# Creating a High Performing Supervised Learning Model to Predict Diagnosis of Heart Disease(Angiographic Disease Status) Using Statistical Classification Based on Clinical Data

| | |
|---|---|
| Kaggle Team Name | (Aplus) more like C- |
| Name (SID) | Kyungchae Baek (105548975)<br>Euijun Kim (705788156)<br>Sungwon Lee (405837554)<br>Christopher Apton (105373471) |
| Model | XGBoost |

## • Introduction: context and background info.

The goal of this paper is to create a high performing supervised learning classification model to predict the diagnosis of heart disease(angiographic disease status) using features based on clinical data. Some highly correlated features with heart disease include blood pressure, cholesterol, obesity, and diabetes (CDC).

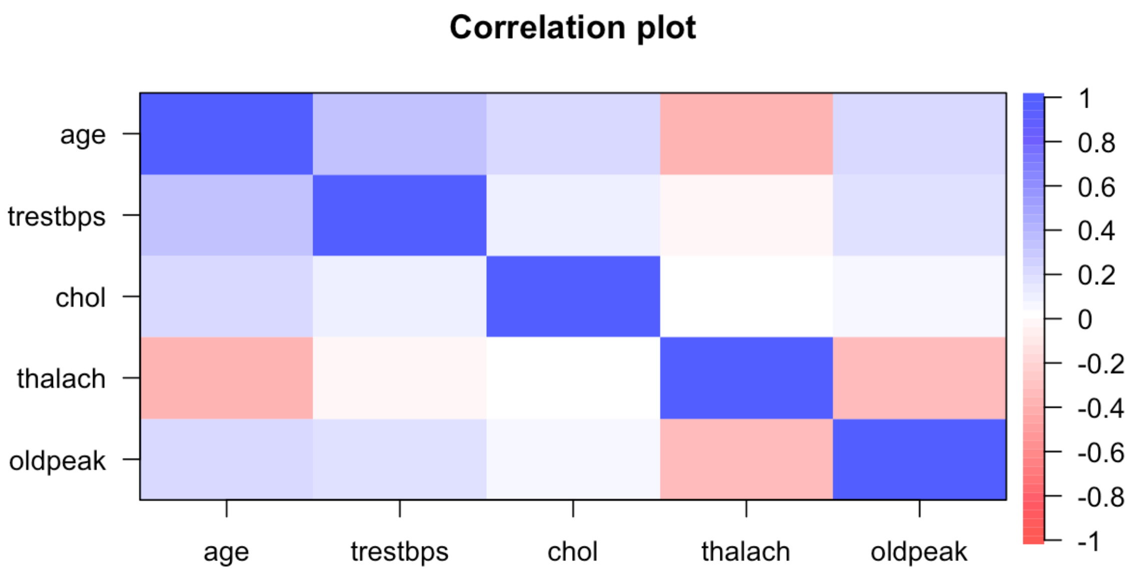
## • Exploratory Data Analysis

**The Response Variable: num: Response Variable you are predicting**

We have a total of 13 predictors, such as age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, short term depression induced by exercise relative to rest, the slope of the peak exercise short term segment, number of major vessels (0-3) colored by fluoroscopy, and thalassemia (thal). In this project, we have both numerical and categorical variables. First of all, we want to measure the relationship within or between variables. For the correlation between numeric variables and the response variable, the thalach variable has positive correlation, 0.4127409 with num. The boxplot below shows the correlation between thalach and the response variable is relatively stronger than the other numeric variables.

| | Age | Trestbps | Chol | Thalach | Oldpeak |
|---|---|---|---|---|---|
| Correlation | -0.2072177 | -0.135499 | -0.02657477 | 0.4127409 | -0.4069525 |

The correlation plot below indicates the correlation between numerical variables, which are age, trestbps, chol, thalach, and oldpeak. Thalach and age as well as thalach and oldpeak have relatively strong relationships.

