The Heart of the Issue: Predicting the Diagnosis of Heart Disease

Andra Velea, Aryan Mistry, Mateo Umaguing
University of California, Los Angeles
Professor Miles Chen, STATS 101C
August 2nd, 2022

1 Introduction

The purpose of this project was to act as a statistician to find a valid model with significant variables that best predicts the diagnosis of heart disease in adults. The provided training dataset contained one response variable, 13 predictors, and 227 observations based on data provided by the University of California, Irvine machine learning repository. We expected the age and cholesterol variables to be associated with the response variable, as intuitively cholesterol levels increase with age, but left other potential relationships to be discovered without initial predictions.

2 Exploratory Data Analysis (EDA)

As a starting point, we plotted all predictors grouped by response values and investigated these relationships (Figures 1, 2, and 4).

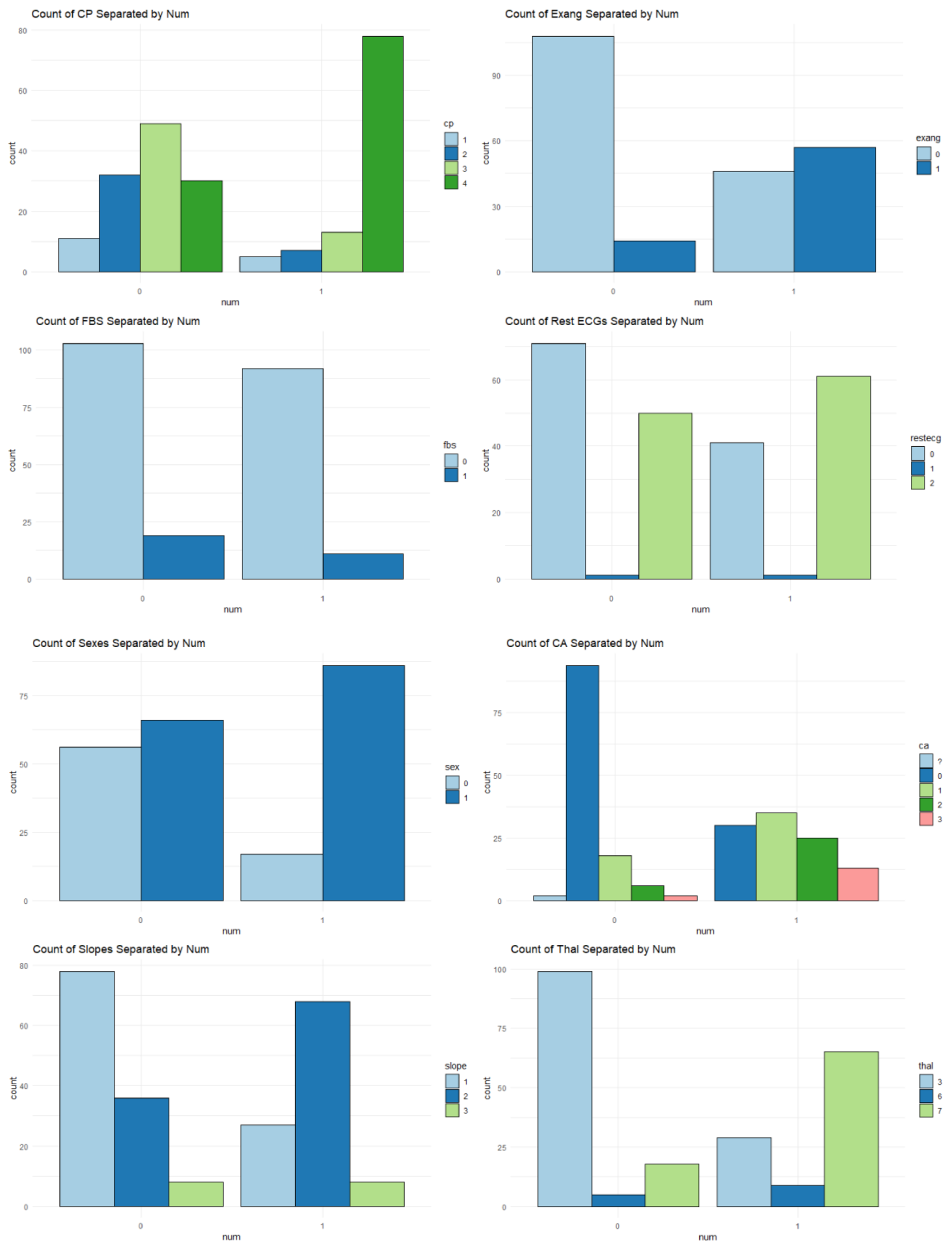

Figure 1. Bar plots of predictor counts separated by num = 0 versus num = 1. We selected predictors whose distribution of values was visibly different across the different values of the response. The ‘fbs’ and ‘restecg’ predictors have similar distributions across the different values of num, therefore they were excluded in the final model.

