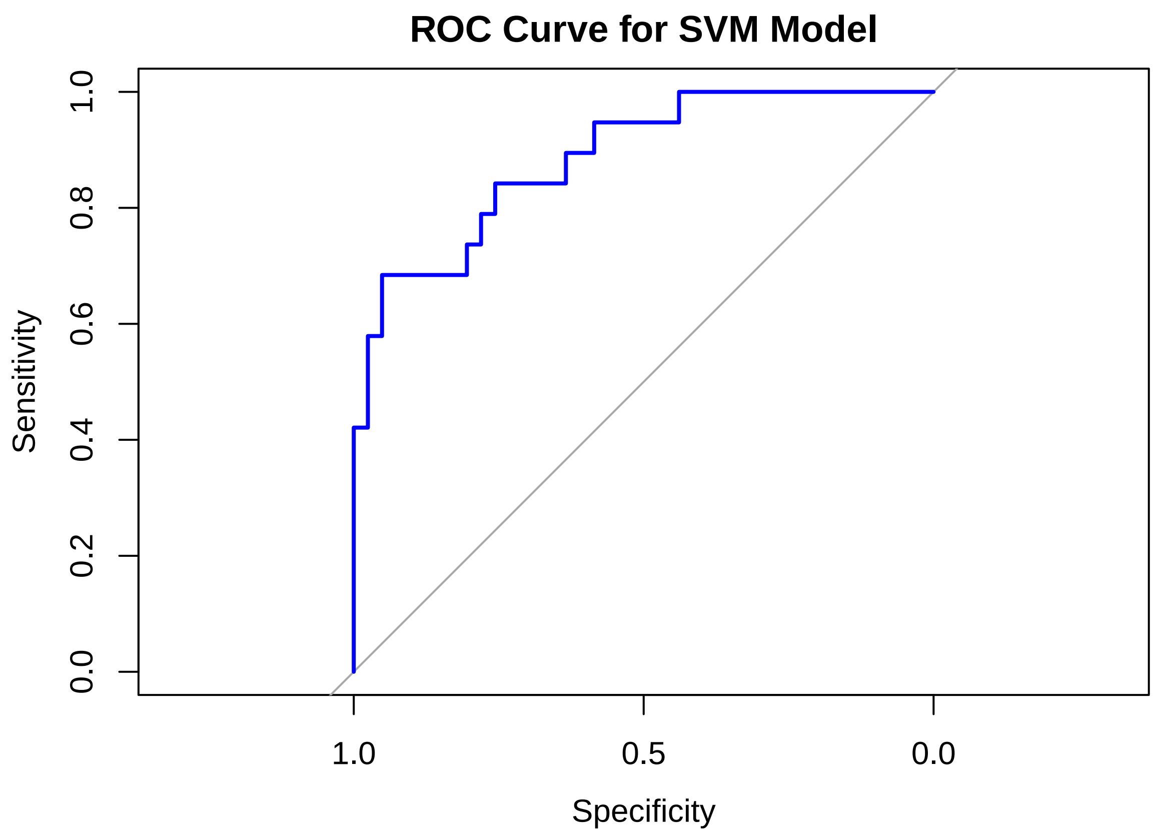
Support Vector Machine Support Vector Machines (SVMs) are primarily binary classifiers. When dealing with more than two classes, SVMs can handle multi-class classification by applying techniques like "one-vs-one" or "one-vs-all," where multiple binary classifications are combined. One of the key advantages of SVMs is their ability to perform non-linear classification, which increases their flexibility and allows them to handle complex decision boundaries. This handels linear and non-linear decision boundaries. Using a linear kernel is good for approximately linear relationships, which our data is. It does not assume any specific distribution of predictors since none of our predictors are normal.

We began with our SVM model looking to the Support Vectors and the parameters of interest. The support vector are important to understanding the model's decisions. They are the informative points and make it the most critical for classification. They demonstrate the most ambiguous points of data.

From this code we conclude that there are only 13 predictive variables we should consider. We also considered the accuracy of how the SVM model this allows us to understand its predictive capabilities. We also conclude that this model will have an accuracy of 0.817 this indicates the model has a 81.7% of making correct predictions.



- The SVM Boundry analyzes the age vs serum creatinine using the death effect for males vs females. We can see that overall serium creatinine has little effect on the death effect especially in males. Meanwhile age and serum creatinine has a large response on females.

