Final Report - At the Heart of the Matter

Annika Cleven, Yajie He, Tyler Humpherys, Dinelka Nanayakkra, Matthew Sutcliffe

```
# load R package
load_all("heaRt")
```

```
## i Loading heaRt
```

1 Introduction

1 (i) Background

Heart disease is a leading cause of death in the United States, accounting for over 700,000 deaths each year and posing a significant healthcare burden [1]. Accurate and early detection of heart disease is essential for improving patient outcomes, with implications for treatment plans, medication strategies, and lifestyle modifications. Traditional diagnostic methods, such as stress testing and electrocardiography (ECG), provide detailed, valuable insights of possible signals for the presence of heart disease. Thus, integrating this data together can enhance heart disease identification. The objective of our project is to develop a data-driven approach to predict the presence or severity of heart disease.

1 (ii) Data

To accomplish this objective, we will analyze a publicly available dataset collected by the Cleveland Clinic between 1981 and 1984. This dataset includes 303 patients with or without heart disease, and includes a variety of their demographic features (e.g. age and sex) and clinical diagnostic features (e.g. resting blood pressure, serum cholesterol, chest pain, and response to exercise) on which to train a predictive model. Specifically, the heart disease outcome of interest is a categorical value with 0 being no heart disease and 1 to 4 being increasing heart disease severity. In our analysis we will use both the full outcome spectrum (0-4) and a simplified, binary outcome of 0 (no heart disease) and 1 (heart disease, any severity). The link to the dataset repository is found here: https://doi.org/10.24432/C52P4X. In addition to the age and sex demographic variables, the table below provides more information on the clinical variables available in the dataset.

Table 1: Clinical variables available in the dataset.

Variable	Name	Type	Explanation
cp	Chest Pain Type	Cat	Typical angina, atypical angina, non-anginal pain, or asymptomatic
trestbps	Resting BP	Int	In mm Hg on admission to the hospital
chol	Serum Cholesterol	Int	Elevation associated with heart disease
fbs	Fasting Blood Sugar > 120 mg/dl	Cat	1 = true; 0 = false. Elevation is associated with heart disease

restecg	Resting ECG Results	Cat	Normal, having ST-T wave abnormality, or showing probable/definite left ventricular hypertrophy by Estes' criteria
thalach	Max Heart Rate Achieved	Int	During exercise
exang	Exercise Induced Angina	Cat	1 = yes; 0 = no
oldpeak	ST Depression Induced by Exercise	Int	Commonly presents in ischemia. The deeper and earlier the depression during exercise, the higher the likelihood of obstructive coronary disease
slope	Peak Exercise ST Segment Slope	Cat	Upsloping, flat, or downsloping. Flat or downsloping ST segments more often associated with coronary artery disease
ca	Number of Major Vessels Colored by Fluoroscopy	Int	Appearing to contain calcium deposits; marker of atherosclerosis and higher CAD disease
thal	Thallium Stress Test Result	Cat	3 = normal; 6 = fixed defect (scar tissue from prior heart attacks); 7 = reversible defect (impaired blood flow during stress, normal at rest). Radioactive tracer shows blood flow in heart muscle

A quick check for suspicious or possible mismeaured data values revealed a serum cholesterol value of 409 mg/dl, which is almost double what is considered high, as well as revealed a maximum heart rate during exercise of 71 beats per minute. It was also discovered (see below) that 6 data values were missing: two missing exercise thallium scintigraphic defect values and four missing number of major vessels appearing to contain calcium deposits values. Since our goal is prediction and the number of missing data was really small, then these patients were dropped from our analysis.

Table 2: Number of missing data values by variable.

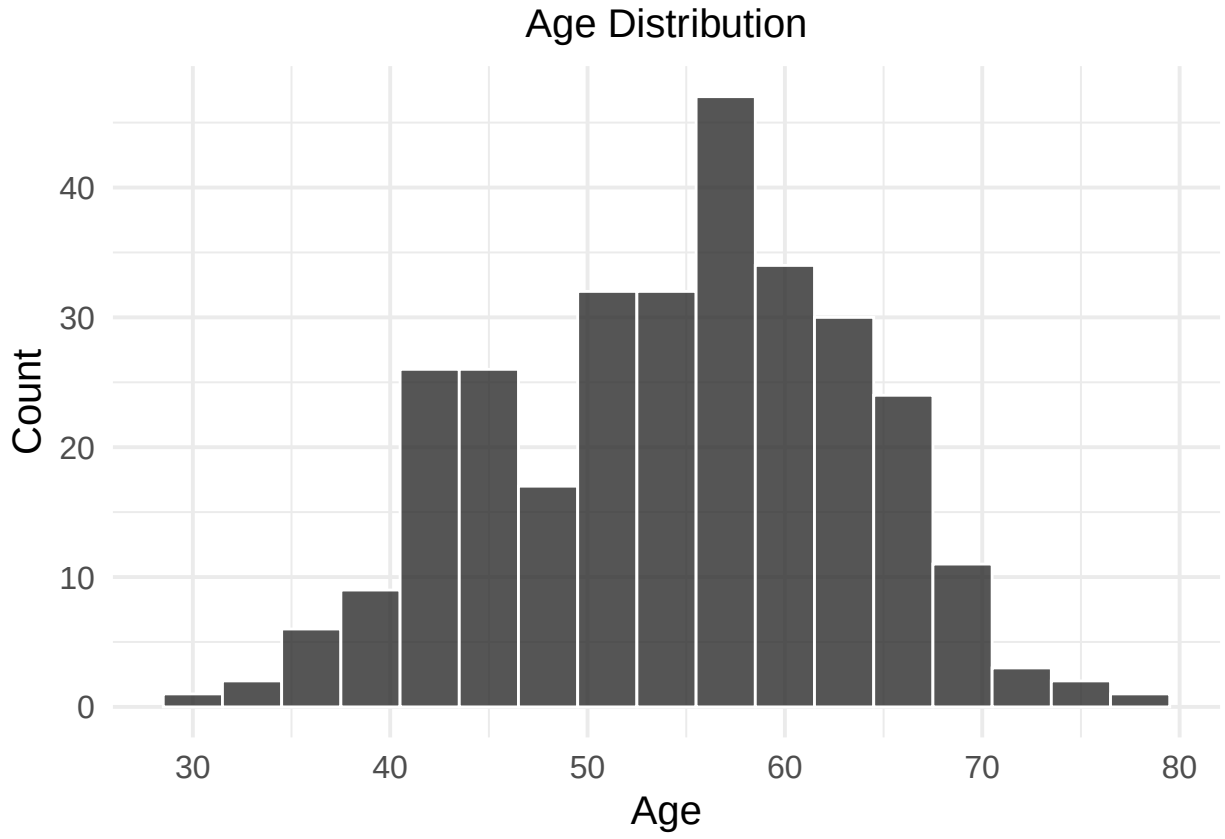
	missing_count
X	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	4
thal	2
num	0

missing_count

After this we investigated several properties of our data. First we looked at the distributions of sex, heart disease outcome (num), and age.

Table 3: Distributions of sex and heart disease outcome in the dataset.

Column	Value	Count	Percentage
sex	0	97	32.01
sex	1	206	67.99
num	0	164	54.13
num	1	55	18.15
num	2	36	11.88
num	3	35	11.55
num	4	13	4.29



From these we see that the distribution of ages is approximately normal centered around the upper 50s, about 68% of the dataset is male, and about 54% of the patients in our data do not have heart disease. Although there is a small number of patients in each disease severity category, it is important to note that when we dicotomoze the outcome we have a more balanced dataset.

The other property we investigated was the correlation between variables.