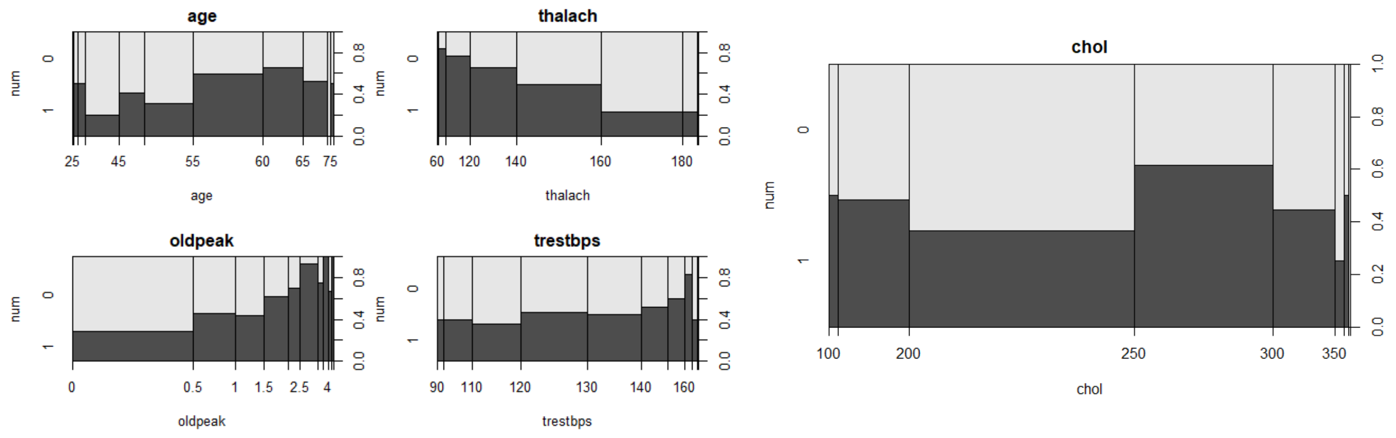

Figure 2. Plots of response versus all numeric predictors. Age, thalach, oldpeak, and trestbps all seem to be skewed, implying a log transformation may be necessary to normalize the data. The cholesterol variable seems to be insignificant as well, and maybe even trestbps.

The log transformation of Figure 2 is shown in Figure 3.

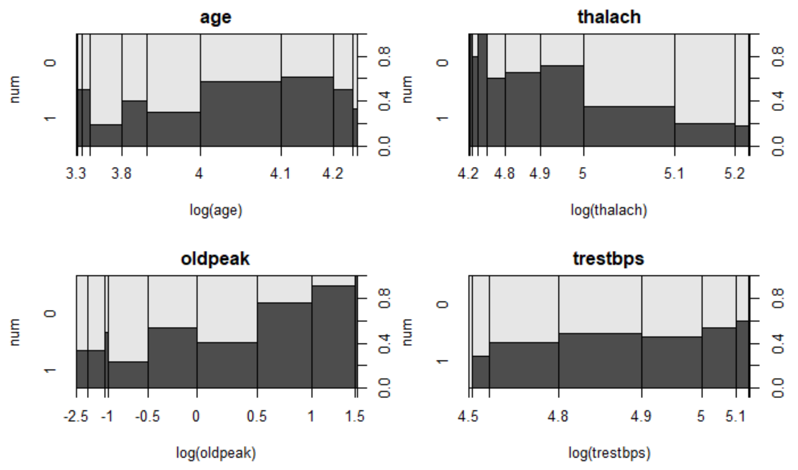

Figure 3. Response versus log transformation of some numeric predictors. It seems apparent that taking the log did not help in mitigating the skewness; in fact, all variables have maintained the same shape. However, trestbps continues to seem like an insignificant variable based on its shape.