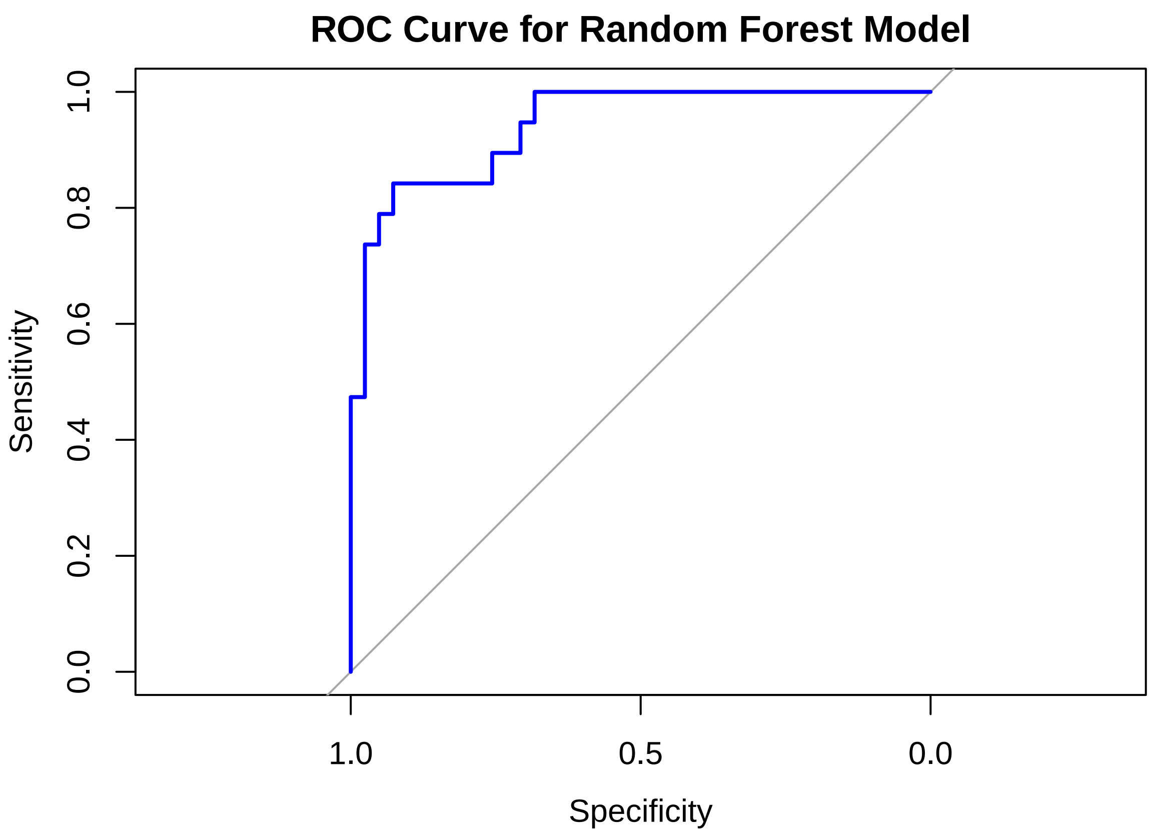


- The SVM model, tuned with `C = 0.5` and `sigma = 0.0562`, had an ROC of 0.86, a sensitivity of 86%, and a specificity of 66%.

Random Forest Random Forest is a powerful machine learning algorithm used for both classification and regression. It works by building multiple decision trees and aggregating their predictions to improve accuracy and reduce overfitting. Key advantages include its ability to handle complex, non-linear relationships, manage missing data, and automatically capture feature interactions. Random Forest is also robust to overfitting, particularly compared to individual decision trees, and provides a built-in estimate of model performance through out-of-bag error. Additionally, it offers valuable insights into feature importance, helping to identify which variables most influence the outcome. Overall, Random Forest is particularly effective for high-dimensional datasets, imbalanced classes, and when model interpretability is secondary to prediction accuracy.

```
##          Actual
## Predicted  0  1
##         0 38  4
##         1  3 15
```

```
## Accuracy:  0.883
```



- The Random Forest model performed best with `mtry = 2`, giving an ROC of 0.91. It was effective at identifying true positives (91% sensitivity) but had a moderate specificity of 69%.
- Overall, the Random Forest model performed better, especially in terms of overall accuracy and detecting positive cases.

«««< HEAD

======= # Results

None of the three logistic regression models performed well. The model without regularization performed very similarly to those which implemented ridge regression and lasso, indicating that regularization is unnecessary for this data. Roc_auc values of 0.10 - 0.13 indicate that the model is performing worse than chance (0.5). A low roc_auc value indicates poor discrimination ability. Having a high accuracy value with a low roc_auc value might suggest that accuracy is being misleadingly driven up by the model assigning predictions to the majority class (in our case, death event = 0). »»»> 06e66d4c39f850846dcef0b1529f6760960600e5

References

Davide Chicco, Giuseppe Jurman: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone.

BMC Medical Informatics and Decision Making 20, 16 (2020). https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5 Creatine phosphokinase level: measurement of creatine phosphokinase in blood. CPK is an enzyme that indicates tissue stress at high levels https://www.mountsinai.org/health-library/tests/creatine-phosphokinase-test

ejection fraction: how much oxygen-rich blood is pumped out to the body's organs with each contraction https://my.clevelandclinic.org/health/articles/16950-ejection-fraction

platelet concentration: the number of platelets in a unit of blood volume https://www.regenlab.com/2022/04/07/platelet-concentration-concentration-factor/#:~:text=Platelet%20concentration%20is%20the%20number,expressed%20in%

serum creatine: a measruemnt of the level of creatine in blood which is linked to kidney function https://www.kidneyfund.org/all-about-kidneys/tests/serum-creatinine-test

serum sodium: a measurement of sodium levels in blood https://www.healthline.com/health/sodium-blood#:~:text=A%20sodium%20blood%20test%20is,for%20nerve%20and%20muscle%20function.