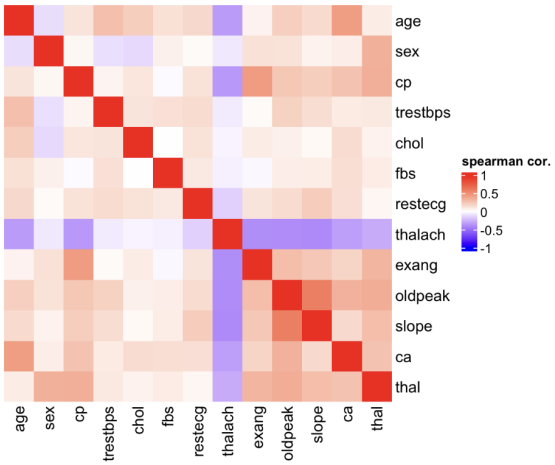


Figure 1: Figure: Correlation between variables.

A few variables had mild correlation with each other. Most of these correlated variables are from ECG graph results (e.g. slope, wave depression). A check on multicollinearity with VIF was performed and nothing of concern was found.

2 Methods

2 (i) Data Preprocessing

For categorical variables, we applied one-hot encoding to convert each category into a separate binary indicator variable. After this transformation, the full model included a total of 20 predictor variables. To address potential overfitting and improve model parsimony, we employed stepwise selection based on the Akaike Information Criterion (AIC). This approach iteratively added or removed predictors to identify a subset of variables that provided the best balance between model fit and complexity, ultimately selecting a more streamlined and interpretable model for subsequent analysis.

2 (ii) Logistic Regression Model

Logistic regression is a widely used statistical model for predicting a binary outcome based on a set of predictor variables. The model assumes that the log-odds of the probability of the outcome is a linear function of the predictors. Specifically, if Y is a binary outcome taking values in $\{0, 1\}$ and $\mathbf{x}$ is a vector of covariates, then the logistic regression model is given by:

$$\log\left(\frac{\Pr(Y=1|\mathbf{x})}{\Pr(Y=0|\mathbf{x})}\right) = \mathbf{x}^\top \boldsymbol{\beta}$$

where $\boldsymbol{\beta}$ is the vector of regression coefficients to be estimated. The corresponding probability model is:

$$\Pr(Y=1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{x}^\top \boldsymbol{\beta})}$$

To estimate the coefficients $\boldsymbol{\beta}$, maximum likelihood estimation is employed. The likelihood function for a sample of size n is:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \left(\frac{1}{1+\exp(-\mathbf{x}_i^\top \boldsymbol{\beta})}\right)^{y_i} \left(\frac{\exp(-\mathbf{x}_i^\top \boldsymbol{\beta})}{1+\exp(-\mathbf{x}_i^\top \boldsymbol{\beta})}\right)^{1-y_i}$$

and the log-likelihood function is:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \log\left(1 + \exp(\mathbf{x}_i^\top \boldsymbol{\beta})\right)\right)$$

Because there is no closed-form solution for $\boldsymbol{\beta}$ that maximizes the log-likelihood, iterative numerical methods must be used. One of the most common algorithms for this purpose is the Newton-Raphson method.

The Newton-Raphson method is a second-order optimization technique that updates the parameter estimates iteratively. At each iteration k, given a current estimate $\boldsymbol{\beta}^{(k)}$, the next estimate $\boldsymbol{\beta}^{(k+1)}$ is obtained by:

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \left(\nabla^2 \ell(\boldsymbol{\beta}^{(k)})\right)^{-1} \nabla \ell(\boldsymbol{\beta}^{(k)})$$

where $\nabla \ell(\boldsymbol{\beta}^{(k)})$ is the gradient (score function) and $\nabla^2 \ell(\boldsymbol{\beta}^{(k)})$ is the Hessian matrix (second derivative of the log-likelihood).

In logistic regression, the gradient and Hessian have specific forms. The gradient is:

$$\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \pi_i)\mathbf{x}_i$$

where $\pi_i = \Pr(Y = 1|\mathbf{x}_i)$, and the Hessian is:

$$\nabla^2 \ell(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \pi_i(1 - \pi_i)\mathbf{x}_i \mathbf{x}_i^\top$$

Thus, each Newton-Raphson update involves solving a linear system based on the weighted sum of the covariates. The iterations continue until convergence, usually when the change in $\boldsymbol{\beta}$ is smaller than a pre-specified tolerance.

In our model, we set the initial $\boldsymbol{\beta}$ as 0, tolerance as 10^{-3} and iterative times as 10^3.

2 (iii) Proportional Odds Cumulative-Logit Model (POLR)

Although a binary classification model such as logistic regression can be easy to interpret, making use of all available categories in our response variable may help us with better prediction, or provide better interpretability/insight clinically. Given that our response variable is ordinal in nature, a proportional odds cumulative-logit model seemed ideal to preserve interpretability and simplicity.

This approach models the logit of the cumulative probabilities as a linear combination of an intercept (unique to each category) and the covariates of the model. This can be written out as follows:

$$logit(P(Y \leq j) = \theta_j + X\beta + \epsilon$$

where, $\epsilon \sim N(0, \sigma^2)$, β's are the fixed coefficients, and the θ's refer to an intercept pertaining to each category.

This model measures how likely the response is to be in category j or below versus in a category higher than j. Given that $P(Y \leq k) = 1$ when there are k categories, we do not need a coefficient θ_k. Thus, we will have $k - 1$ intercepts and p beta coefficients to estimate in our model, where p is the number of covariates in our model including the intercept.

Notice in our model that the intercepts can differ but that the slope stays the same regardless of the change in the response category. This comes from our proportional-odds assumption. To estimate our parameters from this model, we use the BFGS algorithm. This is laid out in the next section.

Estimation: using BFGS to estimate POLR parameters

The BFGS algorithm requires the use of the objective (log-likelihood) function and the first derivative of it to go about the optimization process. As the user, we do not need to input second derivatives but the method will approximate the Hessian matrix. Our initial plan was to use Newton Raphson (NR) instead, but the complexity of the 2nd derivative (Hessian) prompted us to use BFGS instead. The updating equation for BFGS can be written out as follows (assuming α refers to the parameter space):

$$\alpha^{(t+1)} = \alpha^{(t)} - (M^{(t)})^{-1} f^{'}(\alpha)$$

where, $M^{(t)}$ is the matrix that approximates the Hessian. Quasi-newton methods (such as that laid our below) are used to approximate the Hessian matrix. The update to the approximated Hessian matrix can be written out as follows:

$$M^{(t+1)} = M^{(t)} - \frac{M^{(t)} z^{(t)} (M^{(t)} z^{(t)})^T}{(z^{(t)})^T M^{(t)} Z^{(t)}} + \frac{y^{(t)} (y^{(t)})^T}{(z^{(t)})^T y^{(t)}} + \delta^{(t)} (z^{(t)})^T M^{(t)} z^{(t)} d^{(t)} (d^{(t)})^T$$

where,

$$d^{(t)} = \frac{y^{(t)}}{(z^{(t)})^T y^{(t)}} - \frac{M^{(t)} z^{(t)}}{(z^{(t)})^T M^{(t)} z^{(t)}}$$

where,

$$z^{(t)} = \alpha^{(t+1)} - \alpha^{(t)}$$

and,

$$y^{(t)} = f^{'}(\alpha^{(t+1)}) - f^{'}(\alpha^{(t)})$$

The above class of algorithms is indexed by $\delta^{(t)}$ and when $\delta^{(t)} = 0$, this update is the BFGS update.

Whilst we coded up the log-likelihood and the first derivative by hand, we did not derive the BFGS algorithm from scratch. The optimx function from the optimx package on R was used for the optimization step. The steps used to derive the estimates using BFGS is given below:

Derive log-likelihood function

$$ll = \sum_{i=1}^{n} \sum_{j=1}^{J} I(y_i = j) log[\frac{1}{1 + exp(-(\theta_j - x_i^T \beta))} - \frac{1}{1 + exp(-(\theta_{j-1} - x_i^T \beta))}]$$

where, $I(y_i = j)$ is the indicator function indicating if a response is category j or not. Each individual i contributes only to one category.