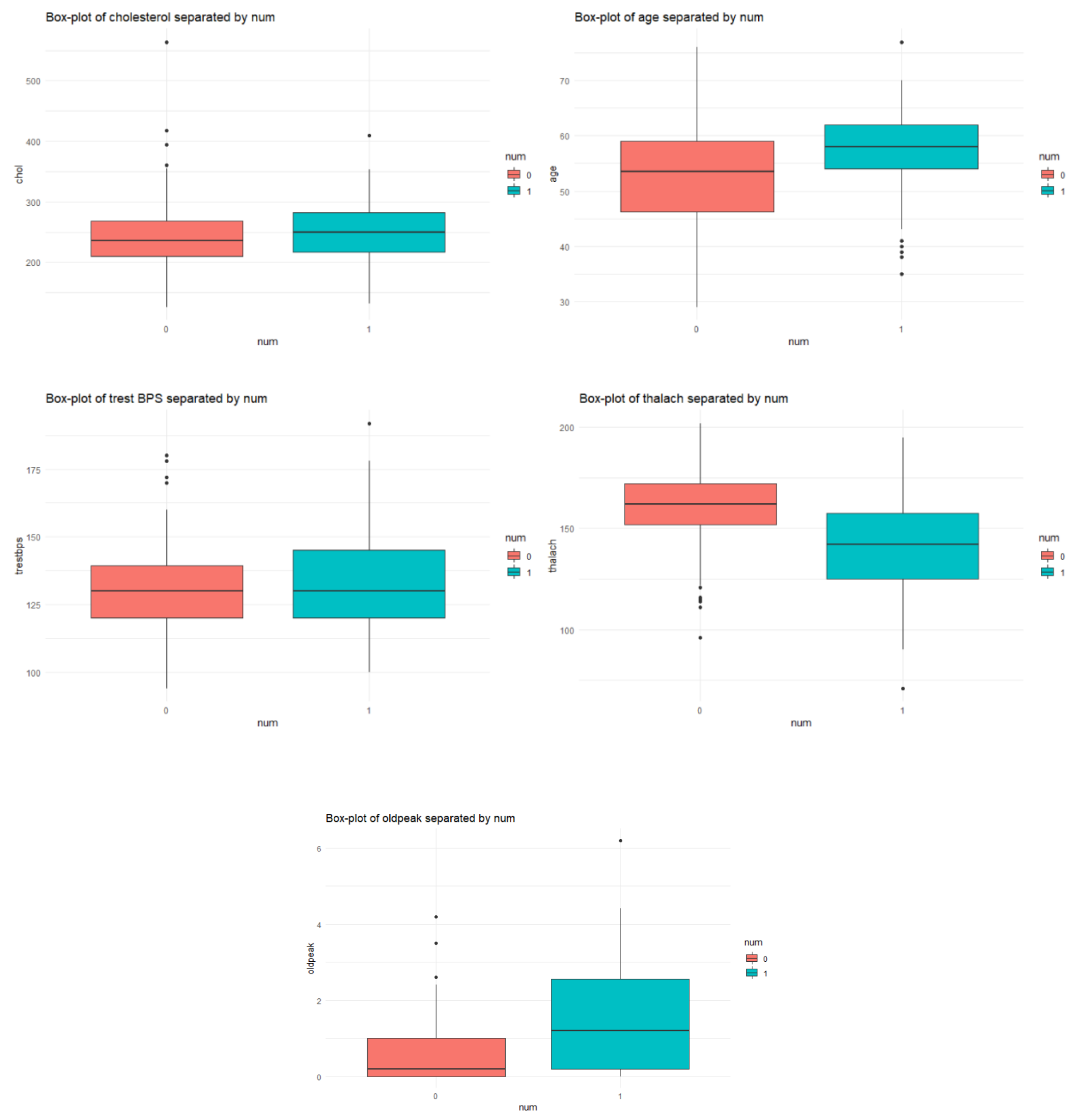

Figure 4. Box-plots of numeric predictors grouped by response. The 'trestbps' and 'chol' predictors did not vary across values of 'num'; therefore they were excluded in the final model.

Next, we examined our initial prediction for cholesterol versus age, as we expect the cholesterol level to increase as one gets older (Figure 5). However, the results seemed to show there is not, in fact, a relationship between these two variables.

