Capstone 2 Report

I am using medical data from patients to make a prediction as to if a patient is at a high risk for heart disease or not. Some common health features are used such as heart rate, cholesterol, age, gender, diet and many other factors that are often associated with heart attacks. A big issue in medical diagnostics is that people don't get screened for medical problems such as heart disease. The goal of my algorithm is to allow patients to know if they are at a high risk for heart attacks and hopefully drive people to get screened who may have otherwise not gotten screened. My approach was to start with some of the factors that are generally thought to be associated with heart attacks such as diet, age, and previous heart attacks. I used count plots to measure how the presence of the target feature affects the risk of heart disease. After this, I wanted to see which features correlated with each other and created other visualizations to see what affects those features made on heart attack risk. To my surprise, the data suggested that no one individual feature contributed much to heart attack risk. None of the features seem highly correlated with one another which may be a good thing. Perhaps the combination of different features will serve as a better predictor for heart attack risk.

In order to create more features, I split blood pressure into diastolic and systolic. I also made the diet feature into a categorical ordinal type ranging from 0 - 3. Lastly, I created dummy variables for the sex feature. Doing feature engineering allowed me to pass in more numeric data to my models and to give it more features which should hopefully improve model performance. After some testing of base models and even doing several grid searches on models such as Random Forest, XGBoost and more, I was not getting very accurate predictions. My first approach was to simply sample an even amount of my target feature of heart attack risk, but I had to drop quite a bit of data in order to do this. My predictions didn't see much change from this resampling so I tried another approach. My dataset is a bit unbalanced, so therefore I decided to use the SMOTE library which helps to create synthetic data and rebalance an unbalanced dataset. This proved to be very useful and I was finally seeing some reasonable performance for base models.

My two best performing models were Random Forest and XGBoost. The main issue I was running into was that Random Forest was the overall best model but the XGBoost model consistently had the best recall score for positive values which is what I want to optimize for in this particular problem. After model tuning, the Random Forest still performed best overall but still had a low recall score for positives. I would like to perform further research and experiment with ensemble models, where certain values that the random forest does not predict well would be passed into another model. I would also further investigate with more resampling strategies to see if the data could be presented in a way that makes predictions more reliable.

I would recommend using the Random Forest model because it does have a decent precision of 75% of the positive cases. This means that 75% of the patients who are flagged for having heart attack risk will end up actually having a risk and the other 25% wont. The weakness of the model is that it will miss a large amount of the positive cases. One strategy could be to run all of the cases that were not flagged in the Random Forest through the XGBoost model because the

XGB model has a higher recall for positive cases, and would likely pick up some of the positive cases that were not detected by the first model. Another option would be to just use the XGBoost model since it has the best recall for positive outcomes. The XGBoost will classify some patients who were not at risk as being at risk, but it will capture a larger percentage of the total positive cases which is ultimately what we want. My last recommendation would be the reverse of the first recommendation and to use the XGBoost model as a prescreener, and then pass on patients who were not flagged into the Random Forest model. The Random Forest model is best at predicting negative cases, so it should mostly catch on to positives that were not picked up by the XG model. Doing so will prevent too many false positives and still save time overall.