University of California, Los Angeles

Predictive Analysis of Heart Disease

Tristan Dewing, Vivian Luk, Karina Santoso, Brandon Wang

Stats 101C

Lecture 1

Professor Almohalwas

December 12, 2021

Abstract

The goal of this Kaggle project is to predict the existence of heart disease using statistical learning models based on a given data set. This report provides a clear and detailed description regarding the process of how we developed our best classification model from start to end, and includes information about our exploratory data analysis, data cleaning, feature selection, model construction, results, limitations and conclusion.

Our best model is based on logistic regression and uses all predictors: 7 numerical predictors and 12 categorical predictors. It has a Kaggle score of 0.80973, and our final rank in Lecture 1 is 6th.

1. Introduction

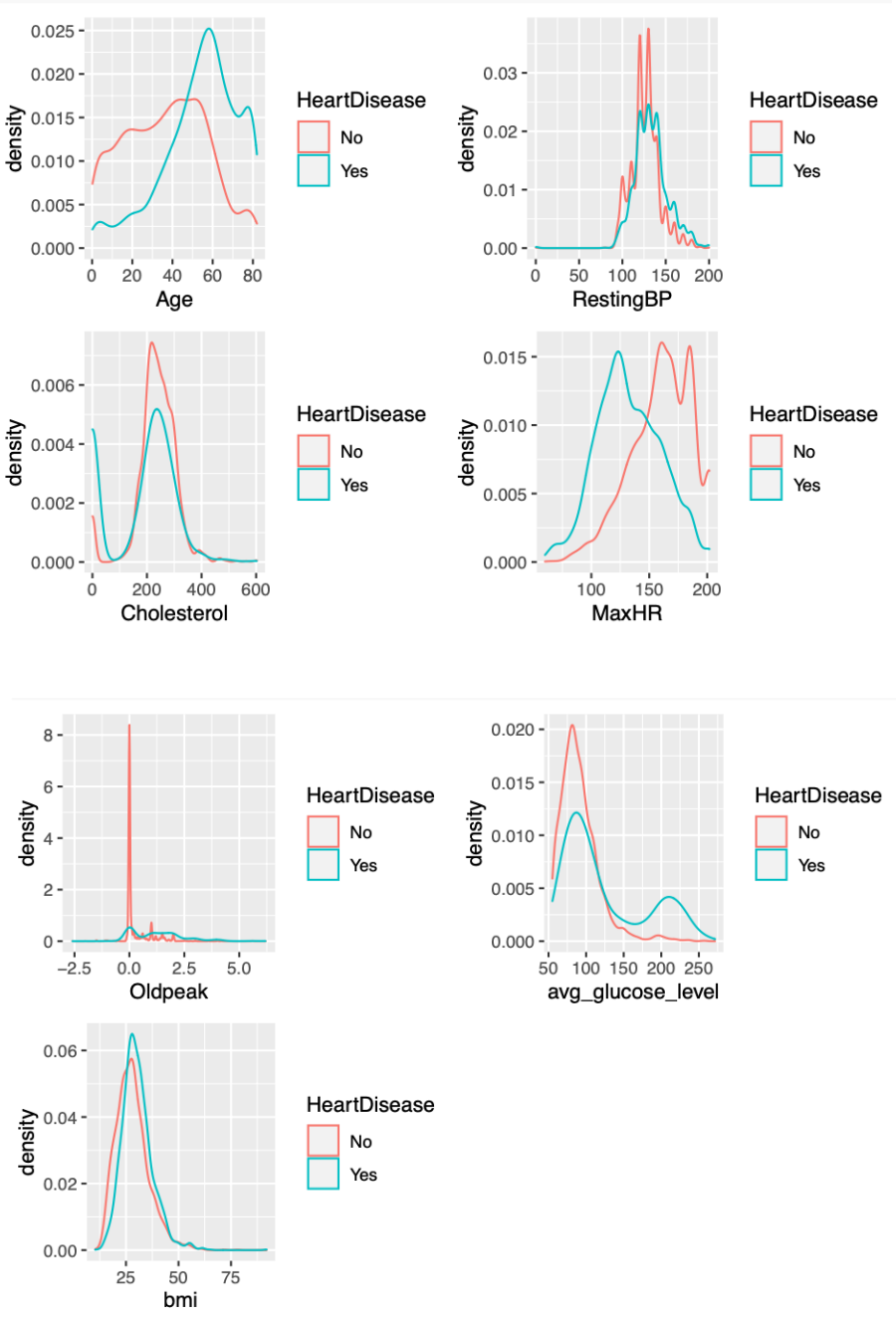
An overarching term encompassing all different types of heart conditions, heart disease is the top cause of death in the United States, the most common of which being coronary artery disease, which causes heart attacks. As heart disease causes one in four deaths in the United States, it is important for all to be aware of the behaviors and symptoms that tend to decrease and increase one's risk of heart disease. Some of the factors that increase one's risk for heart disease can be controlled to decrease risk, the largest three being high blood pressure, high cholesterol, and smoking. Thus, a patient at high risk of heart disease can make lifestyle changes to reduce their risk of heart disease, such as by eating a healthier diet, getting enough exercise, and lowering levels of alcohol and tobacco consumption (CDC). Being able to identify patients at high risk of heart disease is essential in the efficacy of these preventative measures and could potentially save numerous lives each year.