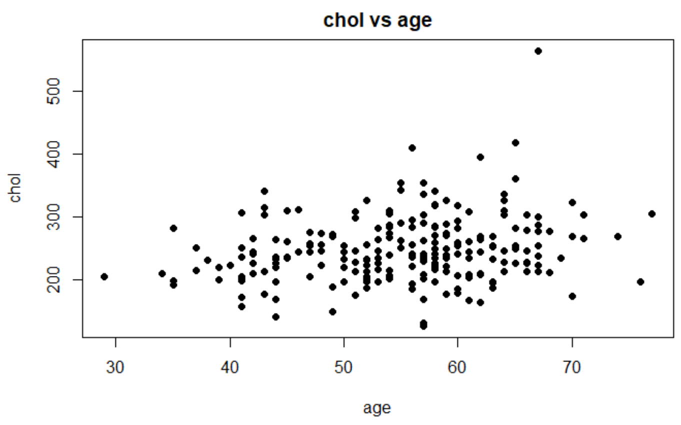

Figure 5. Cholesterol vs age. There does not appear to be any significant relationship or correlation between these variables, which is an interesting finding. Hence, our initial prediction yields no significance for the final model.

Furthermore, from Figures 1 and 2, we infer the variables slope, fbs, restecg, trestbps, and chol to be noise in predicting the response. In efforts to verify these claims, we ran a feature importance selection via the Boruta() function in the Boruta() library. The output supported our hypothesis, giving us four unimportant variables: chol, fbs, restecg, and trestbps with the remaining nine as important. The only difference was that we considered slope to be insignificant.

Moreover, we looked at the correlation matrix of the training data set for any potential interaction terms. Because the data did not have values between -1 and 1 (even after scaling), we could not produce a visual representation of the correlations. However, we manually inspected the correlation matrix. Many of the values did not go above 0.5, but even the maximum correlation was below 0.6. Hence, we deemed there to be no significant interactions between the variables.

3 Preprocessing/Recipes

The dataset needed fairly little in the way of preprocessing. The only major task was to type-cast the response variable. The response values were initially numbers stored as characters, so they were converted to integers. The integer values were then stored as factors. From the EDA (Section 2), it became evident that the columns id, trestbps, chol, fbs, and restecg were not significant predictors so a subset of the training dataset, heart_train2, was created without these variables. Lastly, we detected several "?' values within the dataset. These were converted to NA values by nature of as.numeric() in base R.

4 Candidate Models

Since the first model we submitted ended up being the best model, we chose four other models to discuss before the final one (Table 1).

Using the output from Boruta algorithm (see Section 2), we performed a logistic regression model (lr_model1) for classification on this subset of the training data. This did not include any interaction effects, but it did include two tuned hyperparameters, penalty and mixture, for optimal accuracy. The best logistic regression model was then chosen based on the best accuracy from 10-fold cross-validation.

We then used a support-vector-machine-based approach. Our initial model, stats101c_c1, used the "LiblineaR" engine and employed all available predictors. Since this was meant to be a preliminary test, the model did not use tuned hyperparameters, and the "cost" and "margin" parameters were set to standard values of 1 and 0.1, respectively. The model earned a poor training accuracy score of 0.686, likely due to its simplicity.

Next, we fit a support vector machine with a polynomial kernel on the data using all of the predictors as before and the "kernlab" engine. We grid-tuned the cost and degree hyperparameters using the 'grid_latin_hypercube()' and 'tune_grid()' functions and 10-fold cross validation. The flexibility and computational inexpensiveness of the model made it favorable to include as a candidate model.

We then ventured into gradient-boosted trees. One of our initial models, c2, was an xgboost algorithm. The model used every predictor available, but made extensive use of hyperparameter tuning and 5-fold cross-validation. The model with the best accuracy out of the 5 folds was chosen. Despite its very simple recipe and formula, the model performed exceedingly well in private scoring, although it ultimately was not used for the final selection because the xgb_1_classification model matched its score while using less predictors, thereby being less computationally expensive.

The final model, xgb_1_classification, incorporated gradient-boosted trees with the model engine "xgboost." Again, there were no interaction effects, and we tuned the two hyperparameters mtry and trees. Similar to lr_model1, xgb_1_classification was chosen based on the best accuracy from 10-fold cross-validation.

Table 1. Summary of selected models.