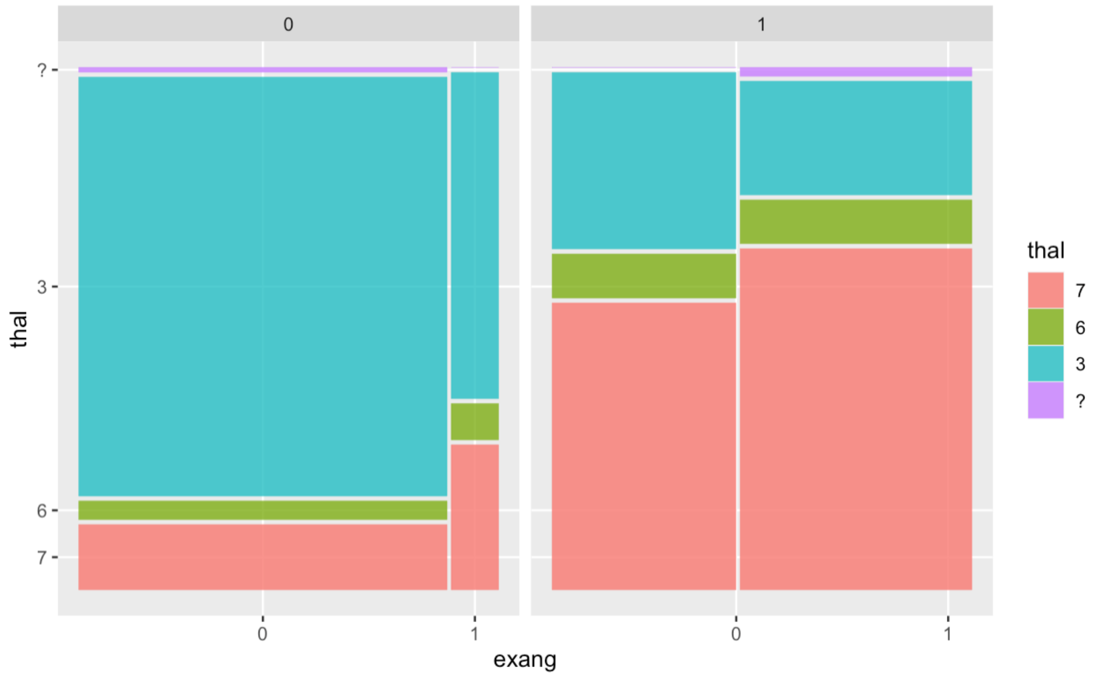


We have 8 categorical predictor variables, such as sex, cp, fbs, restecg, exang, slope, ca, and thal. The variable ‘thal’ is a categorical predictor with values 3, 6, and 7. These values represent normal, fixed defect, and reversable defect accordingly, which our team believed that these are not ordinal outcomes where a natural ordering exists (e.g. small, medium, large). Likewise, we also thought there was no intrinsic ordering for the numeric predictor ‘cp’ and ‘slope’. We made them categorical variables and split them into multiple columns for better performance. On the other hand, for the categorical predictor ‘ca’, we turned it into a numeric variable because it seemed to be ordinal outcomes where a natural ordering exists.

