

Predicting the Ten Year Risk of Future Heart Disease

Andrew Mashhadi, Ajay Patel

Introduction

Every year, cardiovascular disease accounts for nearly 18 million lives and more than 4 out of 5 cardiovascular deaths are due to heart attacks and strokes (WHO). Reducing cardiovascular deaths starts with identifying the disease early. Demographics, medical history, genetics, and behavioral risks can help detect an early prognosis. With the abundance of medical data collection, a natural question arises: Would combining these factors with machine learning models help aid early detection? In this project, we will investigate various binary classifiers and the factors most pertinent to predicting the 10-year risk of future heart disease. Note that this project was also intended to improve upon the methodology of a prior presentation given in *Stats 402* using new methods and techniques introduced in this course, *Stats 412*.

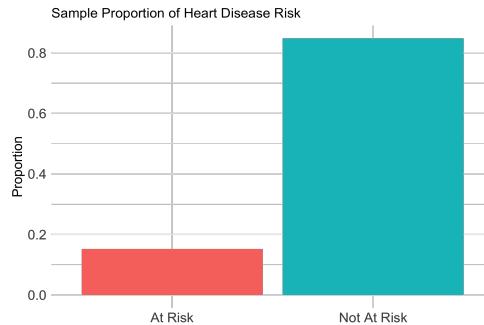
Data

Our dataset is an ongoing cardiovascular study on the residents from Framingham, Massachusetts, and the data is publicly available on Kaggle. We have nearly 4000 observations and 15 different predictor variables describing patients demographics, behaviors, medical history, and current medical measurements.

Exploratory Data Analysis

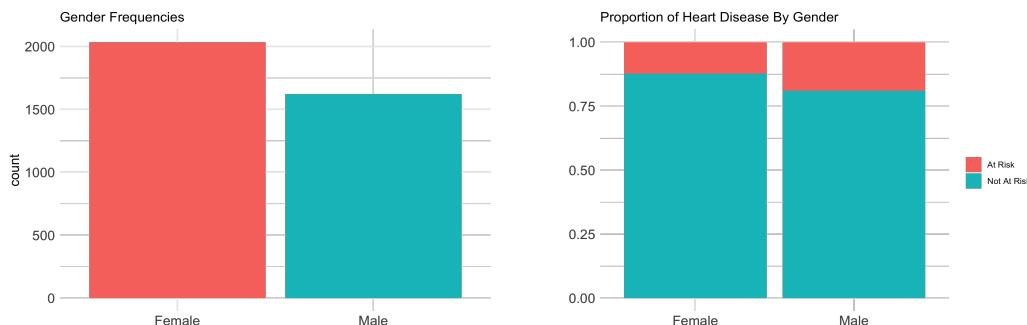
Ten Year Risk of Coronary Heart Disease

According to the proportions plot of the response variable below, just over 80% of the patients in our study do not have a 10-year risk of coronary heart disease.



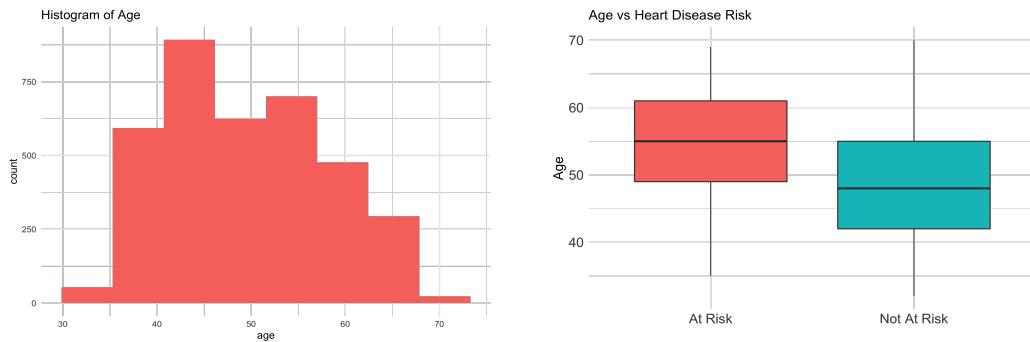
Sex

In our study, 55% of the patients are female and 45% are male. According to the two-way proportions plot below, we see that males are slightly more likely to have heart disease compared to the females in this study.



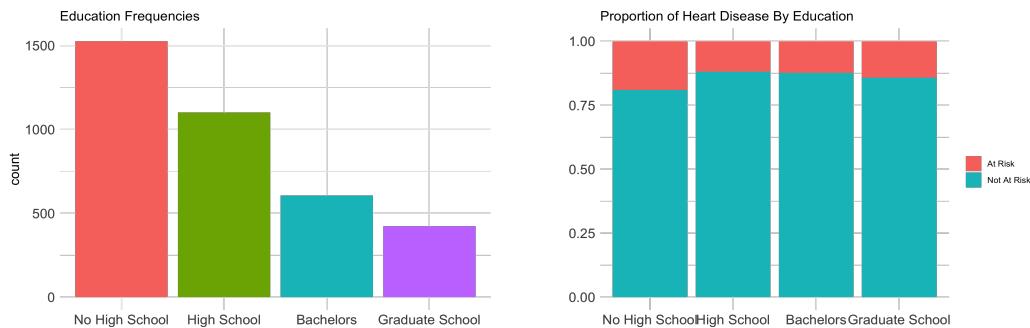
Age

The most common age groups in the data are patients in their 40's and 50's. We can see that older patients' are generally associated with more heart disease risk.



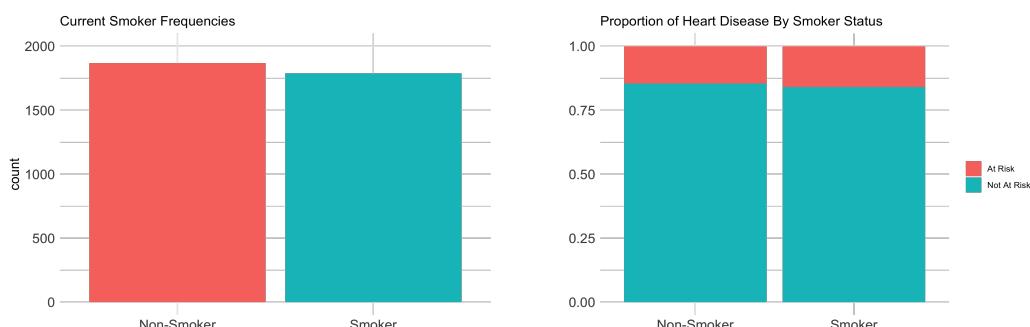
Education

According to the plot below, nearly half of the patients do not have a high school education while one-third of patients have a bachelor's degree or a graduate school degree. The risk of 10-year coronary heart disease among patients with at least a high school education appears less than the proportion of heart disease among patients with less than a high school education.



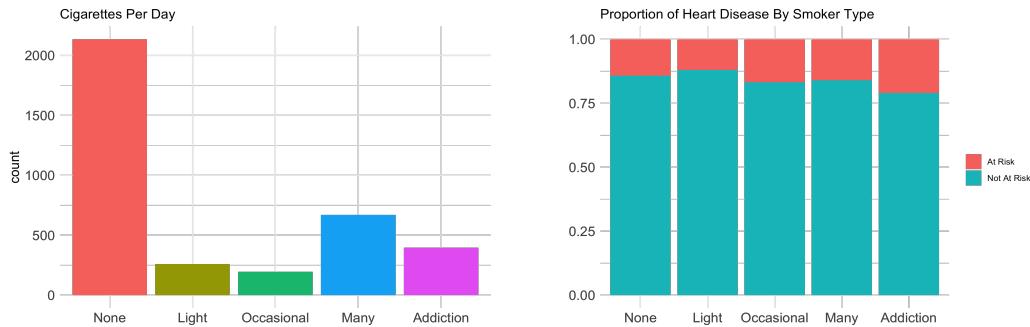
Smoker Status

In our dataset, the number of smokers is nearly equivalent to non-smokers. Surprisingly, the risk of 10-year coronary heart disease across both groups is approximately equal. Typically, we would expect smokers to have a slightly larger risk of 10-year coronary heart disease.