Model Identifier	**Model Type**	**Model Engine**	**Recipe/List of Predictors Used**	**Hyperparameters**
c2	Boosted Tree	xgboost	All	trees = 1808, tree_depth = 4, min_n = 10, loss_reduction = 6.05627e-05,

				sample_size = 0.5706762, mtry = 11, learn_rate = 0.0001252926
lr_model1	Logistic Regression	glmnet, family= "binomial"	All but trestbps, chol, fbs, restecg	Penalty = 0.07742637 Mixture = 1/3
svm_poly_mu	Support Vector Machine	kernlab	All	Cost = 0.694, Degree = 1.90
stats101c_c1	Support Vector Machine	LiblineaR	All	Cost = 1, Margin = 0.1
xgb_1_classification (final model)	Boosted Tree	xgboost	All but trestbps, chol, fbs, restecg	Mtry = 36 Trees = 1143 Default for the rest

5 Model Evaluation and Tuning

Lr_model1 produced one of the worst private scores from every submission (omitted). However, these results are not surprising; the low F-score aligns with the findings from Figure 3. Since log-transforming the data did not improve the distribution/skewness, it made sense for lr_model1 to poorly predict the response.

The stats101c_c1 support vector machine model did not employ hyperparameter tuning, or any purposeful variable selection. The "cost" and "margin" values were chosen as a baseline, but the model itself did not produce an adequate accuracy score on the training data. Although it performed decently in private scoring, earning an F-score of over 0.82, the SVM approach was abandoned in favor of boosted-tree models.

The model 'svm_poly_mu', which used a polynomial kernel and a different engine ("kernlab"), performed decently but not outstandingly well on both the testing and training data, demonstrating the robustness and flexibility of the SVM model. However, we decided to fit other models that had lower training performance but higher testing performance.

Due to their high performance, we focused on gradient-boosted trees for improving our prediction accuracy. The c2 boosted-tree model produced very good results in private scoring, but the model itself was fairly rudimentary. It used every predictor available, and did not make use of any recipes or transformations, although it did utilize 5-fold cross validation. It made use of 7 tuned hyperparameters, and the best subset was chosen based on highest accuracy. The model itself performed very well, but it was not chosen due to the fact that the xgb_1_classification model produced the exact same score, but with 4 fewer predictors and 5 fewer tuned hyperparameters, making it far more efficient.

To make sure our models were not overfitting to the training data and remained generalizable, we ran 10-fold cross validation, tuned hyperparameters, and compared all models against each other for the best accuracy. The model comparisons are summarized in Table 2, and the corresponding plot of accuracies can be found in Figure 6.

Table 2. Model comparisons.

Model Identifier	**F-Score (Private)**	**Accuracy**	**SE of Accuracy**
c2	0.89285	0.8429293	0.02045797
lr_model1	0.76785	0.8065217	0.02423929
svm_poly_mu	0.80357	0.8328063	0.01946013
stats101c_c1	0.82142	0.6860987	0.02726738
xgb_1_classification (final model)	0.89285	0.7934783	0.02316978