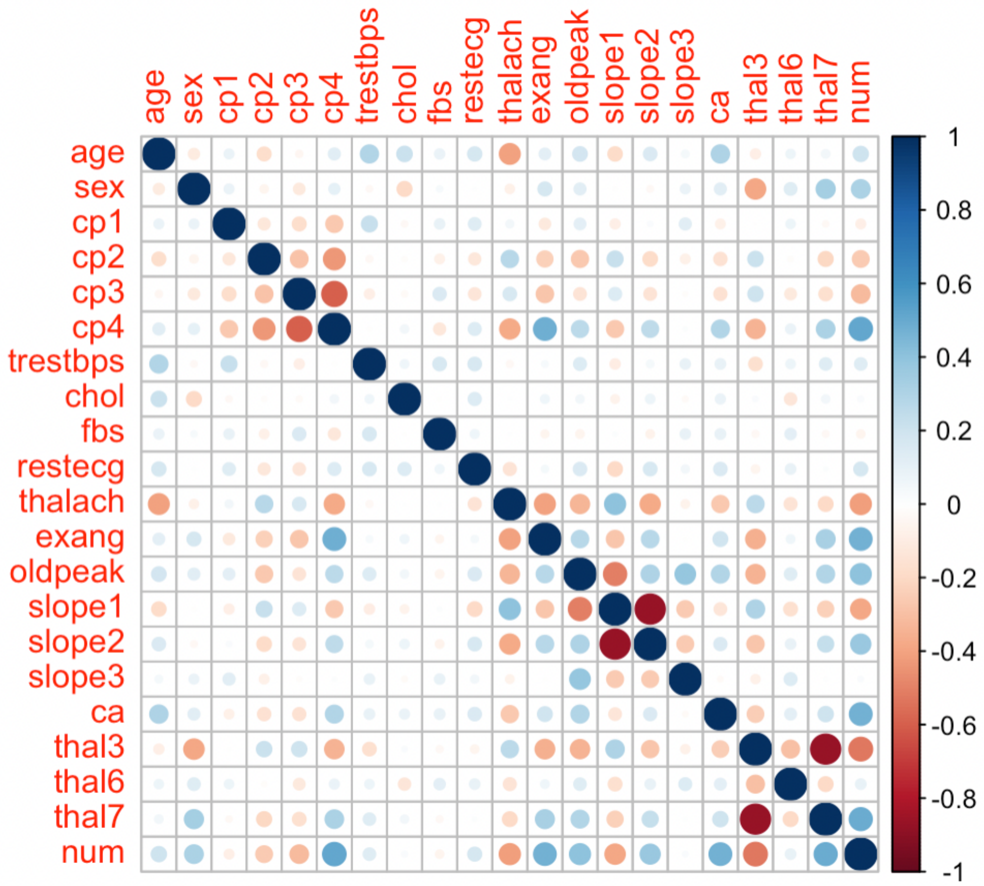
For the correlation between categorical variables and response variable, the thal, exang, and ca indicate that a significant and positive relationship exists between the two variables. In order to find a correlation between categorical variables, we are not able to use the pearson correlation coefficient because it is for the correlation between continuous numerical variables. In terms of our categorical data, I believe they are close to ordinal categorical variables, so I use polychoric correlation to calculate the correlation between ordinal categorical variables, and the function 'polychor(x, y)' (ZACH). The correlation between thal and num is 0.7330274 and between exang and num is 0.7062067. Those values indicate that there are a strong positive association between those variables.

| | sex | cp | fbs | restecg |
|---|---|---|---|---|
| Correlation | 0.5093051 | 0.5773208 | -0.1125437 | 0.2675348 |
| | exang | slope | ca | thal |
| Correlation | 0.7062067 | 0.4423677 | 0.6590545 | 0.7330274 |

From the correlation table above, I wanted to show "the distribution of multiple categorical variables involves visualizing counts and proportions"(University of Lowa). So in the plot below, I used the geom_mosaic function in R in order to visualize the joint distribution of exang and thal with the response variable involves visualizing counts and proportions by an area chart. I researched that "the resulting plots are often called mosaic plots" (University of Lowa).



The values 0 and 1 are the response variable num which is diagnosis of heart disease (angiographic disease status). Values of 0 mean < 50% diameter narrowing and values of 1 mean > 50% diameter narrowing. There is one missing row ‘?’ from thal variable, so factors end up as 3, 6, 7, and ?. It clearly shows that thal 3 and exang 0 are highly related with less than 50% diagnosis of heart disease, and that thal 7 and exang 1 are highly related with more than 50% diagnosis of heart disease.



After cleaning the data using one-hot encoding, most predictors have low correlations. The variable ‘thal’ is a categorical predictor with values 3, 6, and 7. These values represent normal, fixed defect, and reversable defect, which our team believed that these are not ordinal outcomes where a natural ordering exists (e.g. small, medium, large). Likewise, we also thought there was no intrinsic ordering for the numeric predictor ‘cp’ and ‘slope’. On the other hand, for the categorical predictor ‘ca’, we turned it into a numeric variable because it seemed to be ordinal outcomes where a natural ordering exists.

We can observe that num has strong correlation with sex, cp3, cp4, thalach, exang, oldpeak, slope1, slope2, ca, thal3, and thal7. Also, that exang has strong correlation with cp4. Cp3 has an inverse relationship with cp4. Overall, this correlation matrix helps us determine what variables are useful for the model and interaction effects that may occur. It appears that chol doesn’t have much impact and should be looked at to figure out why that may be.