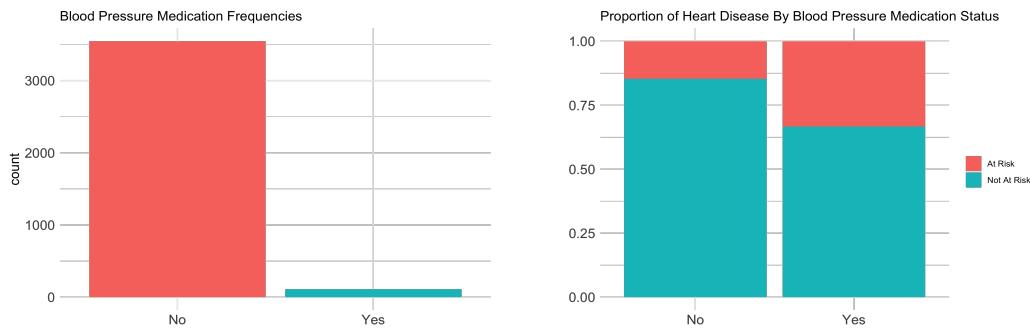
Cigarettes Smoked Per Day

As mentioned above, nearly half our dataset consists of non-smokers. But, for those who do smoke, we see that most patients either smoke many cigarettes per day or have a cigarette addiction. As the smoker type increases from light to addiction, we see the risk of 10-year coronary heart disease increases as expected.



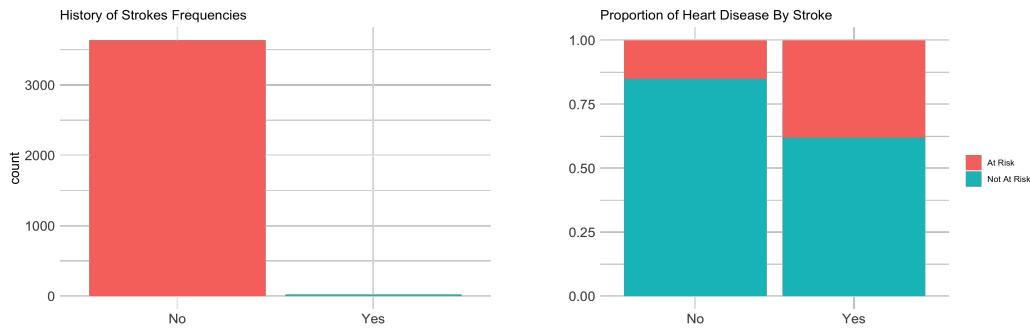
Blood Pressure Medication

The majority of patients in the study are not on blood pressure medication. For patients on blood pressure medication, over 25% of patients are at risk for 10-year coronary heart disease.



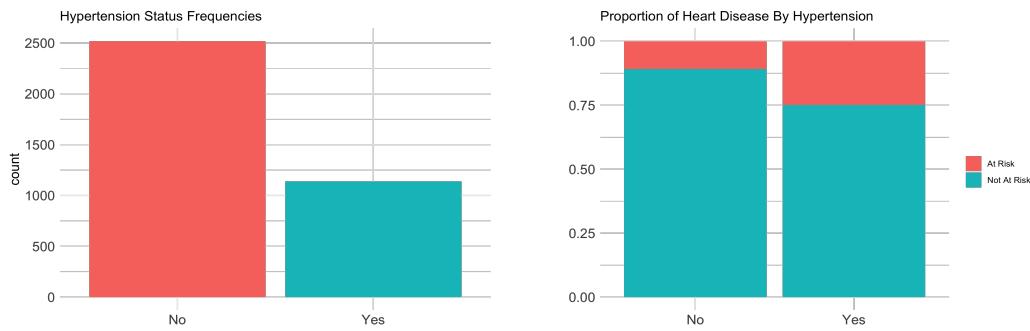
Prevalent Stroke

Similarly to blood pressure medication, the majority of patients do not have a history of stroke. As expected, the proportion of patients at risk for 10-year coronary heart disease is greater among patients who have had a stroke. In fact, if a patient in our study has had a stroke, they are twice as likely to be at risk for 10-year coronary heart disease compared to a patient who has not had a stroke.



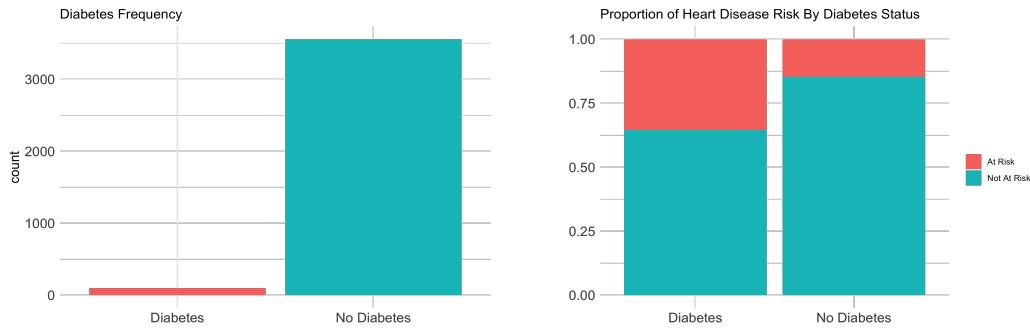
Prevalent Hypertension

Nearly one-third of patients in the study do not have hypertension. Similarly to prevalent strokes, if a patient in our study has hypertension, they are twice as likely to be at risk for 10-year coronary heart disease.



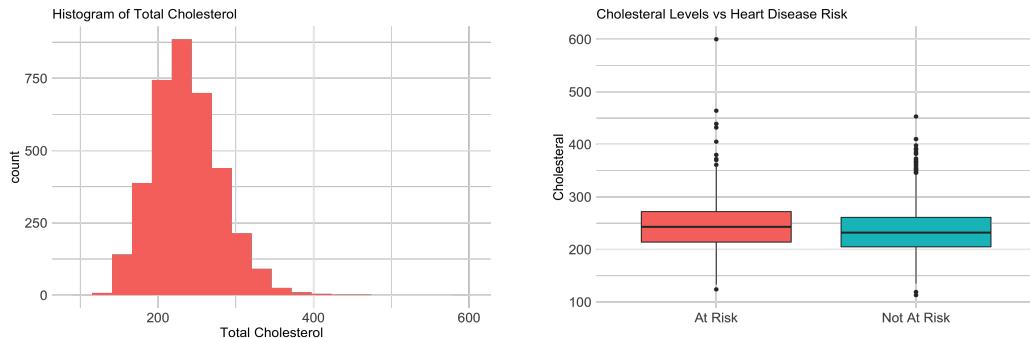
Diabetes

The majority of patients do not have diabetes, and yet, approximately 15% of these patients are at risk for 10-year coronary heart disease. On the other hand, over 30% of patients with diabetes are at risk for 10-year coronary heart disease.



Total Cholesterol

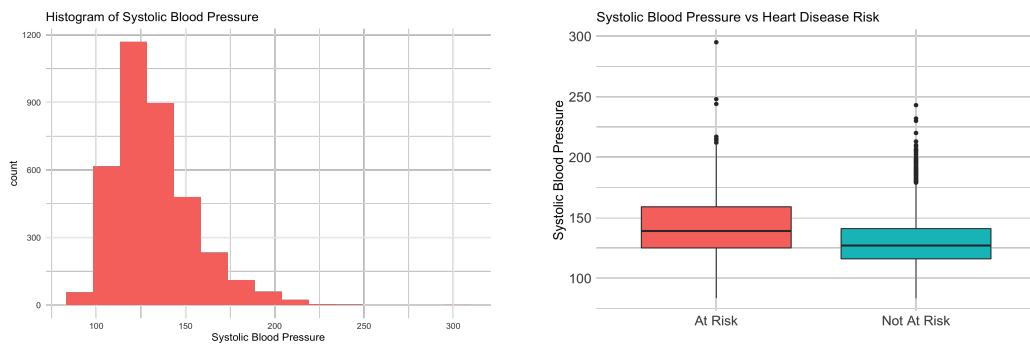
The distribution of total cholesterol values appears approximately normal. According to the boxplots below, the distribution of total cholesterol values appears to be the same regardless of whether or not patients are at risk for 10-year coronary heart disease.