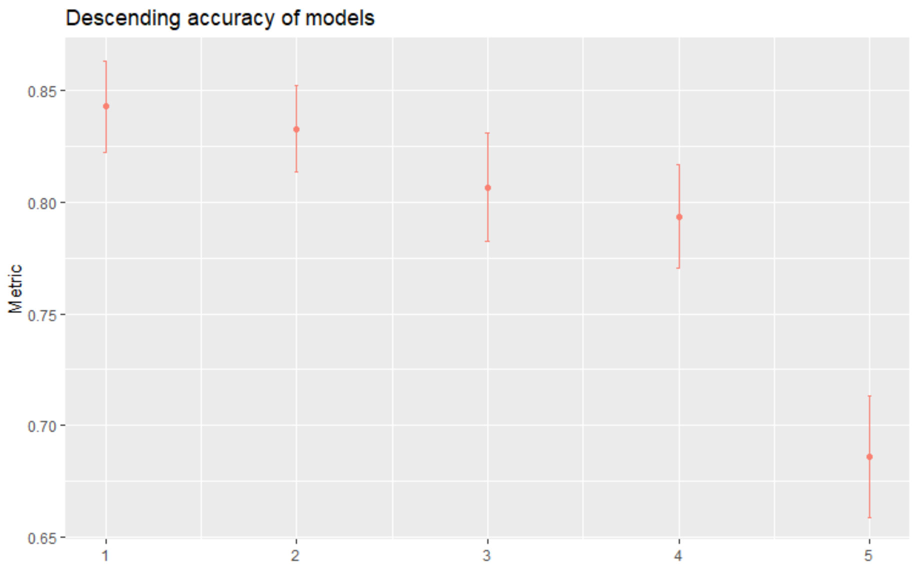

Figure 6. Plot of descending model accuracies. 1, 2, 3, 4, and 5 correspond to models c2, svm_poly_mu, lr_model1, xgb_1_classification, and stats101c_c1, respectively. Interestingly, the final model that produced the best results had the second-worst accuracy. The discrepancy can be attributed to testing the model on the selected 26% of the test data on Kaggle versus the accuracy of the model on the remaining 74%.

6 Final Model Discussion

Ultimately, we used the xgboost model with the least important predictors omitted. Despite its accuracy being one of the lowest in Table 2, we were aware of the possible variation in F-scores on Kaggle. Thus we concluded xgb_1_classification would predict the response well given our analysis in Section 2 and reasoning in Section 5. Our team's final model was chosen for a variety of reasons. One of the most important was that hyperparameters could be tuned alongside the internal regularization methods (both LASSO and ridge) that help prevent overfitting. The advantages of this model compared to the others included the aforementioned tuning of hyperparameters and removal of insignificant predictors at the expense of accuracy.

Some potential sources of error may stem from the lack of interaction effects and variable transformations in our final model. The model itself used no recipes, and there was very little in the way of preprocessing. Some variables, like age, thalach, and oldpeak, were heavily left-or right-skewed, and though we attempted to log-transform them, it did not normalize their distribution. Similarly, we did not center or standardize the predictors. Despite these limitations, however, our model is ultimately a very good predictor for the diagnosis of heart disease.

Appendix A: Final Annotated Script

```
# standard imports
library(tidyverse)
library(tidymodels)
# install.packages("doParallel") # if not installed already

# load in training data, turn columns from character into
# numeric, turn response into factor
heart_train <- read_csv("heart_train.csv")
heart_train <- heart_train %>%
  mutate_if(is.character, as.numeric) %>%
  suppressWarnings() %>%
  mutate(num = as.factor(num))
# remove id, trestbps, chol, fbs, and restecg columns
# call the reduced dataframe heart_train2
heart_train2 <- heart_train %>% select(-c(id, trestbps, chol,
fbs, restecg))

# load in testing data, turn columns from character into
# numeric
heart_test <- read_csv("heart_test.csv")
heart_test2 <- heart_test %>%
  mutate_if(is.character, as.numeric) %>%
  suppressWarnings()

# create initial xgboost model
xgb_model <- boost_tree(mode = "classification",
                         trees=tune(), mtry=tune())

# create recipe
xgb_rec <- recipe(num~., data=heart_train2)

# create cross validation folds
set.seed(100)
heart_train2_folds <- vfold_cv(heart_train2, 10)

# workflow for tuning model
xgb_wf <- workflow() %>%
  add_model(xgb_model) %>%
  add_recipe(xgb_rec)

# create parameter grid for tuning
param_grid <- grid_latin_hypercube(
  mtry(range(c(1,50))),
  trees(),
  size = 10
)

# fit models from grid
doParallel::registerDoParallel()
set.seed(100)
xgb_res <- xgb_wf %>%
  tune_grid(
     resamples = heart_train2_folds,
     grid = param_grid,
     control = control_grid(verbose = T, save_pred = T)
  )

# find model with best accuracy and use in workflow
best_xgb <- xgb_res %>% select_best("accuracy")
final_xgb_wf <- xgb_wf %>% finalize_workflow(best_xgb)

# fit model to training data and apply to testing data
set.seed(100)
xgb_1 <- fit(final_xgb_wf, data = heart_train2)
final_test_results <- heart_train2 %>% select(num) %>%
  bind_cols(predict(xgb_1, new_data = heart_train2))
pred <- predict(xgb_1, heart_test2)

# write predictions to csv
newpred <- data.frame(id = heart_test2$id, Predicted = pred)
names(newpred)[2] <- "Predicted"
write.csv(newpred, row.names = FALSE, file =
"xgb_1_classification.csv")
```

Appendix B: Team Member Contributions

Andra Velea

Initial final paper draft, all sections of paper, final model, intermittent models

Aryan Mistry

All sections of paper, intermittent models

Mateo Umaguing

All sections of paper, final script, graphics, intermittent models