**• Preprocessing / Recipes**

First of all, we wanted to convert the categorical variables in the data to multiple columns using one-hot encoding. This will increase the performance of most other models that we attempt. However, for the XGboost model, it shouldn’t change the performance too much from what I researched online. So, it wasn’t done on the final resulting XGboost model. When one-hot encoding was tested by other members in the group, it was confirmed that it did help performance based on the kaggle private score. Next, we found 1 row with missing values in the data on row 18, so we removed the row containing the NA value in the training data. Also, we had to convert the response variable to a factor so that it works with models. Later, we used a normal recipe in tidymodels to get our data ready for modeling. The normal recipe just contained the y variable we are looking for num and the other variables as the x variables which are named from the training data. The model we used had 1 outcome and 13 predictors for the XGboost model.

**• Candidate models**

We looked into multiple final candidate models after tuning, xboost engine with 500 trees and 0.1 learn rate, xboost engine with 200 trees and 0.1 learn rate, support vector machines, and k-nearest neighbors with k = 10. However, we didn’t have enough time to tune the

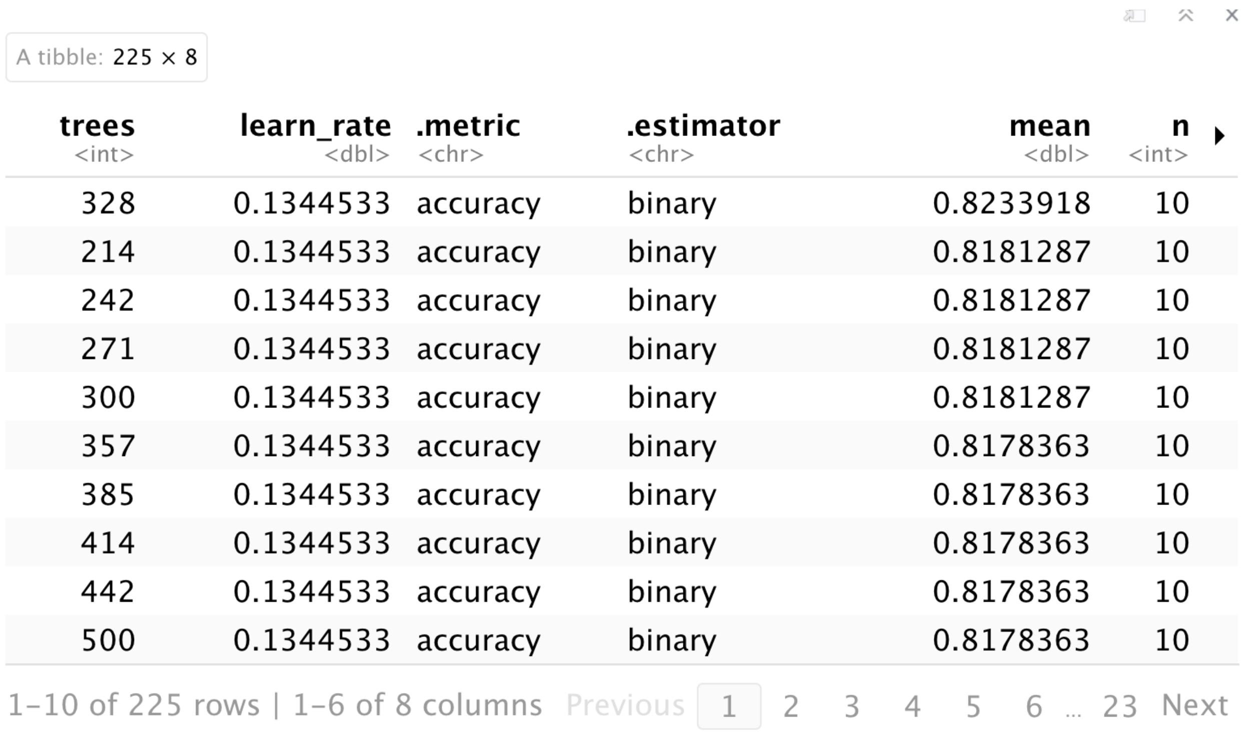
XGmodels and we ended up using the same 2 xgboost models as final contenders from the previous project.