Systolic Blood Pressure

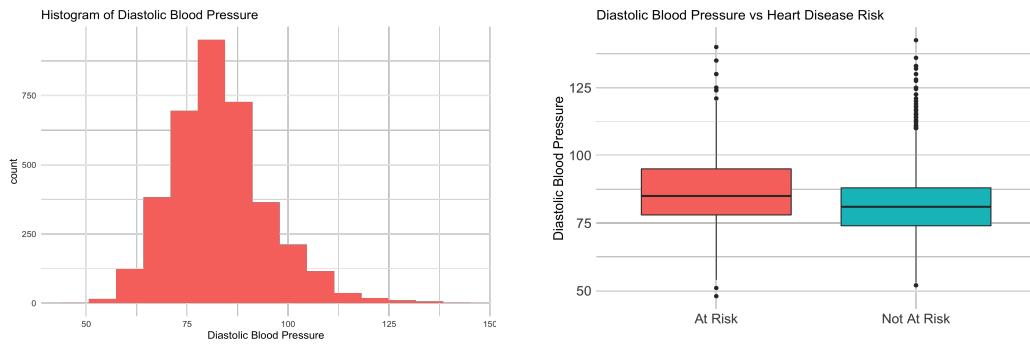
The distribution of systolic blood pressure appears approximately normal, centered around 125 mmHg. We can see that an increased level of systolic blood pressure is associated with a larger proportion of patients at risk for 10-year coronary heart

disease.



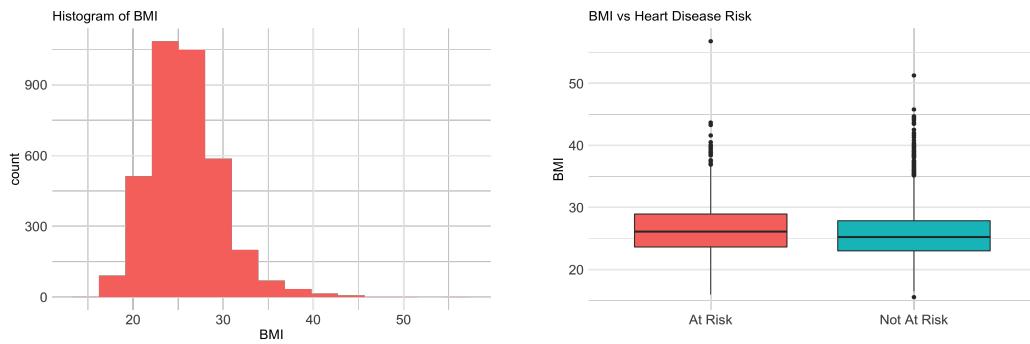
Diastolic Blood Pressure

As expected, the distribution of diastolic blood pressure also appears approximately normal, centered around 85 mmHg. Notice that the trend continues as before - an increased level of the diastolic blood pressure is associated with a larger proportion of patients at risk for 10-year coronary heart disease.



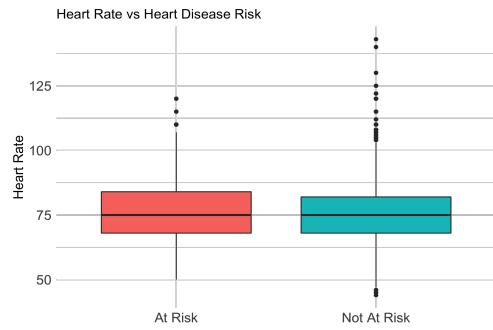
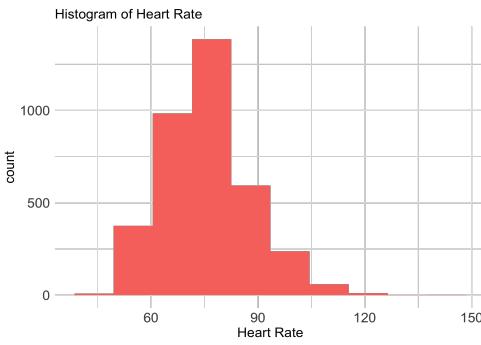
Body Mass Index

The majority of patients in our study are either at a healthy weight or overweight. Very few patients are underweight in the dataset. Despite the unequal number of patients per group, the proportion of patients at risk for 10-year coronary heart disease is approximately equal across groups.



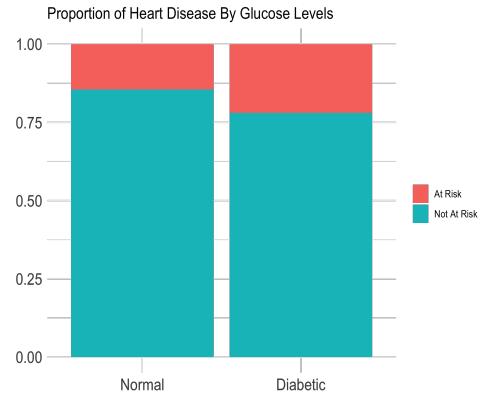
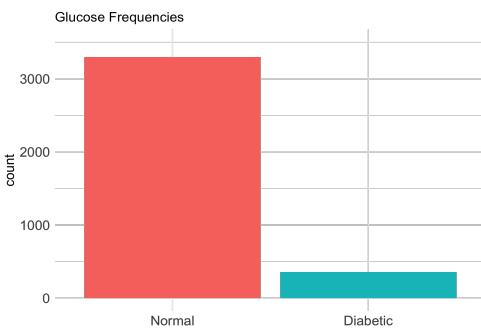
Heart Rate

According to the histogram below, patients' heart rates are approximately normally distributed. The average patient heart rate is 75 bpm and the standard deviation is 12 bpm. The distribution of patients' heart rates at risk for 10-year coronary heart disease is nearly identical to the distribution of patients' heart rates not at risk for 10-year coronary heart disease.



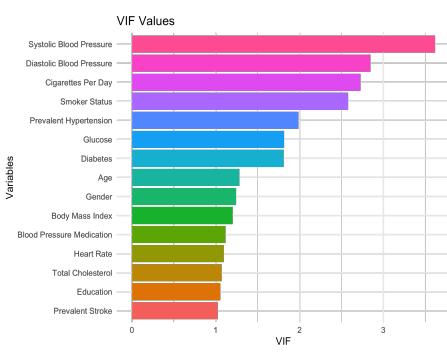
Glucose

Very few patients in the study have a diabetic glucose level. The majority of patients fall in the normal glucose range. However, a little over 15% of patients in the normal glucose range are at risk for 10-year coronary heart disease. Nearly 25% of patients in the diabetic glucose level are at risk for 10-year coronary heart disease.

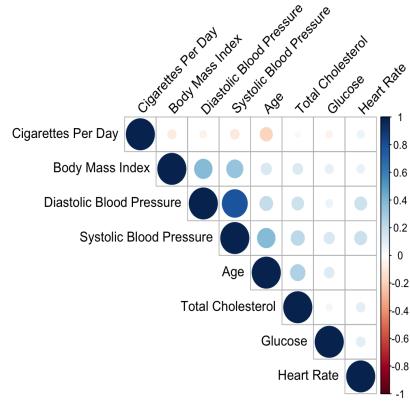


Multicollinearity

To prepare for potential multicollinearity, we examined the correlation between the predictor variables and assessed the variance inflation factors (VIFs). Although diastolic and systolic blood pressure demonstrated strong correlation, the remaining pairs of variables did not display large correlations. As expected, we found that the two largest VIF scores belong to the systolic and diastolic blood pressure predictor variables. From the VIF plot below, we can see the predictors have VIF values less than four. Therefore, we did not investigate multicollinearity further.



(a) Variance Inflation Factor (VIF)



(b) Correlation Plot

During the modeling phase, the correlation between the two blood pressure variables may present multicollinearity. However,

each model we used has its own form of variable selection that should help prevent the negative effects of highly correlated and unnecessary variables. We are confident multicollinearity will not be an issue later.

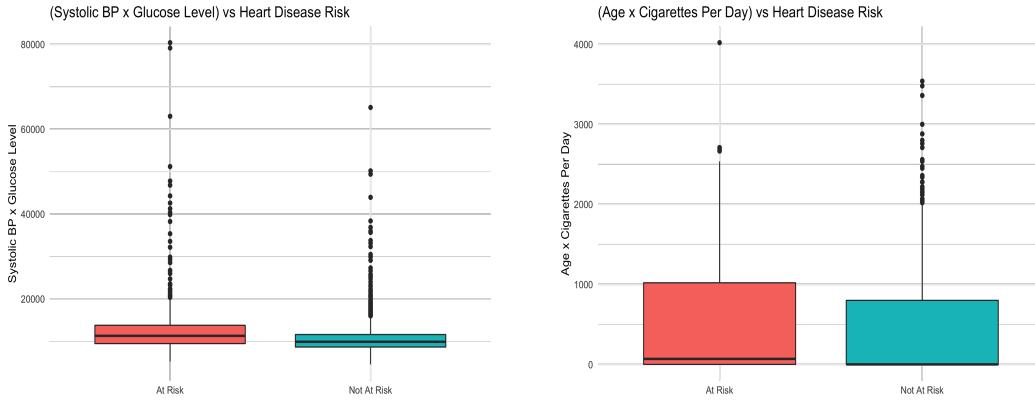
Feature Engineering

After the exploratory data analysis with the original variables, we explored how interaction terms and quadratic terms could have a significant relationship with our response variable.