For our Kaggle project, we were provided with a training dataset with 20 different variables and 4220 rows, where each row is an observation or patient. The testing dataset has 19 columns and 1808 rows, the column missing being the column that denotes whether or not a patient has heart disease or not. Excluding this response variable, the other 19 variables in the dataset are made up of 7 numerical variables and 12 categorical variables. These variables describe the demographic characteristics of each patient as well as some information about their health history and current vitals. Living habits such as what type of work an individual does and whether or not they smoke are also some pieces of information given in the dataset. The goal of this project was to use these 19 predictor variables to predict whether or not an observation has or will have heart disease as accurately and efficiently as possible.

2. Data Analysis

Prior to beginning model selection and training, we conducted an exploratory data analysis on the different predictor variables in an attempt to analyze our dataset and determine which variables may be the most effective in predicting heart disease. In addition, this allowed us to visualize any significant interesting patterns or trends in the variables.

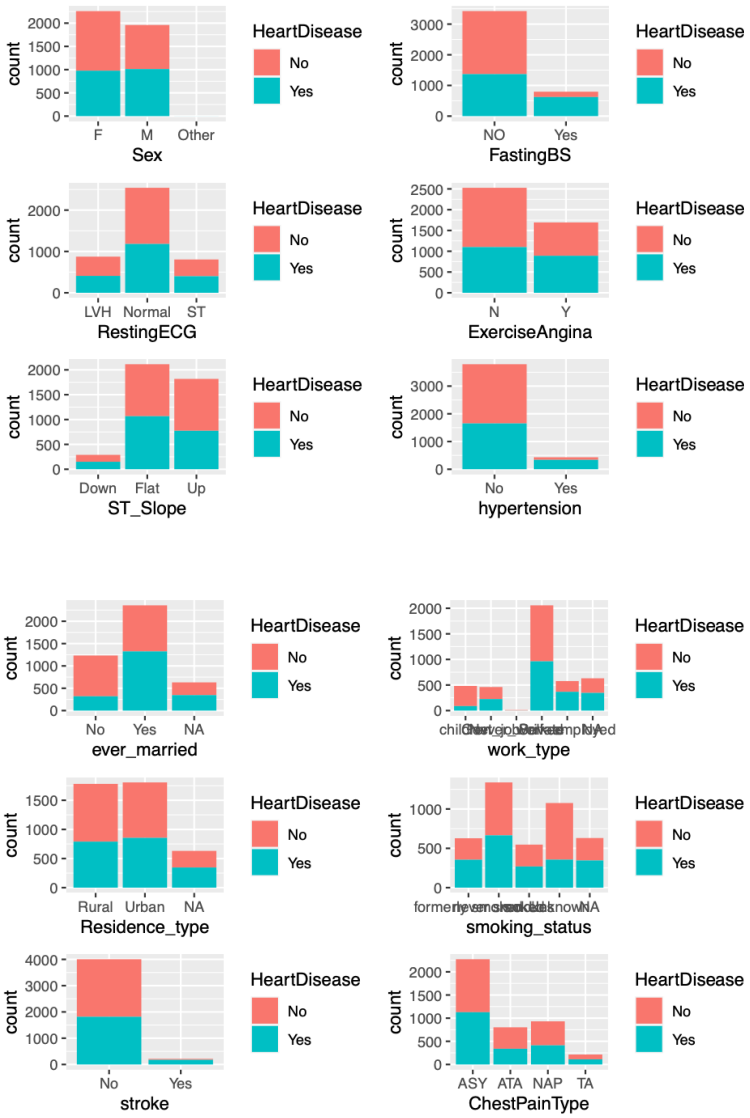
To visualize whether there were differences in the distribution of numerical variables based on whether or not a patient had heart disease, we utilized density plots for our 7 numerical predictor variables:



As we can see from the above, some of the variables' distributions seem to be more significantly different than others between observations that do or do not have heart disease. For example, BMI does not seem like it will be too useful of a predictor variable, while age on the other hand seems like a potentially significant predictor. We further analyze these density plots in

more depth in the following Feature Selection section of this paper. There were no NA values found in these 7 columns.

To visualize whether there were differences in the distribution of categorical variables based on whether or not a patient had heart disease, we utilized bar plots plots for our 7 numerical predictor variables:



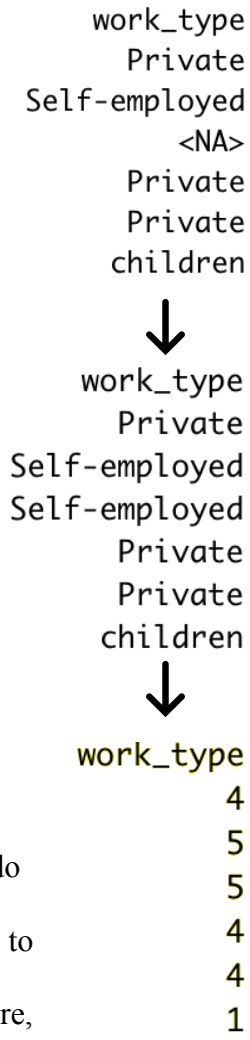
Again, from the above, the proportions of observations that do or do not have heart disease seem more statistically significantly different for some variables than others. For

example, we see that observations who have had a stroke in the past have a higher rate of heart disease, indicating that stroke is a potentially useful predictor variable, while on the other hand we see that the proportion of people with heart disease remains pretty constant despite residence type, indicating that it is likely not as useful of a predictor variable. These bar plots will also be analyzed in more depth in the following Feature Selection section of this paper. Four of these variables were found to have what seemed to be a significant number of NA values.

B. Data Cleaning

In order for the models to train properly, we had to use two techniques to modify the dataset: imputing NA values and label encoding.

With 2524 NA's in the training data and 1148 NA's in the test data, we had to either drop or impute the missing values to ensure our models could successfully run. We tried various methods, including dropping all columns with NA's, as only 4 predictor variables contained missing values, as well as imputing all NA's for numerical predictors with the mean or median of the column and even using the mice package. Ultimately, the most effective method was imputing all NA's with the non-NA value that was closest to them in their respective column, similar to how the na.locf function from the zoo package imputes NA's (Beck).



Some models we trained need all predictors to be numerical and thus do not accept categorical variables as strings. This requires categorical predictors to be encoded as integers so they can be treated as numerical predictors. Therefore, we encoded all 12 categorical predictor variables as integers so that our models could properly

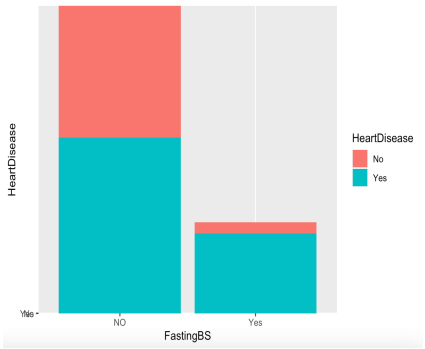
train and run. For instance, in the Sex column, “Female”, “Male”, and “Other” were converted into 1, 2, and 3, respectively.