1. boost_model: I was worried about it any other issues using so many trees when the default parameter was 15 and we were using 500. It took a lot longer to train and had lower accuracy compared to the model with 200 trees which is surprising to me. This model performed worse on the test data on kaggle as well compared to the model with 200 trees.
2. boost_model2: Interestingly, this model had a better accuracy in the test data compared to boost_model. I thought more trees would always increase accuracy, however, this case proves that theory wrong.
3. svm_model: This model had the worst accuracy of all the models. I think preforming tuning would have improved the performance of the model.
4. knn_spec: Uses k-nearest neighbors tuned with k = 10. This model had the best accuracy, however, I think it overfitted or got lucky since the public and private score on kaggle wasn't near as great as the boost models.

| | **Model Identifier** | **Type of Model** | **Engine** | **Recipe used** | **Hyperparameters** |
|---|---|---|---|---|---|
| 1 | boost_model | XGboost | xgboost | norm_recipe | trees = 500, learn_rate = 0.1 |
| 2 | boost_model2 | XGboost | xgboost | norm_recipe | trees = 200, learn_rate = 0.1 |
| 3 | svm_model | Support Vector Machine | kernlab | norm_recipe | |
| 4 | knn_spec | K-nearest neighbors | kknn | norm_recipe | k = 10 |

**• Model evaluation and tuning**

Tuning: I planned on tuning the hyper parameters for the xgboost model. However, they ran out of submissions so I'll show a hypothetical tuning for this classification model. The hyperparameters used for the XGboost models were the ones from the first report and ended up doing pretty well for the most part. I did tune the K-nearest neighbors model and found k = 10 to perform the best accuracy. I created a tuning grid with learning rates from 0.1 to 0.5 and also trees from 100 trees to 500 trees to see what the XGboost models should've been tuned to.



A tibble: 225 × 8

| trees<br>&lt;int&gt; | learn_rate<br>&lt;dbl&gt; | .metric<br>&lt;chr&gt; | .estimator<br>&lt;chr&gt; | mean<br>&lt;dbl&gt; | n<br>&lt;int&gt; |
|---|---|---|---|---|---|
| 328 | 0.1344533 | accuracy | binary | 0.8233918 | 10 |
| 214 | 0.1344533 | accuracy | binary | 0.8181287 | 10 |
| 242 | 0.1344533 | accuracy | binary | 0.8181287 | 10 |
| 271 | 0.1344533 | accuracy | binary | 0.8181287 | 10 |
| 300 | 0.1344533 | accuracy | binary | 0.8181287 | 10 |
| 357 | 0.1344533 | accuracy | binary | 0.8178363 | 10 |
| 385 | 0.1344533 | accuracy | binary | 0.8178363 | 10 |
| 414 | 0.1344533 | accuracy | binary | 0.8178363 | 10 |
| 442 | 0.1344533 | accuracy | binary | 0.8178363 | 10 |
| 500 | 0.1344533 | accuracy | binary | 0.8178363 | 10 |