Interaction Terms

An interaction term is a new term that consists of two or more of the original variables. The table below shows a few of the significant numeric interaction terms.

Interaction	W	P-Value
Age w/ Cigarettes Per Day	935590	0.0007
Systolic-BP w/ Cigarettes Per Day	930899	0.0015
Systolic-BP w/ Glucose	1102094	2.2e-16



Each interaction term has an underlying distribution. Depending on the underlying distribution, we either used a parametric or a non-parametric test to investigate if the interaction term has a significant effect on the response variable. Because the p-values above are small, we conclude that all three interaction terms are significant. The box-plots above provide further evidence that the interaction terms have an effect on the risk of heart disease.

Quadratic Terms

Quadratic terms transform variables. In particular, quadratic terms square each value in a predictor variable and can be applied to every numeric variable in our data. However, it is unwise to do so. Squaring every variable in the dataset can lead to overfitting. Instead, we determined which numeric variables to square using a more mathematical approach.

Assuming that our response variable, Y , can be modeled with logistic regression, it can be shown that if a predictor variable, X , is normally distributed with different variances when conditioned on the two values of the response, then the log odds,

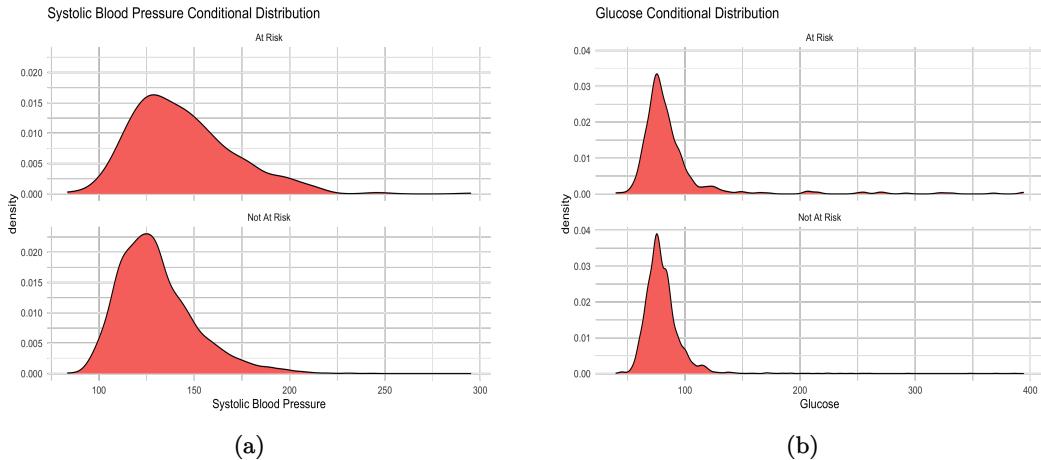
$$\log \frac{P(Y = 1|X = x)}{P(Y = 0|X = x)}$$

is a quadratic function of X such that,

$$\log \frac{P(Y = 1|X = x)}{P(Y = 0|X = x)} = \beta_0 + \beta_1 x + \beta_2 x^2$$

for constants β_0, β_1 , and $\beta_2 \in \mathbb{R}$ ([key] [cook]).

Consequently, for each numeric variable, we compared the normalized conditional distributions similar to the plots shown below. If the conditional distributions appeared approximately normal but had significantly different sample variances, the variable became a candidate for a quadratic term.



Predictor Variable	"At Risk" Std. Dev.	"Not at Risk" Std. Dev.
Systolic BP	26.966	20.414
Diastolic BP	14.399	11.320
Glucose	40.786	19.129

The three predictor variables shown in the table above all approximately satisfied the requirements outlined before, so we chose to include their quadratic terms as new variables before we started the modeling.

Methodology

In this section, we describe the methods used to best predict the 10-year risk of future heart disease. First, our response variable is binary, so standard regression models will not work with our data. Instead, we used three different logistic regression models and one modern classification model to predict the probability of 10-year risk of future heart disease.

Second, each model attempted to find the best combination of the predictor variables above. For all categorical variables, we created dummy variables with treatment coding (one-hot encoding). Note that the cigarettes-per-day and glucose variables above are actually continuous variables in the dataset. We portrayed them as categorical above to extract a more meaningful interpretation.

Third, we randomly split 80% of the data into a training set and used the other random 20% as the test set. The test set was not used by any model during the training process. Additionally, we partitioned 20% of the training data into a validation set. Recall that our dataset is largely imbalanced. Approximately 85% of the patients are not at risk of heart disease and approximately 15% of the patients are at risk for heart disease. We ensured that each split of the data had a similar balance of patients that were not at risk and at risk of heart disease.

Since the response variable in the training data is imbalanced, we used randomized oversampling to combat any bias toward not having a risk of future heart disease. By oversampling, we hoped that the models would better detect when patients would be at risk for future heart disease.

Lastly, we tuned each model's hyper-parameters. Using standard k -fold cross validation on the oversampled training data to tune our model's hyper-parameters would have led to overfitting and non-optimal hyper-parameters. Therefore, to ensure our models were not overfit to the oversampled training data, we used the independent validation set to tune each model's hyper-parameters. Once the hyper-parameters were obtained, we retrained the model on a new oversampled dataset made up of the original training set (without oversampling) and the validation set.

We evaluated and tuned each model with the area under the receiver operating characteristic (ROC-AUC) performance metric. We used the ROC-AUC over other metrics because it summarizes how well each model separates the two response classes for all possible thresholds (as opposed to a single predetermined threshold).

Logistic Regression with Backwards Elimination

Our first model was a standard logistic regression model that started with all of the available predictor variables. This model used backwards elimination for feature selection. This process starts with all predictors in the model (full model), iteratively

removes the least contributive predictors (using t-tests), and stops when the model's Akaike information criterion (*AIC*) can no longer decrease from a removal of an individual predictor variable. Ideally, this process will lead to a smaller, more significant, set of predictor variables and will be less likely to suffer from any overfitting or multicollinearity issues. After backwards selection, we refit the logistic regression model with the remaining significant variables.

Logistic Regression with PCA

Our next model uses a different approach for dimension reduction before fitting a logistic regression model. Principal component analysis (PCA) is often used to reduce the dimensions of a dataset by linearly transforming the data. A principal component is a linear combination of the predictor variables, and the entire set of principal components is mutually orthogonal. Each principal component attempts to explain as much of the variation in the feature space as possible, and all n principal components account for 100% of the variation in the data. Similar to backward selection, the goal here is to use $k < p$ principal components to mitigate any effects of multicollinearity and to prevent the chance of overfitting.

Recall that many of our predictors are in different units, so we chose to center and scale our data to prevent variables with larger values from dominating the sample variance explained. After we calculated all p principal components, we used a combination of scree plots and validation data to evaluate the variance explained and predictive performance of the first k principal components where k ranges from 1 to p . Once we obtained the optimal number of principal components, \hat{k}_{best} , we refit the logistic regression model with the first \hat{k}_{best} principal components from the entire training set.

Logistic Regression with Elastic Net Regularization

Our last logistic regression model uses another form of variable selection known as *regularization*. In general, regularization adds a penalty for each additional predictor variable in the model, and minimizes overfitting by removing or shrinking the effect of unnecessary predictor variables. Elastic net regularization combines two common regularization techniques, LASSO and Ridge (also known as *L1* and *L2* penalties), with the following formula:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left(-\frac{1}{n} \sum_{i=1}^n l(x_i, y_i; \beta) + \lambda \left[(1-\alpha) \frac{\|\beta\|_2^2}{2} + \alpha \|\beta\|_1 \right] \right)$$

where $l(x_i, y_i; \beta)$ is the log-likelihood contribution for the i th observation. To ensure a fair penalization of each predictor variable, we centered and scaled the data before training the elastic net model. Using results from our dedicated validation set, we tuned this model to find the optimal values for the mixing parameter, α , and the regularization parameter, λ . We then refit the logistic regression model to the entire training set with the optimal hyper-parameters, $\hat{\alpha}$ and $\hat{\lambda}$.

eXtreme Gradient Boosting (XGBoost)

Lastly, we fit an extreme gradient boosting classification model to predict the 10-year risk of heart disease. XGBoost is a decision-tree based algorithm that uses penalized trees, proportional leaf node shrinking, and automatic variable selection, among other features. In general, this model minimizes its respective loss function by adding the individual contributions from an ensemble of weak learners. Additionally, XGBoost is trained using a second order gradient optimization algorithm. For simplicity, we used the validation set to tune the number of trees (weak learners) used and the maximum depth of each tree.