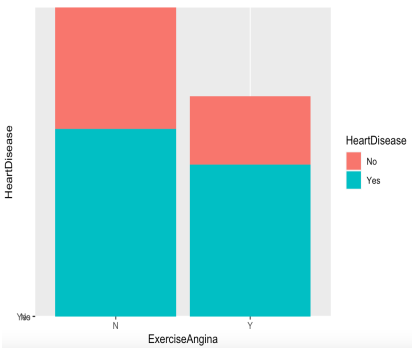
One other consideration we made when cleaning the data was whether or not the values in each column were recorded properly and consistently. In the Sex column, while the main categories were ”Female”, “Male”, and “Other”, there were some values that were recorded as “F” and “M” instead of female and male, which would lead to 5 categories being represented after encoding the variable even though there are only 3 categories. Yet, even after replacing “F” and “M” with “Female” and “Male”, none of the models improved significantly, so the change in the labels did not make much of a difference.

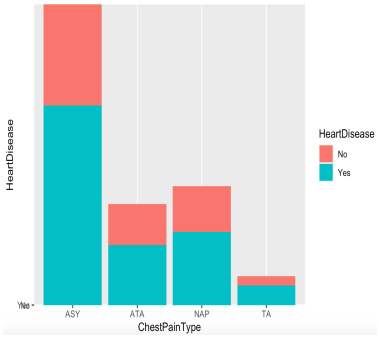
C. Variable Selection

After performing EDA and cleaning the missing values, we started our feature selection process which helped us dig deeper into the data and choose predictors for our models that might improve the classification accuracy rate. Based on the combination of bar plots and density plots, we determined that overall, categorical variables are generally better than numerical variables at separating the categories of the response variable.

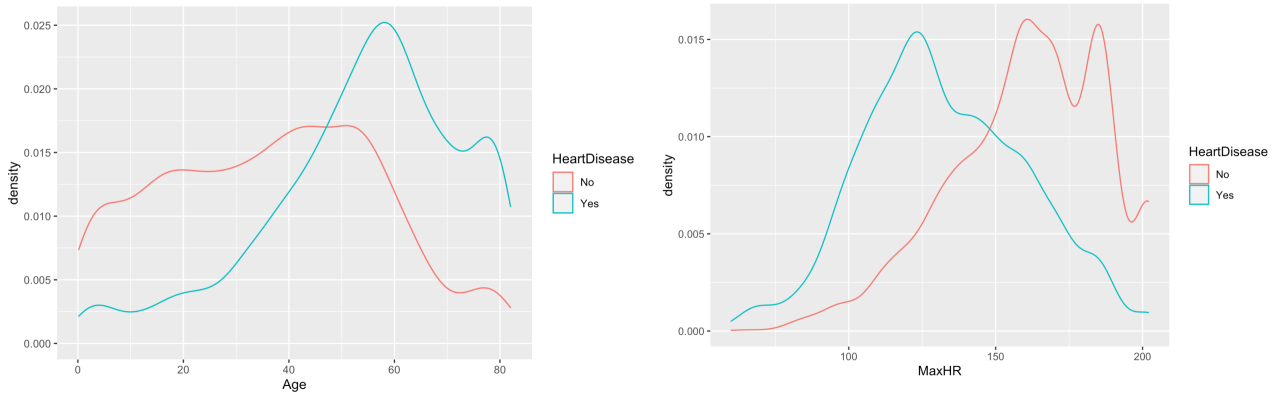
For the categorical variables selection, we used stacked bar plots to determine if the predictor was useful in separating the categories of the response variable. From the bar plots, we determined Fasting Blood Sugar, Exercise Angina, and Chest Pain Type were the best categorical predictors.