Here, we can see that the 2 terms within log are Cumulative Distribution Functions (CDFs) related to the logistic function. For ease in the next steps we can denote these two terms as F_j and F_{j-1} respectively. Let's also denote $\pi_{ij} = F_j - F_{j-1}$

Derive the score function (first-derivative) The score function can be written as,

$$f^{'}(\theta) = \begin{pmatrix} \frac{\partial ll}{\partial \beta} \\ \frac{\partial ll}{\partial \theta_j} \end{pmatrix}$$

We first show the first-derivative w.r.t. β.

$$\frac{\partial ll}{\partial \beta} = \sum_{i=1}^{n} \sum_{j=1}^{J} I(y_i = j) \frac{f(t_{j-1}) - f(t_j)}{\pi_{ij}} x_i$$

where, $f(t_{j-1}) = f(\theta_j - \eta_i)$ the pdf of the logistic function at $(\theta_j - \eta_i)$, $\eta_i = x_i\beta$

We now show the first-derivative w.r.t. θ_j. The derivative for the rest of the categories follow the same form.

$$\frac{\partial ll}{\partial \theta_j} = \sum_{i=1}^{n} I(y_i = j)(\frac{f(t_k)}{\pi_{ik}} - \frac{f(t_k)}{\pi_{i,k+1}})$$

Note that each i will contribute to only one partial derivative calculation.

Substituting the log-likelihood and the first derivative functions to optimx, we can obtain the parameter estimates for both the β's and the θ_j's. BFGS requires initialization of the parameters.

Initialization of parameters

The β coefficients were all initialized at 0 (including the intercept), whilst the θ intercepts were initialized empirically. In further detail, the θ_j's were initialized as follows:

Step 1: Find the cumulative probabilities $P(Y \leq j)$ for $j = 1, ..., 5$.

Step 2: Find the quantiles of these cumulative probabilities in a logistic distribution.

These quantiles are which were used as the initial estimates. The rationale behind using this was (i) to make use of the data at hand, and (ii) to take into account the relationship between the θ_j's and the cumulative probabilities.

We did not try different initializations because (i) the algorithm converged very fast, and (ii) the output from using this approached matched with that produced by the 'polr' function by R.

Class prediction

Once we estimate the parameters laid out in our model, we perform class prediction. This was done as follows:

Step 1: Obtain the θ_j intercepts as laid out above.

Step 2: Now predict the class. Let c be the predicted class, and $P(Y_i < j)$ be the estimate of the probability of individual i's category being $\leq j$.

$$c = \begin{cases} 0, & P(Y_i \leq 0) \leq \theta_j \\ 1, & \theta_0 \leq P(Y_i \leq 1) \leq \theta_1 \\ 2, & \theta_1 \leq P(Y_i \leq 2) \leq \theta_2 \\ 3, & \theta_2 \leq P(Y_i \leq 3) \leq \theta_3 \\ 4, & P(Y_i \leq 4) \geq \theta_3 \end{cases}$$

Once prediction has been carried out as above, heatmaps in the form of a confusion matrix was derived for both the training and test data.

In addition to this, we used (i) misclassification distance, (ii) accuracy, (iii) weighted Kappa score (with quadratic weights) metrics to measure model performance in both the training and test data.

The weighted Kappa score holds a similar definition to that in the binary framework. This score assigns different weights to the disagreements between predicted and observed values based on the magnitude of the disagreement.

As a step further to what was laid out above, we also looked into model performance after collapsing categories $1-4$ to one category. That is, we dichotomized our outcome and then looked at model performance. Model performance was measured using exactly the same three metrics used for POLR.

Model Selection

One last procedure that was conducted in our analysis was that of model selection using AIC. In comparison to forward/backward selection which proceeds only in one direction, stepwise AIC looks to iteratively add and subtract predictors to find the model with the lowest Akaike Information Criterion (AIC). AIC penalizes for having too many covariates in the model, and therefore provides a more unbiased justification towards model selection compared to metrics such as R^2 or Adjusted R^2.

We derived a heatmap (with a confusion matrix) and computed the misclassification distance after selecting the best model using AIC.

2 (iv) Ordinal Random Forest

We used an ordinal random forest model to predict heart disease status based on a variety of clinical and demographic predictors. This method is designed specifically for ordinal outcomes—where the response variable has a natural ordering but unknown spacing between classes. The outcome variable has five ordered levels (0–4), where 0 indicates no heart disease and higher values indicate heart disease of increasing severity.

The model was implemented using the `ordinalForest` R package. This package works by assigning a numeric score to each class level. These scores are predicted using random forests and then mapped back to the class labels. This makes the model functionally similar to regression approaches, while still handling classification tasks with ordered outcomes.

We pre-processed our dataset by formatting the predictors (x) and outcomes (y) into the appropriate format. We used both the original five-level outcome and a simplified binary outcome (0 = no heart disease, 1+ = heart disease of any level) to evaluate model performance under different class definitions. We split our data into a training set (80% of samples) and a test set (20% of samples) and ran all model fitting on the training data.

Hyperparameter tuning

Hyperparameter tuning was conducted using 5-fold cross-validation to minimize the mean absolute error (MAE) between the predicted score and the true class label. We considered two of the hyperparameters for tuning: (1) the minimum node size, indicating the smallest number of observations allowed in a terminal node, and (2) the minimum number of decision trees trained. We tested combinations of minimum node size (ranging from 2 to 32) and minimum number of trees (ranging from 10 to 1000). The combination with the minimum average MAE across the 5-folds was selected for final model training using the entire training dataset.

2 (v) R package

We made an R package called "heaRt" that uses Newton Raphson and BFGS to fit and implement the logistic regression and proportional odds cumulative-logit models with only a single function called "glm_v2". To use this function from our package, a user simply specifies a model formula (e.g. Y ~ X1 + X2), a dataframe, and whether they would like to do logistic or proportional odds logistic regession. This function returns the estimated effects of the covariates, the probabilities between different groups (for POLR only), the log-likelihood value, and the time it took to fit. Note that random forest was not included in the R package because we are not implementing it from scratch.

3 Results

3 (i) Logistic Regression Model

The table presents the variables after stepwise AIC selection, and their estimated coefficients from the logistic regression model with Newton Raphson method. It shows major vessels involved was strongly associated with the outcome, especially when two vessels were involved (ca2, coefficient = 3.183). The other major contribution to heart disease is being male (Sex = 1), which increased the log-odds by 1.398.

```
# Stepwise AIC
lr_formula_full = y_bin ~ age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope+ca+thal
pop_lr_fit = glm(lr_formula_full, data = train, family = binomial())
stepwise_mod <- stepAIC(pop_lr_fit, direction = "both", trace = FALSE)
# Logistic model + Newton Raphson
# Training set
lr_formula = y_bin ~ sex+cp+trestbps+thalach+oldpeak+slope+ca+thal
lr_fit = glm_v2(lr_formula, df = train, method = "lr")
```

```
## Algorithm converged in 5 iterations.
```

```
## train matrix
lr_train_x <- model.matrix(lr_formula, data = train)
colnames(lr_train_x)[colnames(lr_train_x) == "(Intercept)"] <- "Intercept"
lr_train_x <- lr_train_x[, names(lr_fit$beta)]
lr_train_y <- train[,ncol(train)]
## predict probabilities
lr_eta_train <- lr_train_x %*% lr_fit$beta
lr_p_train <- 1 / (1 + exp(-lr_eta_train))
lr_pred_train <- ifelse(lr_p_train >= 0.5, 1, 0)
## Confusion matrix
lr_conf_train <- table("Prediction" = lr_pred_train, "True Label" = lr_train_y) |> as.matrix()
## Normalize (row-wise proportion: prediction rows)
lr_conf_train_prop <- t(apply(lr_conf_train, 1, function(i) i / colSums(lr_conf_train)))
## Calculate kappa
lr_train_kappa <- psych::cohen.kappa(lr_conf_train)
paste0("Weighted Cohen's Kappa for Training set is:", round(lr_train_kappa$weighted.kappa,2))
```

```
## [1] "Weighted Cohen's Kappa for Training set is:0.73"
```

```r
# Testing set
## test matrix
lr_test_x <- model.matrix(lr_formula, data = test)
colnames(lr_test_x)[colnames(lr_test_x) == "(Intercept)"] <- "Intercept"
lr_test_x <- lr_test_x[, names(lr_fit$beta)]
lr_test_y <- test[,ncol(test)]
## predict probabilities
lr_eta_test <- lr_test_x %*% lr_fit$beta
lr_p_test <- 1 / (1 + exp(-lr_eta_test))
lr_pred_test <- ifelse(lr_p_test >= 0.5, 1, 0)
## Confusion matrix
lr_conf_test <- table("Prediction" = lr_pred_test, "True Label" = lr_test_y) |> as.matrix()
## Normalize (row-wise proportion: prediction rows)
lr_conf_test_prop <- t(apply(lr_conf_test, 1, function(i) i / colSums(lr_conf_test)))
## Calculate kappa
lr_test_kappa <- psych::cohen.kappa(lr_conf_test)
paste0("Weighted Cohen's Kappa for Testing set is:", round(lr_test_kappa$weighted.kappa,2))
```

```
## [1] "Weighted Cohen's Kappa for Testing set is:0.64"
```