Model Evaluation and Comparison

To examine and compare each model's predictive performance, we use our (completely unseen) test data. For each model, we generated predicted probabilities for the test set to obtain visuals such as ROC Curves, accuracy plots, sensitivity plots, specificity plots, and false positive plots. In addition to the visuals, we also computed quantitative measures such as the area under the ROC curves, confusion matrices, and a variety of other performance metrics.

Although visuals and metrics such as the ROC curve and ROC-AUC consider *all* thresholds, most of the other measures mentioned above require a specified probability threshold to generate class predictions. Given that our models are attempting to predict the presence of a life-threatening condition, we wanted to find a threshold that prioritizes true positives while not overly accepting too many false positives. In addition, we prefered to optimize our threshold in a way that does not unfairly suffer from the large class imbalance in our data. Therefore, we used the geometric mean of the *sensitivity* (or true positive rate) and the *specificity* (or true negative rate),

$$G_{mean}(t) = \sqrt{\text{Sensitivity} \cdot \text{Specificity}} = \sqrt{\frac{TP \cdot TN}{(TP + FN)(TN + FP)}}$$

to determine an optimal threshold for each model. That is, for each model, we found the threshold $t^* \in (0, 1)$ that maximizes the geometric mean, $G_{mean}(t)$. Note that only the training and validation sets were used to find t^* so that we can retain a completely unseen testing set when evaluating and comparing model performances.

Results and Discussion

Logistic Regression with Backwards Elimination

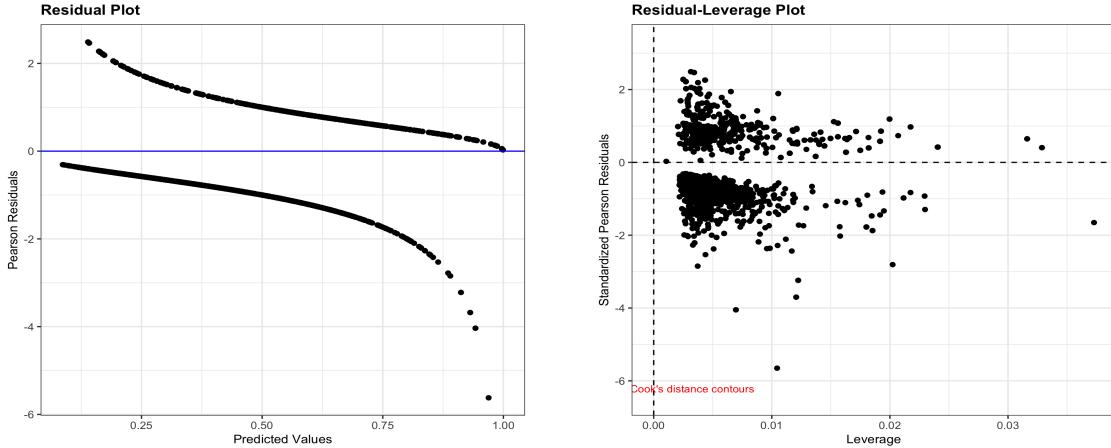
Using backwards elimination, the logistic regression model went from the original set of 23 predictors to 13 predictors. Most of the remaining predictor variables were statistically significant at the $\alpha = 0.05$ level. The backwards elimination dropped the AIC score from 2851.2 with the full model to 2836.6 with the reduced model, indicating potential improvement in the model. We tested the hypothesis that the reduced model is adequate by comparing the residual deviance of the reduced model to the residual deviance of the full model using all of the predictors. The table below shows the results of this test. We can see from the large p-value that we did not have enough evidence to reject the null hypothesis, and concluded that the model using the reduced number of predictors is adequate.

	Resid. Df	Resid. Dev	Pr(>Chi)
Reduced Model	2327	2808.6	
Full Model	2317	2803.2	≈ 0.864

Again using residual deviance, we also tested our model against the null model (with $H_0 : \beta_i = 0$ for all i). As indicated by the table below, we rejected the hypothesis and concluded that our model demonstrated strong significance between the predictors and the response.

	Resid. Df	Resid. Dev	Pr(>Chi)
Null Model	2340	3245.3	
Model Using Predictors	2327	2808.6	≈ 0

To assess our model's fit, we visually examined the Pearson Residuals vs Predicted Values plot and used the Pearson's χ^2 goodness-of-fit test to test the hypothesis that our model fit the data sufficiently well. The Pearson's χ^2 goodness-of-fit test had a p-value of 0.22, which indicated that our model fit the data sufficiently well.



Plots of Std. Pearson Residuals vs Leverage and Cook's distances were also assessed to check for potentially dangerous influential points. No observations demonstrated an unusually large combination of leverage and residual magnitude. Therefore, no observations were removed from the training set.

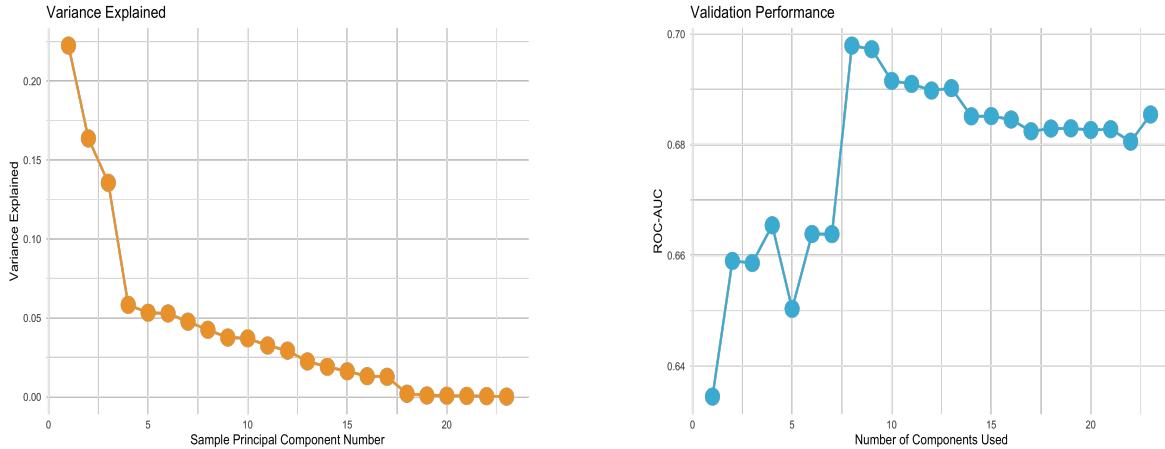
The remaining coefficients $\beta_1, \beta_2, \dots, \beta_{13}$ for this logistic regression model are presented in the table below. Since we centered and scaled our data before fitting the logistic regression model, each coefficient from the table below can be interpreted very easily. Recall that for continuous variables, each coefficient effectively represents the increase in the log odds for every standard deviation increased in the predictor variable, holding all else constant. Additionally, for categorical variables, each coefficient simply represents the increase in log odds when the variable is associated with the corresponding category as opposed to the reference category, holding all else constant. This is also known as the *log-odds ratio*.

We can exponentiate the coefficients to obtain an equivalent interpretation in terms of the odds. For example, we can see from the table below that one standard deviation increase in age is associated with a 6% ($e^{\hat{\beta}_2} = 1.06$) increase in the odds of a patient having a 10 year risk of heart disease. As another example, we can see below that a patient with hypertension is associated with a 15% ($e^{\hat{\beta}_8} = 1.15$) increase in the odds of having a 10 year risk of heart disease.