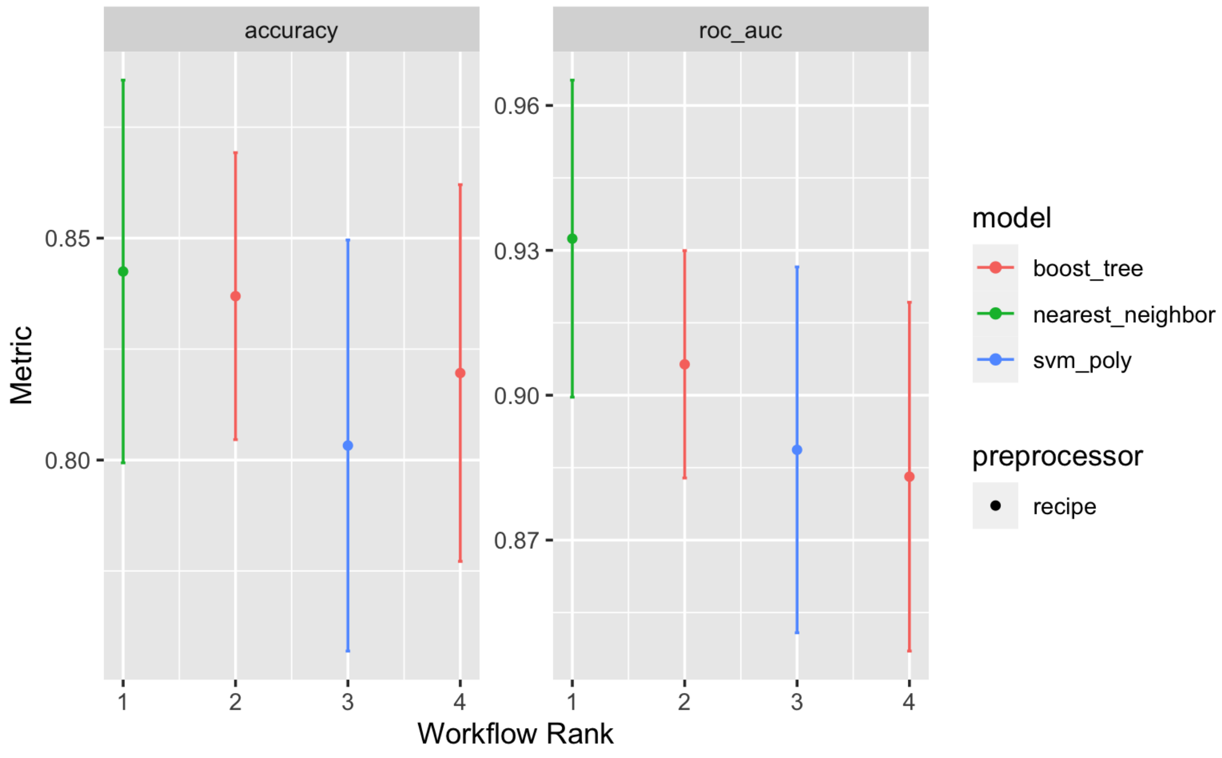
1–10 of 225 rows | 1–6 of 8 columns Previous 1 2 3 4 5 6 ... 23 Next

We can observe that between 200 and 300 trees that the results are pretty good with the lowest mean accuracy. The tuning grid used a learning rate of 0.13, however, I used 0.1 in the XGboost model. Overall, I think using trees = 200 and learn_rate = 0.1 was not too bad of a choice for our predictions. Next, we also trained K-nearest neighbors to obtain k = 10 as the best one.

Model Evaluation: We compared the models using the accuracy of the validation data and also using the public score on kaggle. Using v-fold cross validation we can observe in the code output below the mean accuracy of each model with 10 folds.

| | **Model Identifier** | **Metric Score** | **SE of Metric** |
|---|---|---|---|
| 1 | boost_model | 0.8196078 | 0.02580021 |
| 2 | boost_model2 | 0.8369281 | 0.01965288 |
| 3 | svm_model | 0.8032680 | 0.02815351 |
| 4 | knn_spec | 0.8424837 | 0.02621088 |

From these results, we can say the knn model with k = 10 performs the best over the xgboost model with 200 trees from boost_model2. We can also look at the plot below for a visual comparison of the 4 models.



It's hard to tell which XGboost model is which. So, from the data we can see that workflow rank 2 is boost_model2 and workflow rank 4 is boost_model. The reason KNN was not selected was due to having a lower public score on kaggle compare to boost_model2.

**• Discussion of final model**

The selection of the final model was a little difficult between the models. Having more trees did end up harming performance on the model, so now I'll be more careful of that. With our data, a model with 200 trees would be better compared to the 500 tree model. Even though KNN had the highest accuracy, it was hard to trust due to the lower public score. This decision to use the boost_model2 worked out since it led to a higher private score compared to the KNN model. I believe that using k = 10 led to overfitting of the data. Other team members attempted to use multiple models to get a higher public score. They used models such as lm_model, glm_model, and boost_model, then taking the average. However, the results didn't improve.

Strengths and weaknesses: Our model did alright compared to other models on kaggle. However, this was mostly due to luck and there are a lot of potential improvements. Another weakness of the model, was not using a stack/ensemble. I was planning on trying an ensemble with KNN and XGBoost. However, other group members used up all the submissions so I didn't have the attempts to try out that technique. One strength of the model was the reliability of the model. XGboost models are generally decent from what I researched and performed alright in the past competition compared to other models.

Some potential improvements would be to have more time to tune the model. Also, stacking the boost model with other models could such as logistic model or KNN to improve performance.

Another improvement would be to clean the data more and observe the relationships between variables closely to make sure we are only using variables we need when fitting the model.