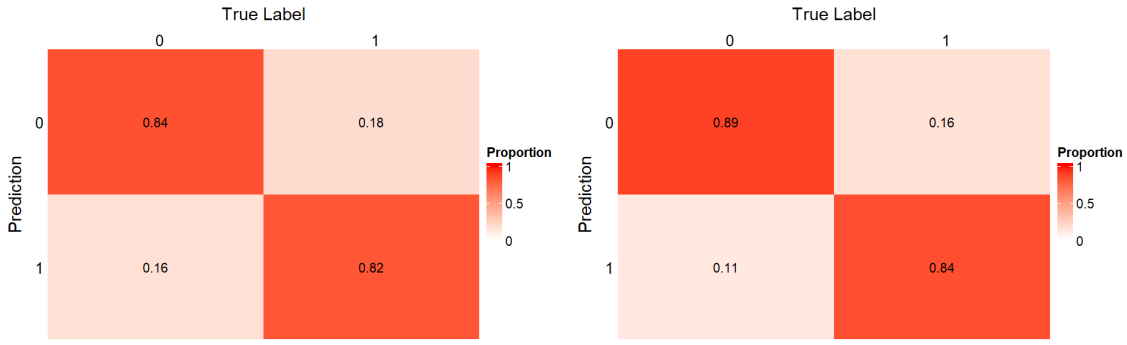
```r
knitr::include_graphics("results/figures/stepwise_lr_coefs.jpg")
```

Variable	Coefficient Estimate
Intercept	-6.716
Resting Blood Pressure (trestbps)	0.031
Maximum Heart Rate Achieved (thalach)	-0.016
ST Depression (oldpeak)	0.535
Male (Sex = 1)	1.398
Chest Pain Type	1.234 (cp2), 0.389 (cp3), 2.635 (cp4)
Slope of ST Segment	1.341 (slope2), -0.028 (slope3)
Major Vessels Involved	1.937 (ca1), 3.183 (ca2), 1.953 (ca3)
Thalassemia Type	-0.593 (thal6), 1.107 (thal7)

Figure 2: Caption for the image

To assess and compare the classification performance of different models, we calculated Cohen's kappa statistic. Kappa measures the agreement between predicted and true class labels while adjusting for agreement expected by chance. A kappa value of 1 indicates perfect agreement, while a value near 0 suggests agreement no better than random guessing. The weighted Cohen's Kappa for training and testing set of Logistics regression model is 0.73 and 0.64, indicating good predictive performance.

The heatmaps below illustrate the model performance on both the training and testing datasets, evaluating how well the model generalizes to unseen data and identify any potential overfitting or underfitting issues. Consistency between the two heatmaps indicates stable model performance.



3 (ii) Proportional Odds Cumulative-Logit Model (POLR)

Following the logistic model, our proportional odds logistic model captures the ordinal nature of our outcome. We fit a proportional odds logistic model fitted with all 14 predictors and our response with severity values from 0-4, where 0 indicates no heart disease and 1-4 indicate increasing heart disease.

```
polr_formula = y_mult ~ age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope+ca+thal
polr_fit = glm_v2(polr_formula, df = train, method = "polr")
```

```
## Maximizing -- use negfn and neggr
```

```
## fit prop odds on training
## Create X matrix
polr_train_x <- model.matrix(polr_formula, data = train)[, -1]
polr_train_x <- polr_train_x[, names(polr_fit$beta)] # Reorder columns
## Linear predictor
polr_eta_train <- polr_train_x %*% polr_fit$beta
## Predict class
n_class <- length(polr_fit$theta) + 1
###  get default levels 1,2,3,4,5
polr_temp_pred_train <- cut(as.numeric(polr_eta_train),
                 breaks = c(-Inf, polr_fit$theta, Inf), #cut by theta
                 right = TRUE)  # gives levels 1,2,3,4,5
###  subtract 1 to match 0-4
polr_train_pred <- as.numeric(polr_temp_pred_train) - 1
## True labels for training
polr_train_y <- train$y_mult
## Confusion matrix
polr_train_table <- table("prediction" = polr_train_pred, "true label" = polr_train_y) |> as.matrix()
## Normalize to proportions
polr_train_table_prop <- t(apply(polr_train_table, 1, function(i) i / colSums(polr_train_table)))

#testing follows same procedure but with testing dataset
```

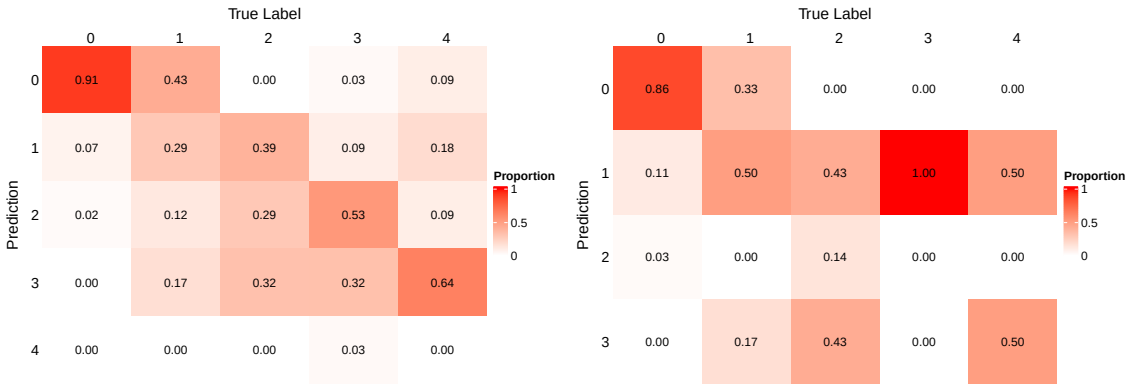


Figure 3: Figure: Heatmap of the Training Data Performance (Left) and Testing Data Performance (Right)

From our fitted model, we can see that the model does relatively well at predicting the participants with no heart disease. In the training dataset we accurately predict 91% of the no heart disease participants.In the testing dataset, we see that we accurately predict 86% of the no heart disease participants.

In evaluating our model, we look at the test set performance primarily. Initially, we see that there were no participants predicted to be a 4 in severity, the most severe case. Another thing to note is that our model seems to be biased towards zero. We see that 33% of participants the 1 severity were underdiagnosed by our model of being severity level 0 (no disease). Similarily 43% of participants with level 2 severity were predicted to have level 1 severity and 100% of participants with level 3 severity were predicted to have level 1 severity. Ideally our model would have 100% accuracy and this would indicate a heatmap with a perfect diagonal. We see that our proportional odds model often strays from this pattern, indicating some missclassification.