Predictor	Estimate ($\hat{\beta}_i$)	$e^{\hat{\beta}_i}$	P-Value (Wald)
Gender (male)	0.592	1.80	$\approx 2.71e - 11$
Age	0.064	1.06	$< 2e - 16$
Education (High School)	-0.093	0.911	≈ 0.362
Education (Bachelor's)	-0.452	0.641	≈ 0
Education (Graduate)	-0.186	0.830	≈ 0.161
Cigarettes per Day	0.024	1.02	$\approx 2.16e - 11$
BP Medication	0.801	2.228	≈ 0.003
Prevalent Hypertension	0.141	1.151	≈ 0.242
Total Cholesterol	0.004	1.004	≈ 0
Systolic BP	0.011	1.011	≈ 0.001
Diastolic BP	-0.118	0.888	≈ 0
Diastolic BP (Squared)	0.001	1.001	≈ 0
Glucose (Squared)	0.000	1.00	≈ 0

Logistic Regression with PCA

After performing PCA on our training set, we found that the first 9 sample principal components explained over 80% of the sample variance of our data, and the first 12 principal components explained over 90% of the sample variance. The plot (left) below shows the sample variance explained from each sample principal component computed (23 in total). There is a clear “elbow” in the plot at the 4th sample principal component.



We also found that $k = 8$ components provided the maximum ROC-AUC value of about 0.697 on our validation set. Since we know that $k = 8$ components explain almost 80% of the variance, and it achieves the best performance on our validation set, we trained our logistic regression model using the first $\hat{k}_{best} = 8$ sample principal components as input.

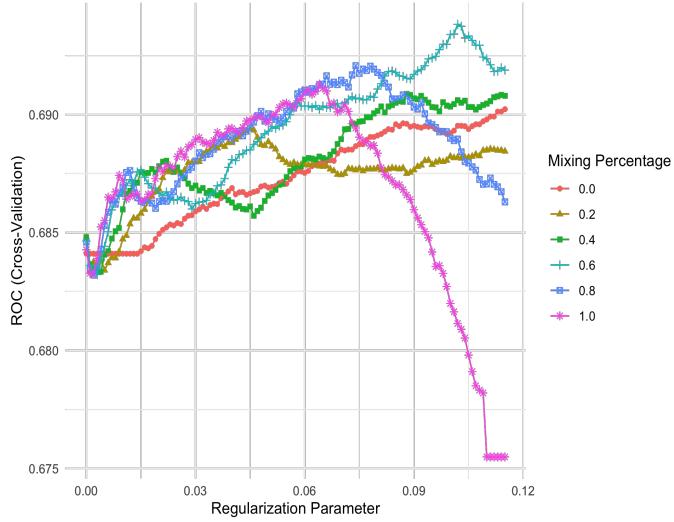
Consequently, using the first 8 principal components reduced the dimension of the feature space from 23 to 8. We found that 6 out of the 8 components were significant at the $\alpha = 0.05$ level. We used residual deviance to test our model against the null model ($H_0 : \beta_i = 0$ for all i). Again, we rejected the hypothesis and concluded that our new model demonstrated strong significance between the predictors and the response. The details of this test are shown in the table below.

	Resid. Df	Resid. Dev	Pr(>Chi)
Null Model	2340	3245.3	
Model Using Predictors	2332	2898.5	≈ 0

As we did with the previous model, we also assessed this model's fit by visually examining the Pearson Residuals vs Predicted Values plot and conducting a Pearson's χ^2 goodness-of-fit test. The Pearson's χ^2 goodness-of-fit test had a p-value of 0.30, so we rejected the hypothesis that our model did not fit the data well, and concluded that this model fits the data sufficiently well.

Logistic Regression with Elastic Net Regularization

In our final logistic regression model, we used the (scaled) training and validation data to find the optimal mixing parameter, α , and regularization parameter, λ . This process consisted of training our elastic net model with a variety of different pairs of mixing parameters ranging from $\alpha \in [0, 1]$ and $\lambda \in [0, 3]$, computing predictions for the validation set, and then comparing the corresponding ROC-AUC scores against each other. Ultimately, we found that the model with the best ROC-AUC (0.698) used a mixing percentage of $\hat{\alpha} = 0.6$ and a regularization parameter of $\hat{\lambda} = 0.098$. The plot below aided in our comparisons of ROC-AUC scores, and helped us determine the optimal hyper-parameters.



After obtaining $\hat{\alpha}$ and $\hat{\lambda}$, we trained our elastic net model one more time using both the training and the validation set. Interestingly, we found that this tuned elastic net model only retained 6 of the original predictors in its model. That is, the regularization effectively reduced the dimension of our feature space from 23 to 6, and the only remaining predictor variables with non-zero coefficients were: Gender, Age, Systolic Blood Pressure, $(\text{Systolic Blood Pressure})^2$, $(\text{Age} \cdot \text{Cigarettes Per Day})$, and $(\text{Systolic Blood Pressure} \cdot \text{Cigarettes Per Day})$.

As shown in the previous models, the deviance for the elastic net model also indicated that the 6 remaining predictors demonstrated a significant relationship with the risk of heart disease. After visually inspecting the Pearson (and Deviance) Residuals vs Predicted Values plots, we found that this model exhibited a similar goodness-of-fit when compared to the previous two logistic regression models.

XGBoost

Although this ensemble based model is very different from the previous three models, the approach for training, tuning, and validation remained similar. We used the associated ROC-AUC scores on the validation set to tune the depth of the trees and the number of trees to use in the XGBoost model. Similar to the elastic net model, we performed a "grid-search" for these two hyper-parameters which effectively trained an XGBoost model with a variety of different pairs of tree depths and numbers of trees. The ROC-AUC score for the validation set was then computed for each pair of hyper-parameters. The plot (left) below was used to find the optimal tree depth and number of trees.

The optimal tree depth is clearly 2, and we can see that once the number of trees exceeds about 13, the ROC-AUC score reaches about 0.67 and then either plateaus or drops significantly. Therefore, to prevent potentially overfitting our data and to keep our model as simple as possible, we decided that the optimal number of trees is 13. Our final XGBoost model with 13 trees and a tree depth of 2 was re-trained on both the training and validation data.

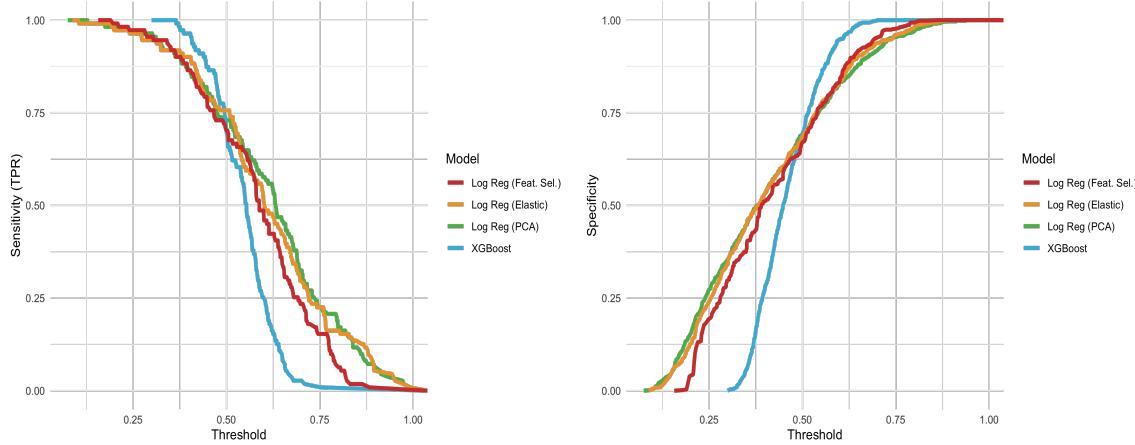
Additionally, we used our XGBoost model to estimate each predictor variable's level of relative importance, or impact, on the model's predicted probabilities. These ordered levels of importance can be seen in the feature importance plot (right) shown below. The x-axis is a measure known as "Gain". Gain is commonly described as the improvement in accuracy brought by the specific predictor to the branches (and trees) it is on [**xgbImp**]. We can immediately see that our XGBoost model places

significantly more importance on Age than it does on any other predictor. In fact, the predictor variables Age, Systolic Blood Pressure, and (Systolic Blood Pressure · Cigarettes Per Day) all demonstrated significant importance in our XGBoost model. Notice that these results seem to agree with our elastic net model. The three most important variables in the XGBoost model are among the six remaining predictors in the elastic net model.