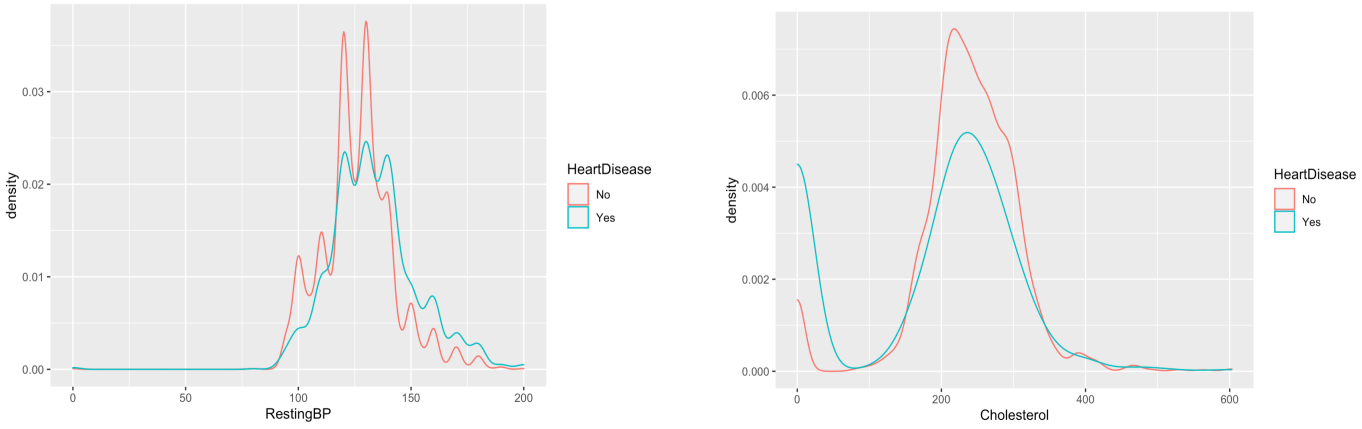




Since our response variable diagnosis is a binary categorical variable, we determined which quantitative variables we wanted to include in our model based on density plots. We plotted two density curves of each predictor for the two categories of heart disease existence. If the density plot shows distinct differences in distribution, we can conclude that this predictor is potentially significant and keep them. Otherwise, we discard predictors that have density plots that have a lot of overlap.



From the above graphs, we can see that there are significant differences between the two density curves, indicating for Age, MaxHR, Oldpeak and avg_glucose_level, the probabilities of existence of heart disease are different, which indicates that these variables are helpful and we should keep them in our model.



In contrast, we can see that density curves above have a lot of overlap, pointing out that for RestingBP, Cholesterol, and bmi, the probabilities of the existence of heart disease are almost the same, so we should reconsider using these predictors in our models.

Finally, we conclude that Age, MaxHR, Oldpeak, and avg_glucose_level are potentially significant and we should use them in our classification model.

3. Methods & Models

After cleaning our data and selecting our variables, we trained an assortment of classification models on the training data, including logistic regression, KNN, SVM, and XGBoost. To evaluate our models, we created confusion matrices and computed correct classification rates for our training data. Models that achieved at least 80% training accuracy were submitted to Kaggle where they were scored for their testing accuracy. The hyperparameters and number of predictors used in each model were continually adjusted using grid search or k-fold cross validation to optimize their performance.

A. Logistic Regression

The first and perhaps most simple model we trained was logistic regression, which calculates class probabilities of a binary response variable using the logistic function. Logistic regression was a natural fit for predicting heart disease because of its specialty in classifying binary target variables. Because the model doesn't require any hyperparameters, the variation between different runs of the model came from the number of predictors we used. The best-performing logistic regression model used all predictors and achieved a training accuracy of 0.8135071 and a testing accuracy of 0.808073.

B. KNN

The next model we tried on the dataset was a KNN, or K-Nearest Neighbors model. KNN classifies new observations based on distance from other observations with known classes. The hyperparameter, k, of the model, determines how many of the new observation's closest neighbors should be taken into account when classifying it, taking the majority vote of these k observations to assign a class to the new observation.

To determine what the optimal value of k is for our dataset and goals, we implemented multiple KNN models, one each with k being all of the odd numbers spanning between 1 and 31. The training dataset was split into a 70-30 split in order to create a mock training set that the model hadn't seen, then the training and 'testing' accuracy were calculated for each of the 16 different possible values of k. The value of k that ended up producing the highest testing accuracy was k = 25, so this is the value of k we used for our KNN model.