In our testing dataset we can report a generic accuracy for the model. With around 60% accuracy in the training and testing dataset, our model does not perform super well.

Table 4: Model accuracy on training and testing sets.

Dataset	Accuracy
Training	60.08%
Testing	66.10%

To more accurately our evaluate the performance of our model we can look at difference in the predicted versus observed values of our model.

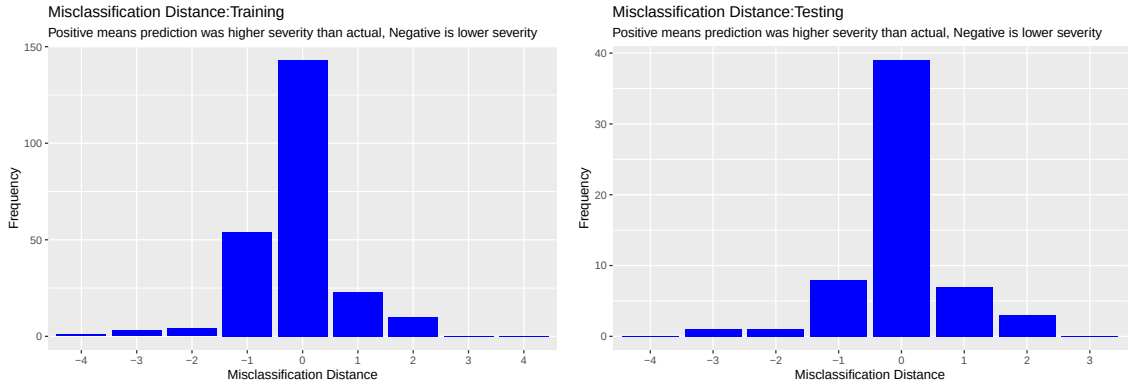


Figure 4: Figure: Distance of Misclassification in the Training Dataset (Left) and Testing Dataset (Right)

When looking at these figures we can visualize our data misclassification in a different way than the heat maps. We can see that our models misclassify most frequently by one category and also by predicting a lower severity than the true severity in both the training and the testing set. This is incredibly evident in the training set.

Similarly to a kappa score with binary outcomes, we can use a weighted kappa score with quadratic weights to evaluate the fit of our model. The weighted kappa assigns different weights to the disagreements between predicted and observed values based on the magnitude of the disagreement. The weighted cohen kappa score shown in the table indicate a relatively poor performance from our model.

Table 5: Weighted Cohen Kappa Score on training and testing sets with quadratic weights.

Dataset	WeightedKappa
Training	.523
Testing	.420

Collapsing the Proportional Odds Model into a Binary Outcome

In order to compare the results of our analysis of dichotomous disease outcomes analyzed with logistic regression, we collapsed the predictions from our proportional odds models into two categories after prediction. To have consistency we collapsed the data into participants with no heart disease (severity level 0) and with heart disease (severity levels 1-4).

```
#refit with binary outcome
polr_train_y_numeric <- as.numeric(as.character(polr_train_y))
polr_train_pred_numeric <- as.numeric(as.character(polr_train_pred))
polr_train_y_binary <- ifelse(polr_train_y_numeric > 0, 1, 0) #make binary
polr_train_pred_binary <- ifelse(polr_train_pred_numeric > 0, 1, 0)
polr_train_binary_table <- table("prediction" = polr_train_pred_binary, "true label" = polr_train_y_bina
polr_train_binary_table_prop <- t(apply(polr_train_binary_table, 1, function(i) i / colSums(polr_train_b
#set up is same for testing
```

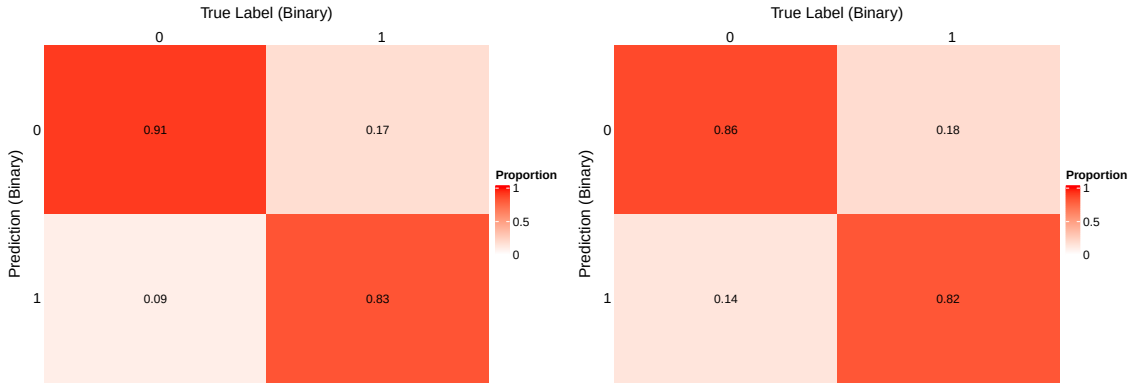


Figure 5: Figure: Heatmap of the Training Data Performance (Left) and Testing Data Performance (Right) using dichotomous outcomes and proportional odds model

We see in the heatmaps that the binary classification of on the backend of our proportional odds model leads to a good performance on both the training and the testing set. The performance is very similar to the results found using logistic model. The accuracy is in the 80%s for both the training and the testing, as seen in the table.

Table 6: Model accuracy on training and testing sets with dichotomized outcomes.

Dataset	Accuracy
Training	86.97%
Testing	84.75%

Again, another way to evaluate the fit is through Cohen's Kappa statistics. We see that the fit is much better with the binary outcome as the kappa score is much closer to 1 than the values were with the ordinal outcome measures.

Table 7: Cohen's Kappa on training and testing sets with dichotomized outcomes.

Dataset	Accuracy
Training	0.74
Testing	0.68

Model Selection with Stepwise AIC

After evaluating our model we performed model selection by doing stepwise AIC model selection. This method iteratively adds and subtracts predictors from the model to find the model with the lowest AIC value. Our stepwise model selection chose a model with the following predictors and their corresponding estimates and odds ratios.

```
#compare to use off the shelf for polr
pop_mod <- polr(y_mult ~ age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope+ca+thal , data
stepwise_mod <- stepAIC(pop_mod, direction = "both", trace = FALSE)
#summary(stepwise_mod) # instead of this, see nice table below made from this
```

Variable	Coef.	Std. Error	t value	OR
sex (male)	0.91161	0.3780	2.4119	2.488
chest pain (2)	0.60386	0.7275	0.8301	1.829
chest pain (3)	0.09619	0.6566	0.1465	1.101
chest pain (4)	1.71115	0.6185	2.7666	5.535
resting bp	0.01443	0.0091	1.5934	1.015
resting ecg (1)	2.01962	1.1525	1.7524	7.535
resting ecg (2)	0.56422	0.3011	1.8740	1.758
max HR	-0.01474	0.0076	-1.9316	0.985
ST segment depression	0.34380	0.1522	2.2590	1.410
Segment slope (flat)	0.88132	0.3798	2.3207	2.414
Segment slope (down)	0.27732	0.6773	0.4095	1.320
Number of Vessels (1)	1.21095	0.3834	3.1585	3.357
Number of Vessels (2) 2	2.16130	0.4452	4.8548	8.682
Number of Vessels (3) 3	2.36401	0.5630	4.1991	10.633
Thal Defect (fixed)	0.12332	0.6305	0.1956	1.131
Thal Defect (reversable)	1.09726	0.3543	3.0969	2.996
Thresholds	**Coef.**	**Std. Error**	**t value**	**OR**
0—1	3.4851	1.9456	1.7912	32.625
1—2	5.2179	1.9621	2.6593	184.555
2—3	6.4529	1.9747	3.2678	634.515
3—4	8.6668	2.0175	4.2959	5806.884

Table 3: Model Coefficients, Standard Errors, t-values, and Odds Ratios

Figure 6: Figure: Stepwise Selection for proportional odds logit regression model

We can see that the model keeps most of the predictors, only dropping predictor variables for serum cholesterol, fasting blood sugar, and exercise induced angina presence.