Model Evaluation and Comparison

In this final results section, we use the dedicated test data to assess and compare the overall predictive performance of each model through a variety of metrics and visuals. We started by examining the metrics of interest over all possible thresholds. Recall that the metrics of interest are true-positive rate and true-negative rate as they are not influenced by the large class imbalance. That is, assessing performance with a measure like accuracy would not be appropriate in our case because a model that always classifies a patient as “not at risk of heart disease” would achieve an 85% accuracy on our test set. Thus, we present the plots of sensitivity and specificity vs threshold:



As we mentioned in previous sections, since our model is attempting to predict the presence of a life-threatening condition, a false-negative is far more dangerous than a false-positive. Ideally, we wanted the model to have the greatest true-positive rate possible without compromising the overall “trust” in its classifications. That is, our goal was to maximize the sensitivity without dropping the specificity too much. The plots above indicate that there exists a range of thresholds between 0.4 to 0.6 that could accomplish this goal. Notice that since the curves for the XGBoost model are much steeper than the curves for the logistic regression models, this optimal range of thresholds may be a bit smaller for the XGBoost model than it is for the three logistic regression models.

As mentioned in the methodology section, we used the geometric mean of the sensitivity and the specificity to find the optimal threshold, t^* , for each model. We evaluated the $G_{mean}(t)$ at a variety of different thresholds between $t \in [0.4, 0.6]$, and stored the threshold that maximized the $G_{mean}(t)$. Using the optimal thresholds for each model, we created the following performance metrics and confusion matrices from our test set:

Model	Sensitivity	Specificity	Balanced Accuracy
Logistic Regression (with Feat. Sel.)	0.802	0.608	0.705
Logistic Regression (with PCA)	0.739	0.700	0.719
Logistic Regression (with Elastic Net)	0.775	0.661	0.718
XGBoost	0.703	0.673	0.688

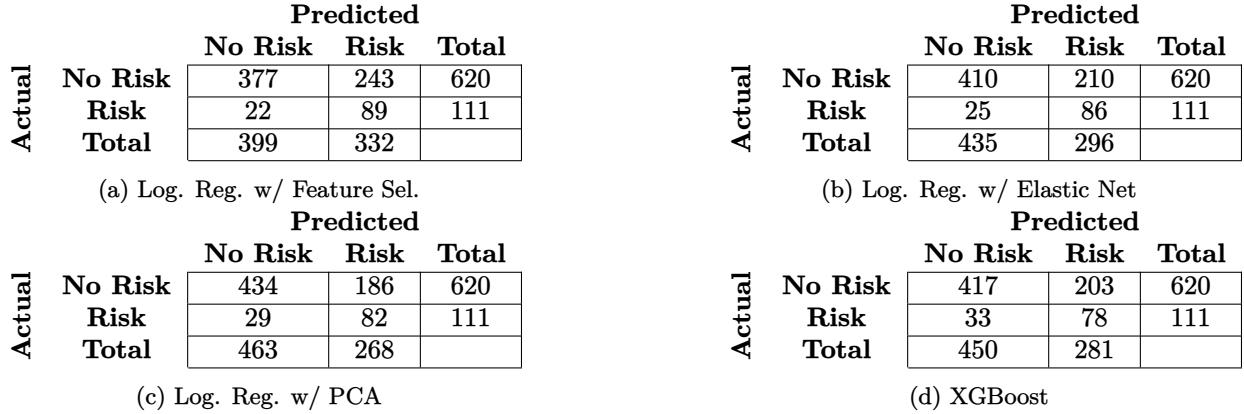
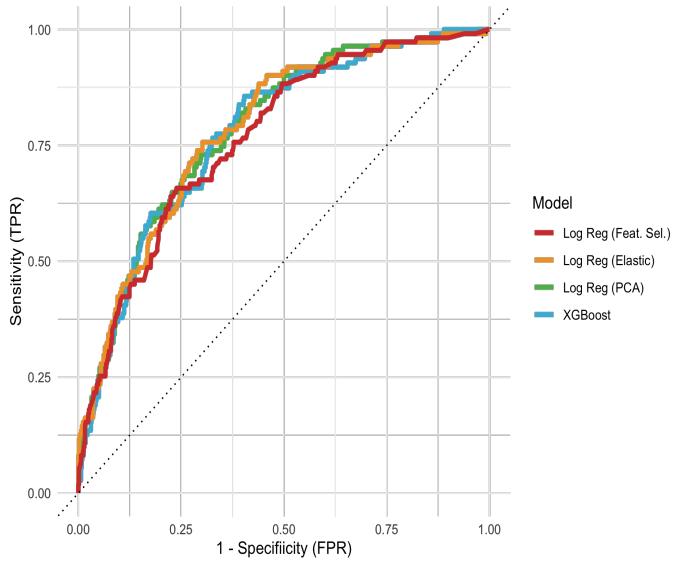


Figure: Confusion Matrix For Each Model

From the performance metrics and confusion matrices above, we can see that all four models demonstrate a good classification performance. More specifically, the logistic regression model using principal component analysis for dimension reduction and the logistic regression model using elastic net regularization demonstrate the best combinations of sensitivity and specificity. Both of these models achieve similar balanced accuracies at approximately 0.72 and clearly do a great job of distinguishing between patients with and without a 10-year risk of heart disease. It's important to note that the elastic net regression model only needed 6 predictors (the smallest feature space) to achieve such an excellent score on the test set.

For a more general assessment and comparison across all possible thresholds, we also used the testing data to compute the ROC curves for each model (shown below) and their associated ROC-AUCs.

Model	ROC-AUC
Logistic Regression with Feat. Sel.	0.785
Logistic Regression with PCA	0.778
Logistic Regression with Elastic Net	0.784
eXtreme Gradient Boosting	0.766



Although it is difficult to compare the models visually from this plot, it's clear from both the plot and the associated table of ROC-AUC scores that all four models demonstrated great predictive performance across the entire range of thresholds. They each exhibit an ability to successfully classify whether or not the patients have a 10-year risk of heart disease, and they appear to produce predictive results far better than a model that purely guesses (which is displayed by the black dotted line). The associated ROC-AUCs from the table above also tell us that the logistic regression model using only feature selection

distinguishes patients with and without a 10-year risk of heart disease the best across the whole range of thresholds. The logistic regression model using PCA for dimension reduction has a very close ROC-AUC score to the model using feature selection, and the difference in performance is so small that it may simply be a product of random variation when we initially split the test data. Interestingly, we can see that out of all the models, the XGBoost model seems to have had the worst testing performance across the entire range of thresholds.

Conclusion and Limitations

In conclusion, we found several models that can successfully predict the 10-year risk of future heart disease. In fact, each model reduced the number of variables needed to predict the 10-year risk of heart disease. The subset of features from each model could help doctors and patients save countless time and money. Theoretically, a doctor could order specific tests related to the optimal variables in our model, and patients with lower income could save thousands of dollars with fewer tests. With so many different factors that can help predict the risk of 10-year heart disease, our results can give medical professionals a starting point. It is quite remarkable that we can obtain these results with machine learning and various forms of feature selection without a medical degree. Perhaps more advanced heart rate metrics and machines could provide better variables to predict the 10-year risk of heart disease. However, many of the features we used can be obtained in an annual physical.

Presentation Questions & Answers

Question #1

Why did you choose XGBoost over a Random Forest?

Although the random forest model generally provides high accuracy, it is also prone to overfitting. Random forests unfortunately tend to overfit the training data when sampled data points are very similar to one another. Especially since we oversampled observations from the minority class in our training data, similar data points would have been seen frequently while training our model. Therefore, in order to prevent the chances of overfitting we decided to use XGBoost instead.

The XGBoost model uses an ensemble of weak learners (trees), and the depth of each tree is generally very short. These shorter trees often prevent overfitting and allow for a better generalization of the model. Therefore, we believed that the XGBoost's ability to counter overfitting would lead to a stronger predictive performance on our test set.

Additionally, we have both used random forest models in other projects throughout the MAS program and have never used XGBoost in a project before. We figured this project would be even more interesting if we modeled our data using XGBoost and officially learned how to tune the parameters in a practical setting.

Question #2

Which transformation method did you use before performing PCA?

Since many of our predictor variables are in different units and had varying scales, we decided to **standardize** the data using the sample mean and standard deviation from the training set. That is, each original feature, \mathbf{X}_i , was centered and scaled such that:

$$\mathbf{X}'_i = \frac{\mathbf{X}_i - \bar{x}_i}{\hat{\sigma}_i}$$

where \bar{x}_i and $\hat{\sigma}_i$ are the feature's associated sample mean and sample standard deviation found from the training set. We then performed our modeling using the transformed features, \mathbf{X}'_i , instead of the original features. Note that using the training set to estimate the mean and standard deviation was essential to reduce any possibility of data leakage (direct information about the testing set being shared with the training set).