This model had a training accuracy of 0.8028436 and a testing accuracy of 0.785399. We see that the model performs well on the training data but that we see a drop in accuracy when we take in the testing data, so the model does not seem to perform as well on new observations. The KNN model is known to fall victim to the problem of overfitting, so this is likely the reason why we see these results (Zhang).

C. SVM

We also tried a support vector machine (SVM) classifier, which finds the decision boundary that best separates classes of the response variable, usually in the form of a hyperplane

that exists in a higher-dimensional space. SVM requires more hyperparameters than logistic regression and KNN and therefore warrants use of grid search to determine the optimal model.

The three main hyperparameters that were tuned in the model were “kernel”, which specifies the function used to determine the similarity of observations and form the decision boundary, “gamma”, a parameter in the non-linear kernel functions, and “cost”, which serves as a regularizing penalty for the cost of misclassification (Karatzoglou, et. al.). Increasing gamma and cost generally led to very high training accuracies above 85% or even up to 100%. However, this always resulted in large disparities between the training accuracy and testing accuracy, indicating overfitting. As a result, SVM was one of the trickier models to train, as the training accuracy was not a reliable indicator of its testing accuracy.

The best-performing SVM model consisted of all predictors and the default hyperparameters which included an RBF (radial basis function) kernel, a gamma value of 1/n where n is the number of observations, and a cost of 1, altogether achieving a training accuracy of 0.8421801 and a testing accuracy of 0.799227. Even though this was the best SVM model, it still slightly overfit the training data.

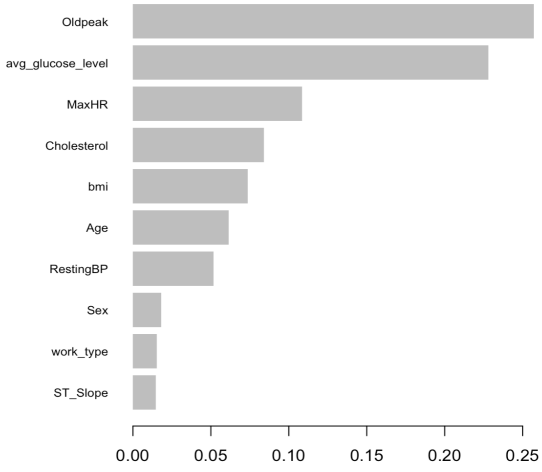
D. XGBoost

XGBoost stands for Extreme Gradient Boosting and is a tree-based ensemble learning method that is applicable for both classification and regression. The method builds gradient-boosted trees one at a time, which allows for new trees to use the results of old trees to classify observations.

For this project, we created two XGBoost models to use for our Kaggle submissions. The first XGBoost model used all of the predictors. For the parameters, we set the max depth as

1000, eta as 0.3, nthread as 2, nrounds as 25, and objective as “binary:logistic”. The maximum depth of a 1000 meant that the model worked well on the training data and resulted in a low classification error rate for the training data. However, since the maximum depth was so high we ran the risk of overfitting. In contrast, the eta of 0.3, which controls step size shrinkage, is relatively conservative. The objective of “binary:logistic” worked the best, which makes sense since the classification is binary and logistic regression yielded our best model. The XGBoost model using all predictors with the above parameters achieved an overall Kaggle accuracy of 0.79964.

After building an XGBoost model with all the predictors, we used feature selection to create a second model. We then created an importance matrix to determine which variables were the most important. The plot of the importance matrix, shown below, demonstrates that Oldpeak, avg_glucose_level, MaxHR, Cholesterol, bmi, Age, and RestingBP are the seven most important predictors. These are the predictors we used in our second XGBoost model. For this model, we set the max depth as 7, the eta as 0.1, nthread as 2, nrounds as 15, and objective as “binary:logistic”. With this model, in comparison to the first XGBoost model, we have a more conservative max depth and eta value, which helps to mitigate the risk of overfitting. However, unfortunately, the more conservartive parameters and feature selection did not improve our classification accuracy. For our XGBoost model with feature selection, we had an overall Kaggle accuracy of 0.785977, which is lower than that of our XGBoost model with all predictors.



5. Discussion and Limitation

The best performing model out of the four was the logistic regression using all predictors and was our final model in the Kaggle competition.