Using our model, we can visualize the perfomance of our model on the test set with this heatmap.

Again, the model does a good job at predicting participants with no disease, but generally poor performance for classifying the level of severity of disease outside of no disease. Similar to before we can see that our this model does not predict any level 4 severity and that our model is biased towards 0.

When evaluating the misclassification distance for this model we see that the model is typically off by a severity level of 1, and slightly overdiagnosing in this model. It makes sense that the fit is not significantly different than the full model because not many changes were made with the predictors in the stepwise AIC model.

Overall, the proportional odds model had an overall poor performance, both using the full model and a nested model determined through stepwise AIC selection.

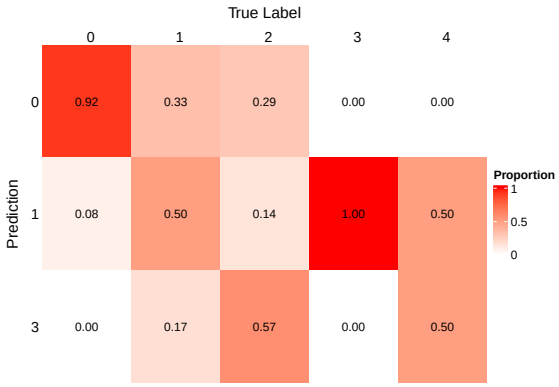


Figure 7: Figure: Heatmap of Stepwise Selection Proportional Odds Model on the Testing Dataset

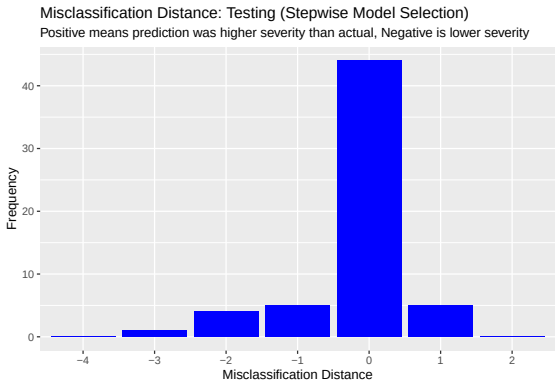


Figure 8: Figure: Misclassification distance from the stepwise proportional odds model on the test dataset

3 (iii) Ordinal Random Forest Model

Data loading and formatting

```
train <- read.csv("./data/derived/df_v1_TRAIN.csv")
test <- read.csv("./data/derived/df_v1_TEST.csv")
## build the train dataset
train %<>%
    dplyr::select(-X) %>%
    dplyr::mutate(y_mult = factor(num),
                  y_bin = factor(ifelse(num == "0", 0, 1)),
                  dplyr::across(c(sex,cp, fbs, restecg, exang, slope, ca, thal), as.factor)
    ) %>%
    dplyr::select(-num)

## build the test dataset
test %<>%
    dplyr::select(-X) %>%
    dplyr::mutate(y_mult = factor(num),
                  y_bin = factor(ifelse(num == "0", 0, 1)),
                  dplyr::across(c(sex,cp, fbs, restecg, exang, slope, ca, thal), as.factor)
    ) %>%
    dplyr::select(-num)

# separate into predictors and outcomes
x_train <- train[, -which(names(train) %in% c("y_mult", "y_bin"))]
y_train_mult <- train$y_mult
y_train_bin <- train$y_bin

# separate into predictors and outcomes
x_test <- test[, -which(names(test) %in% c("y_mult", "y_bin"))]
y_test_mult <- test$y_mult
y_test_bin <- test$y_bin

# Metadata has descriptions of the variables
meta <- read.csv(file = "./data/derived/var_info_upd.csv")
meta$name_plot <- c("age", "sex", "chest pain", "resting BP", "serum cholesterol", "fasting blood sugar

# default values
ntree_grid <- c(10, 50, 100, 500, 1000)
nodesize_grid <- c(2, 3, 4, 6, 8, 12, 16, 24, 32)
```

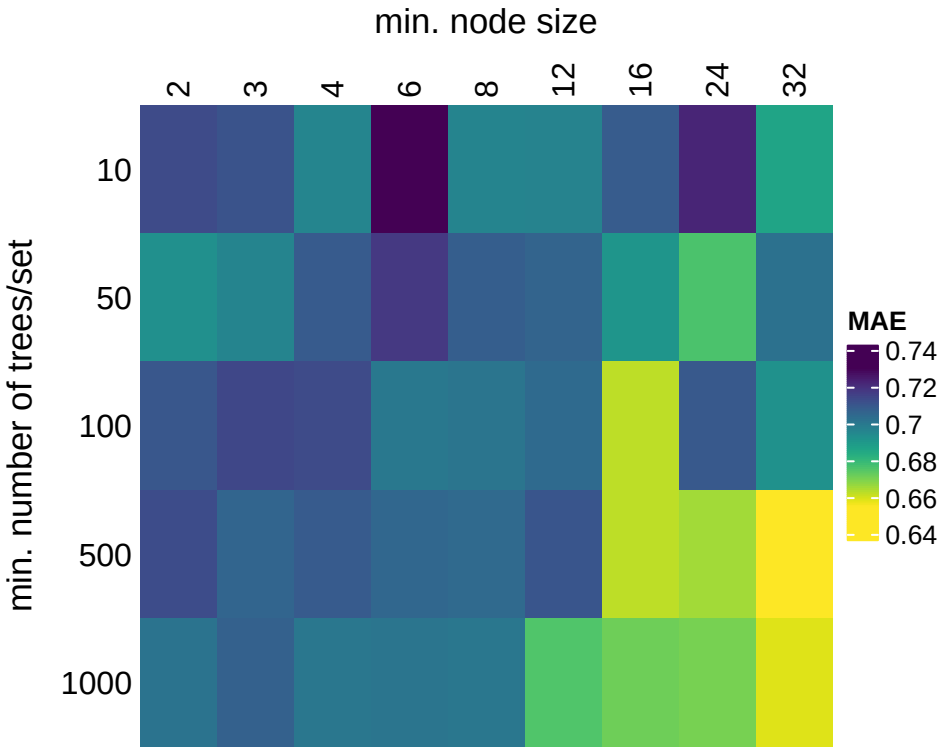
We fit the ordinal random forest model using the `fit_rf` function.

```
# Fit ordinal model
# Takes a while
# res_multi <- fit_rf(x_train, y_train_mult)
```

Hyperparameter tuning

We first examined the results of the hyperparameter tuning.

```
Heatmap(res_multi$mean_mae, col = rev(viridis(n = 100)),
        cluster_rows = FALSE,
        cluster_columns = FALSE,
        name = "MAE",
        row_title = "min. number of trees/set",
        row_labels = c(10, 50, 100, 500, 1000),
        row_names_side = "left",
        column_title = "min. node size",
        column_labels = c(2, 3, 4, 6, 8, 12, 16, 24, 32),
        column_names_side = "top")
```