**• Appendix: Final annotated script**

~~~
---
title: "Stats 101C - Kaggle Competition 2"
author: "Instructions"
date: "Summer 2022"
output:
  pdf_document: default
  html_document: default
---


```{r setup, include=FALSE}
library(knitr)
library(tidyverse)
library(tidymodels)
library(lubridate)
library(glmnet)
library(xgboost)
library(kernlab)
library(LiblineaR)
knitr::opts_chunk$set(echo = TRUE)
```
~~~

~~~
Data
```{r}
data <- read_csv("heart_train.csv")
head(data)
```


Cleaning Data
```{r}
id_data <- data[, "id"]
data <- select(data, !"id")
data[, "num"] <- factor(data$num)
data[, "ca"] <- as.numeric(unlist(data[, "ca"]))
data[, "thal"] <- as.numeric(unlist(data[, "thal"]))
head(data)
```


Splitting Data
```{r}
set.seed(502)
data_split <- initial_split(data, prop = 0.80, strata = num)
train <- training(data_split)
test <- testing(data_split)
```


Recipes
~~~

~~~
```{r}
norm_recipe <-
  recipe(num ~ . , data = train) %>%
  prep(training = train, retain = TRUE)
norm_recipe
```

Models
```{r}
boost_model2 <- boost_tree(mode = "classification", trees = 200,
learn_rate = 0.1) %>%
  set_engine("xgboost")

model_list = list(boost_model_2 = boost_model2)
```

Creating Workflow
```{r}
preproc = list(norm = norm_recipe)
glmnet_models <- workflow_set(preproc = preproc, models = model_list)
glmnet_models
```

Fitting resamples
```{r}
~~~

```
ames_folds <- vfold_cv(train, v = 10)
keep_pred <- control_resamples(save_pred = TRUE, save_workflow =
TRUE)

glmnet_models <-
  glmnet_models %>%
  workflow_map("fit_resamples",
               seed = 1101, verbose = TRUE,
               resamples = ames_folds, control = keep_pred)
```

Importing test data for kaggle

~~~
```{r}
kaggle_data <- read_csv("heart_test.csv")
head(kaggle_data)
```
~~~

Cleaning Data

~~~
```{r}
id_data <- kaggle_data[, "id"]
kaggle_data <- select(kaggle_data, !"id")
kaggle_data[, "ca"] <- as.numeric(unlist(kaggle_data[, "ca"]))
kaggle_data[, "thal"] <- as.numeric(unlist(kaggle_data[, "thal"]))

kaggle_data[9, "ca"] <- 0
~~~

```
kaggle_data[60, "ca"] <- 0

head(kaggle_data)
```
```

Fitting output model

```
```{r}
glmnet_wflow_2 <-
  workflow() %>%
  add_model(boost_model2) %>%
  add_recipe(norm_recipe)

glmnet_fit_2 <- fit(glmnet_wflow_2, train)
```
```

Output for kaggle

```
```{r}
kaggle_test_results_2 <- bind_cols(id_data, predict(glmnet_fit_2,
new_data = kaggle_data))

names(kaggle_test_results_2) <- c("Id", "Predicted")

write.csv(kaggle_test_results_2,"results_2.csv", row.names = FALSE)
kaggle_test_results_2
```
```
```

**• Appendix: Team member contributions**

Kyungchae Baek (105548975): Support other members and writing report

Euijun Kim: (705788156): Report, model, and visualization

Sungwon Lee (405837554): Script verification and supported others on writing the report

Christopher Apton (105373471): Model, report, and script verification

**• References**

CDC, “Know Your Risk for Heart Disease”, Accessed Jul 30, 2022, https://www.cdc.gov/heartdisease/risk_factors.htm

National Center for Health Statistics. Health, United States, 2017. Table 19: Leading Causes of Death and Numbers of Deaths, by Sex, Race, and Hispanic Origin: United States, 1980 and 2016 pdf icon[PDF-776M]. Hyattsville, MD: National Center for Health Statistics; 2018. Accessed Jul 30, 2022.

ZACH, “How to Calculate Correlation Between Categorical Variables”, Accessed Aug 2, 2022, https://www.statology.org/correlation-between-categorical-variables/

University of Lowa, "More on Categorical Data", Accessed Aug 2, 2022, https://homepage.divms.uiowa.edu/~luke/classes/STAT4580/morecat.html