Our group did well overall in the competition, consistently staying in the top three in the public leaderboard, but we dropped to sixth place after the private leaderboard results were unveiled. We chose our best two models on the public leaderboard to use for our final submission, but in hindsight, we could have chosen models that overfit slightly less and ended up performing better when the private results were revealed.

Even if we got first place, however, there would still be intrinsic limitations to every modeling technique. Logistic regression, for example, assumes linearity between the predictor variables and the logit of the response variables. We went along with this assumption, even though it was invalid for certain variables like chest pain type, because our feature-selected models did not outperform our models that used all predictors.

KNN has the limitation of only being able to take numerical predictors into account. Even though they weren't the most significant predictors in our density plots, certain categorical variables like chest pain type and smoking status would still likely be helpful in predicting heart disease, so it makes sense that KNN ended up performing relatively poorly on our testing data. KNN does tend to perform better on smaller datasets with scaled features, so we may have been able to improve our performance with more fine-tuning, but because the Kaggle competition is purely based on model accuracy, we decided our time would be better spent on classification methods that could make better use of all predictors.

Support vector machine classification was the point where our models started to become harder to interpret. Because our dataset had many dimensions, the resulting decision boundary

was complex, and the kernel function acted like a black box. Unlike logistic regression, there is no probabilistic explanation for SVM classification.

Even more nebulous than support vector machines is XGBoost—extreme is in the name for a reason. The complexity of the algorithm makes it the most difficult one to break down in terms of variable significance. Because it is a boosting method, it also has a very high tendency to overfit (Davagdorj et al.), as evidenced by its extremely strong testing classification rate of 0.938 but average testing classification rate of 0.799 in our final model.

Lastly, one of the most significant challenges presented by our dataset was the presence of many NA values. Funnily enough, similarly to how our highest-performing model was the simplest, our highest-performing imputation method was also one of the simplest: merely imputing NA values with the closest non-NA value in their column. There are still countless imputation methods we have yet to try, however, and further exploration of this topic is warranted.

6. Conclusion

By undertaking this project, we learned that some datasets make it hard to outperform some of the simplest models. Our most effective method was logistic regression, and even the first place team only outperformed our best model by less than one percentage point. If we could revise our model knowing what we know now, we would probably start with exploring more imputation methods. It was slightly disappointing that nobody could achieve higher than 81 percent accuracy, meaning false positives and false negatives would make up nearly 20 percent of cases if any of our models were used as part of a diagnostic test. Still, medicine remains an exciting use case of data science, and it's possible that our models could be the start of something

useful as, for example, a way for doctors to get a second opinion on their patients' health. In the end, we ended up learning just as much as our models did: that statistics can help solve the world's most pressing problems, that real-world datasets are very messy, that actually presenting data in a precise and effective manner might be even more important than being able to build the absolute best model, and that we should always know who our daddy is.

References

Almohalwas, Akram. “Introduction to Statistical Models and Data Mining.” Statistics 101C, Fall Quarter 2021, UCLA, Los Angeles. PowerPoint presentation.

Beck, Marcus W, et al. “R Package Impute Testbench to Compare Imputation Methods for Univariate Time Series.” *The R Journal*, vol. 10, no. 1, 4 Nov. 2016, p. 218., https://doi.org/10.32614/rj-2018-024.

Davagdorj, Khishigsuren et al. “XGBoost-Based Framework for Smoking-Induced Noncommunicable Disease Prediction.” *International journal of environmental research and public health* vol. 17,18 6513. 7 Sep. 2020, doi:10.3390/ijerph17186513

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.

Karatzoglou, A., et al. Support Vector Machines In R. *Journal of Statistical Software*, *15*(9). April 2006, https://doi.org/10.18637/jss.v015.i09

"Know Your Risk for Heart Disease." Centers for Disease Control and Prevention, 9 Dec 2019, https://www.cdc.gov/heartdisease/risk_factors.htm.

S. Zhang, "Challenges in KNN Classification" in IEEE Transactions on Knowledge & Data Engineering, vol. , no. 01, pp. 1-1, 5555. doi: 10.1109/TKDE.2021.3049250