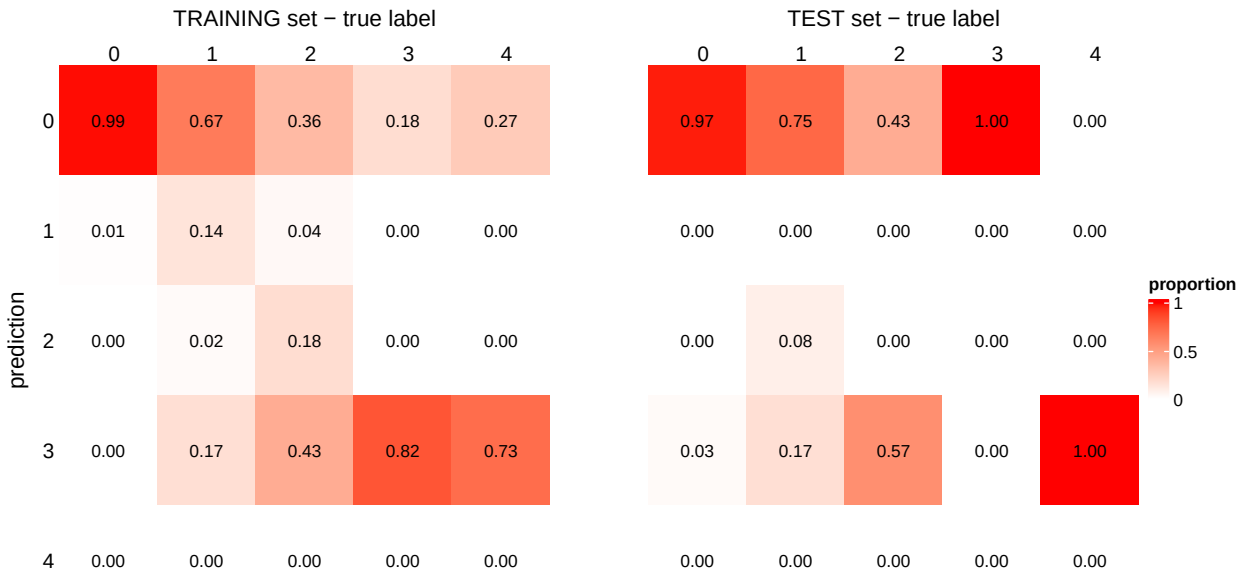


Cross-validation revealed the lowest mean absolute error (MAE) when using 500 trees and a minimum node size of 32. Notably, the optimal minimum node size was the largest value we considered given that this reflects >10% of the training dataset size. This may reflect the general difficulty in predicting an ordinal outcome with imbalanced classes.

Predicting ordinal outcome measurement (0, 1, 2, 3, 4)

```
par(mfrow = c(1, 2))
df_multi_train <- cbind(x_train, y_train_mult) |> as.data.frame()
names(df_multi_train)[ncol(df_multi_train)] <- "Class"
pred_multi_train <- predict(object = res_multi$fit, newdata = df_multi_train)

pred_multi_train_table <- table("prediction" = pred_multi_train$ypred, "true label" = df_multi_train$Cl
pred_multi_train_table_prop <- t(apply(X = pred_multi_train_table, MARGIN = 1, FUN = function(i) i / co
```

```r
# Heatmap of confusion matrix for TRAINING data with all classes
hm_train <- Heatmap(pred_multi_train_table_prop,
        col = colorRamp2(breaks = c(0, 1), colors = c("white", "red")),
        name = "proportion",
        cluster_rows = FALSE, cluster_columns = FALSE,
        row_title = "prediction", row_names_side = "left",
        column_title = "TRAINING set - true label", column_names_side = "top", column_names_rot = 0,
        cell_fun = function(j, i, x, y, width, height, fill) {
            grid.text(sprintf("%.2f", pred_multi_train_table_prop[i, j]), x, y, gp = gpar(fontsize = 10)
        })

df_multi_test <- cbind(x_test, y_test_mult) |> as.data.frame()
names(df_multi_test)[ncol(df_multi_test)] <- "Class"
pred_multi_test <- predict(object = res_multi$fit, newdata = df_multi_test)

pred_multi_test_table <- table("prediction" = pred_multi_test$ypred, "true label" = df_multi_test$Class)
pred_multi_test_table_prop <- t(apply(X = pred_multi_test_table, MARGIN = 1, FUN = function(i) i / colSu

# Heatmap of confusion matrix for TEST data with all classes
hm_test <- Heatmap(pred_multi_test_table_prop,
        col = colorRamp2(breaks = c(0, 1), colors = c("white", "red")),
        name = "proportion",
        cluster_rows = FALSE, cluster_columns = FALSE,
        row_title = "prediction", row_names_side = "left",
        column_title = "TEST set - true label", column_names_side = "top", column_names_rot = 0,
        cell_fun = function(j, i, x, y, width, height, fill) {
            grid.text(sprintf("%.2f", pred_multi_test_table_prop[i, j]), x, y, gp = gpar(fontsize = 10))
        })

draw(hm_train + hm_test, ht_gap = unit(2, "cm")) |> suppressWarnings()
```



The random forest model overall performs poorly at predicting heart disease cases. On the training set, we

observed a correct prediction of no heart disease for 99% of no heart disease cases, but much lower accuracy for heart disease levels 1 - 4. Similarly, on the test set, while we correctly predict 97% of the no heart disease cases, we underpredict for the vast majority of the positive heart disease cases.

Weighted kappa

```
## Weighted Cohen's Kappa

## Training set:     0.718

## Test set:     0.583
```

Predicting binary outcome measurement (0, 1+)

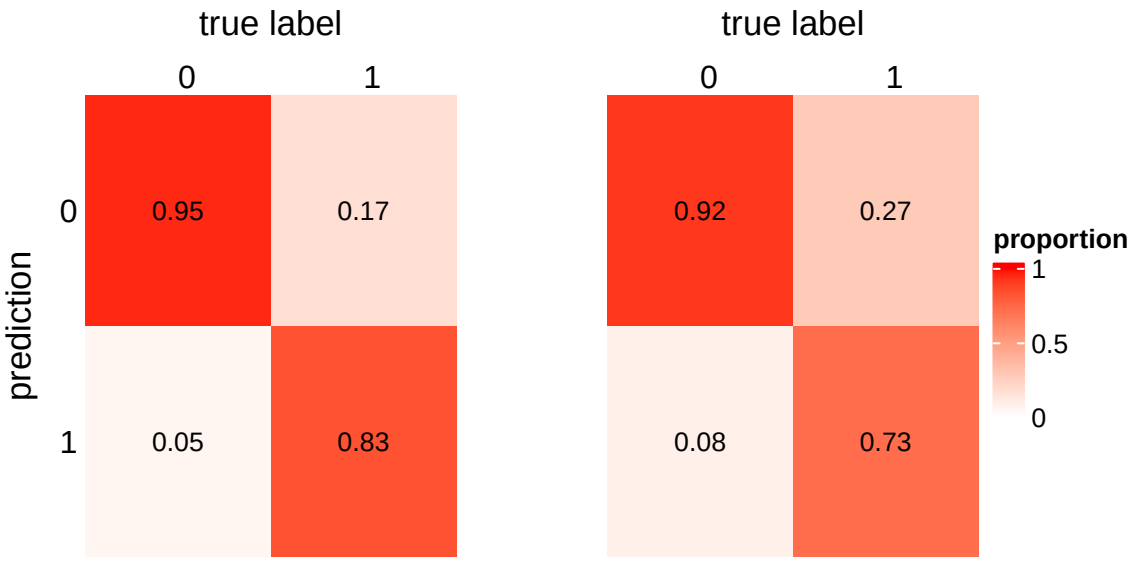
Given the poor performance on the ordinal outcome, we re-framed the problem as a binary classification: no heart disease (0) versus any heart disease (1+). This framing produced a much more balanced class distribution and significantly improved performance.

We fit the ordinal random forest model using the `fit_rf` function.

```
# Fit ordinal model but instead let's classify into the binary output (0, 1+)
# Takes a while
res_binary <- fit_rf(x_train, y_train_bin)
```

```
df_binary_train <- cbind(x_train, y_train_bin) |> as.data.frame()
names(df_binary_train)[ncol(df_binary_train)] <- "Class"
pred_binary_train <- predict(object = res_binary$fit, newdata = df_binary_train)

pred_binary_train_table <- table("prediction" = pred_binary_train$ypred, "true label" = df_binary_train$
pred_binary_train_table_prop <- t(apply(X = pred_binary_train_table, MARGIN = 1, FUN = function(i) i / 

# Heatmap of confusion matrix for TRAINING data with binary outcome
hm_train <- Heatmap(pred_binary_train_table_prop,
        col = colorRamp2(breaks = c(0, 1), colors = c("white", "red")),
        name = "proportion",
        cluster_rows = FALSE, cluster_columns = FALSE,
        row_title = "prediction", row_names_side = "left",
        column_title = "true label", column_names_side = "top", column_names_rot = 0,
        cell_fun = function(j, i, x, y, width, height, fill) {
            grid.text(sprintf("%.2f", pred_binary_train_table_prop[i, j]), x, y, gp = gpar(fontsize = 1
        })

df_binary_test <- cbind(x_test, y_test_bin) |> as.data.frame()
names(df_binary_test)[ncol(df_binary_test)] <- "Class"
pred_binary_test <- predict(object = res_binary$fit, newdata = df_binary_test)

pred_binary_test_table <- table("prediction" = pred_binary_test$ypred, "true label" = df_binary_test$Cla
pred_binary_test_table_prop <- t(apply(X = pred_binary_test_table, MARGIN = 1, FUN = function(i) i / col

# Heatmap of confusion matrix for TEST data with binary outcome
hm_test <- Heatmap(pred_binary_test_table_prop,
        col = colorRamp2(breaks = c(0, 1), colors = c("white", "red")),
        name = "proportion",
```

```
        cluster_rows = FALSE, cluster_columns = FALSE,
        row_title = "prediction", row_names_side = "left",
        column_title = "true label", column_names_side = "top", column_names_rot = 0,
        cell_fun = function(j, i, x, y, width, height, fill) {
            grid.text(sprintf("%.2f", pred_binary_test_table_prop[i, j]), x, y, gp = gpar(fontsize = 10)
        })

draw(hm_train + hm_test, ht_gap = unit(2, "cm")) |> suppressWarnings()
```



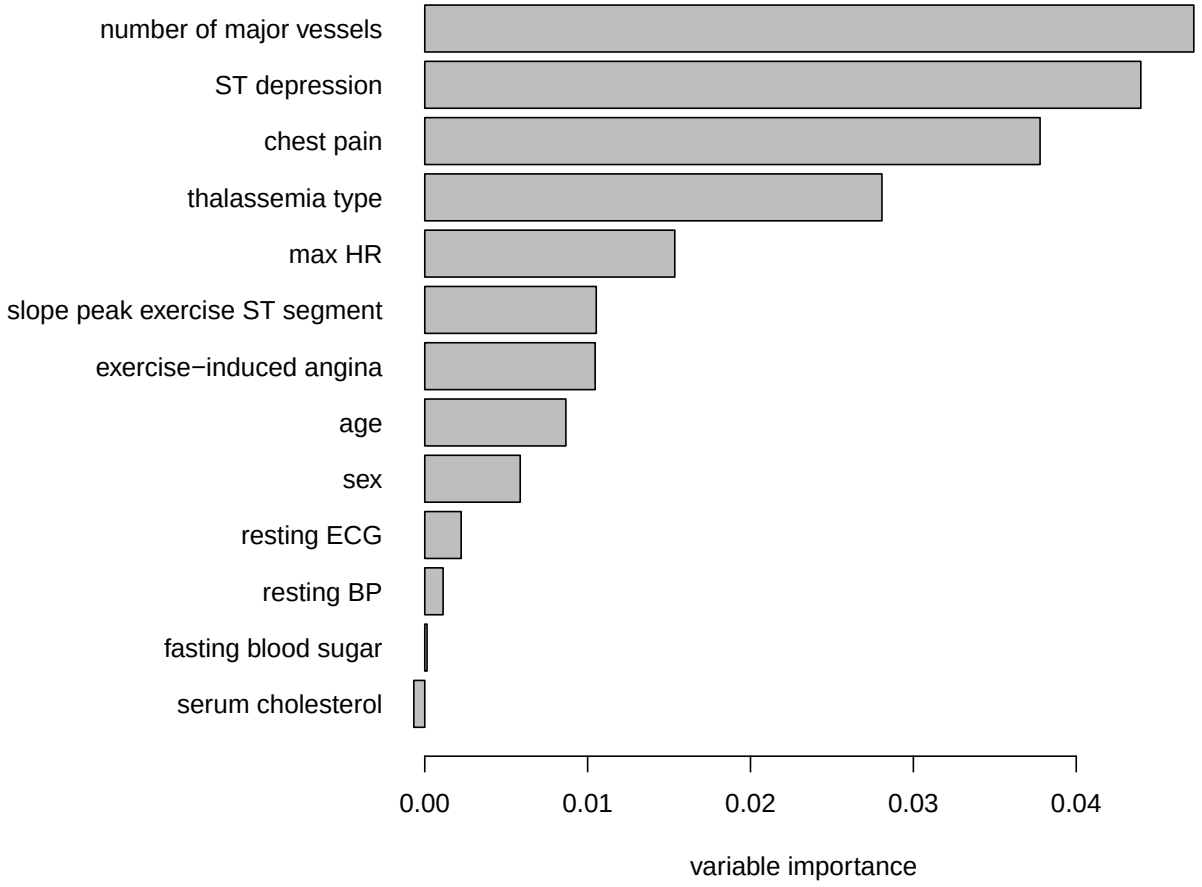
Confusion matrices demonstrated strong performance on both training and test data. The model was able to distinguish between healthy individuals and those with heart disease with high reliability. These results suggest that predicting the presence of heart disease is a relatively easy task, while predicting the severity of heart disease is much more challenging.

Weighted kappa

```
## Weighted Cohen's Kappa

## Training set:     0.789

## Test set:     0.665
```

Variable importance

```
par(mar = c(4, 14, 1, 1))
barplot(sort(res_binary$fit$varimp),
        horiz = TRUE, las = 1,
        names.arg = meta$name_plot[match(x = names(sort(res_multi$fit$varimp)), table = meta$name)], xla
```



From this figure we see that the number of major vessels found with calcium deposits, ST depression induced by exercise, chest pain type, and thalassemia type all had relatively high influence on the models predictions.

4 Discussion/Conclusion

In the binary outcome case, the three models did moderately well. The Kappa values for logistic regression, proportional odds, and random forest were 0.64, 0.68, and 0.67, respectively. Thus the binary models had similar performance with proportional odds being slightly better in Kappa. The logistic regression model had the best recall (around 84%), which is an important metric clinically since a primary objective of the diagnosis task is to identify all (or nearly all) truely positive heart disease cases. In the multi-category outcome case, the proportional odds model (kappa of 0.42) and ordinal random forest (kappa of 0.58) struggled more. The ordinal random forest model successfully distinguished between individuals with and without heart disease when using a binary outcome, but failed to accurately predict multiple levels of disease severity.

These multi-category model limitations likely stems from a combination of class imbalance, a lack of differentiating features for disease severity, and the limited sample size. At the higher heart severity levels like 3 and 4, the sample size within these groups was very small. This constrains the model's ability to learn the data in the training dataset and accurately predict participants into these higher categories. This also explains some of the model's bias towards 0, which was the largest disease severity group. Additionally, since the poor performance was also found in the model selected by AIC stepwise selection, we can contribute some of the errors of the model due to the imbalanced data. This is because even the best fitting model was not a great fit.

In the future, we would advise our collaborators to aim for a larger sample size overall, but also within groups. We could also look into harmonizing the data across the many clinics investigating heart disease.

References

[1] Martin, SS., ... American Heart Association Council on Epidemiology and Prevention Statistics Committee and Stroke Statistics Subcommittee (2024). 2024 Heart Disease and Stroke Statistics: A Report of US and Global Data From the American Heart Association. Circulation, 149(8), e347–e913. https://doi.org/10.1161/CIR.0000000000001209