Appendix

We include the code used for our exploratory data analysis below:

```
1 # Set Environment Variables For Input/Output Paths
2
3 ## define input path
4 PATH_MAIN_DATA    <- Sys.getenv("STATS_412_PROJECT_DATA")
5
6 ## define output path
7 PATH_IMAGE_DIR   <- Sys.getenv("STATS_412_PROJECT_IMAGE_DIR")
8
9
10
11 # Load the necessary libraries and read the dataset
12
13
14 library(tidyverse)
15
16 rawdf <- read.csv(paste0(PATH_MAIN_DATA ,"/framingham.csv"))
17
18 df <- rawdf %>%
19   drop_na()
20
21 # proportion missing
22 1 - nrow(df) / nrow(rawdf)
23
24
25
26 # Summarize the features in the Framingham data
27
28
29 summary(df)
30
31
32
33 # Exploratory Data Analysis
34
35 ## Response
36
37
38 df$TenYearCHD[df$TenYearCHD == 0] <- "Not At Risk"
39 df$TenYearCHD[df$TenYearCHD == 1] <- "At Risk"
40 df$TenYearCHD <- factor(df$TenYearCHD)
41
42 table(df$TenYearCHD)
43
44 df %>%
45   group_by(TenYearCHD) %>%
46   summarise(Proportion = n() / nrow(df)) %>%
47   ggplot(aes(x = TenYearCHD, y = Proportion, fill = TenYearCHD)) +
48   geom_col() +
49   theme_minimal() +
50   theme(plot.title = element_text(size = 14),
51         axis.title.x = element_text(size = 14),
52         axis.title.y = element_text(size = 14),
53         axis.text = element_text(size = 14)) +
54   theme(legend.position = "None") +
55   xlab("") +
56   ggtitle("Sample Proportion of Heart Disease Risk")
57
58 ggsave(paste0(PATH_IMAGE_DIR ,"/bar_chd.png"))
59
60
61
62 ## Education Level
63
64
65 df$education[df$education == 1] <- "No High School"
66 df$education[df$education == 2] <- "High School"
67 df$education[df$education == 3] <- "Bachelors"
68 df$education[df$education == 4] <- "Graduate School"
69
70 education_levels <- c("No High School", "High School", "Bachelors", "Graduate School")
71 df$education <- factor(df$education, levels = education_levels)
```

```

72
73 table(df$education)
74
75 ggplot(df, aes(x = education, fill = education)) +
76   geom_bar() +
77   theme_minimal() +
78   theme(plot.title = element_text(size = 14),
79         axis.title.x = element_text(size = 14),
80         axis.title.y = element_text(size = 14),
81         axis.text = element_text(size = 14)) +
82   theme(legend.position = "None") +
83   xlab("") +
84   ggtitle("Education Frequencies")
85
86 ggsave(paste0(PATH_IMAGE_DIR, "/bar_education.png"))
87
88
89
90 ## Current Smoker
91
92
93 df$currentSmoker[df$currentSmoker == 0] <- "Non-Smoker"
94 df$currentSmoker[df$currentSmoker == 1] <- "Smoker"
95 df$currentSmoker <- factor(df$currentSmoker)
96
97 table(df$currentSmoker)
98 ggplot(df, aes(x = currentSmoker, fill = currentSmoker)) +
99   geom_bar() +
100  ylim(0, 2000) +
101  theme_minimal() +
102  theme(plot.title = element_text(size = 14),
103        axis.title.x = element_text(size = 14),
104        axis.title.y = element_text(size = 14),
105        axis.text = element_text(size = 14)) +
106  theme(legend.position = "None") +
107  xlab("") +
108  ggtitle("Current Smoker Frequencies")
109
110 ggsave(paste0(PATH_IMAGE_DIR, "/bar_smoker_status.png"))
111
112
113
114 ## Gender Frequency
115
116
117 df$male[df$male == 1] <- "Male"
118 df$male[df$male == 0] <- "Female"
119
120 table(df$male)
121 ggplot(df, aes(x = factor(male), fill = factor(male))) +
122   geom_bar() +
123   theme_minimal() +
124   theme(plot.title = element_text(size = 14),
125         axis.title.x = element_text(size = 14),
126         axis.title.y = element_text(size = 14),
127         axis.text = element_text(size = 14)) +
128   theme(legend.position = "None") +
129   xlab("") +
130   ggtitle("Gender Frequencies")
131
132 ggsave(paste0(PATH_IMAGE_DIR, "/bar_sex.png"))
133
134
135
136 ## Cigs Per Day
137
138
139 df$cigsPerDay <- cut(df$cigsPerDay, c(-1, 5, 10, 15, 20, 90))
140 levels(df$cigsPerDay) <- c("None", "Light", "Occasional", "Many", "Addiction")
141
142 table(df$cigsPerDay)
143 ggplot(df, aes(x = cigsPerDay, fill = cigsPerDay)) +
144   geom_bar() +
145   theme_minimal() +
146   theme(plot.title = element_text(size = 14),
147         axis.title.x = element_text(size = 14),

```

```

148     axis.title.y = element_text(size = 14),
149     axis.text = element_text(size = 14)) +
150   theme(legend.position = "None") +
151   xlab("") +
152   ggtitle("Cigarettes Per Day")
153
154 ggsave(paste0(PATH_IMAGE_DIR, "/bar_cigs_per_day.png"))
155
156
157
158 ## BP Meds
159
160
161 df$BPMeds[df$BPMeds == 0] <- "No"
162 df$BPMeds[df$BPMeds == 1] <- "Yes"
163 df$BPMeds <- factor(df$BPMeds)
164
165 table(df$BPMeds)
166
167 ggplot(df, aes(x = BPMeds, fill = BPMeds)) +
168   geom_bar() +
169   theme_minimal() +
170   theme(plot.title = element_text(size = 14),
171         axis.title.x = element_text(size = 14),
172         axis.title.y = element_text(size = 14),
173         axis.text = element_text(size = 14)) +
174   theme(legend.position = "None") +
175   xlab("") +
176   ggtitle("Blood Pressure Medication Frequencies")
177
178 ggsave(paste0(PATH_IMAGE_DIR, "/bar_bp_meds.png"))
179
180
181
182 ## Prevalent Strokes
183
184
185 df$prevalentStroke[df$prevalentStroke == 0] <- "No"
186 df$prevalentStroke[df$prevalentStroke == 1] <- "Yes"
187 df$prevalentStroke <- factor(df$prevalentStroke)
188
189 table(df$prevalentStroke)
190
191 ggplot(df, aes(x = prevalentStroke, fill = prevalentStroke)) +
192   geom_bar() +
193   theme_minimal() +
194   theme(plot.title = element_text(size = 14),
195         axis.title.x = element_text(size = 14),
196         axis.title.y = element_text(size = 14),
197         axis.text = element_text(size = 14)) +
198   theme(legend.position = "None") +
199   xlab("") +
200   ggtitle("History of Strokes Frequencies")
201
202 ggsave(paste0(PATH_IMAGE_DIR, "/bar_strokes.png"))
203
204
205
206 ## Hypertension
207
208
209 df$prevalentHyp[df$prevalentHyp == 0] <- "No"
210 df$prevalentHyp[df$prevalentHyp == 1] <- "Yes"
211 df$prevalentHyp <- factor(df$prevalentHyp)
212
213 table(df$prevalentHyp)
214
215 ggplot(df, aes(x = prevalentHyp, fill = prevalentHyp)) +
216   geom_bar() +
217   theme_minimal() +
218   theme(plot.title = element_text(size = 14),
219         axis.title.x = element_text(size = 14),
220         axis.title.y = element_text(size = 14),
221         axis.text = element_text(size = 14)) +
222   theme(legend.position = "None") +
223   xlab("") +

```

```

224 ggttitle("Hypertension Status Frequencies")
225
226 ggsave(paste0(PATH_IMAGE_DIR, "/bar_hypertension.png"))
227
228
229
230 ## DiaBP
231
232
233 df$diaBP <- as.numeric(df$diaBP)
234 ggplot(df, aes(x = diaBP, fill = "#F8766D")) +
235   geom_histogram(bins = 15) +
236   theme_minimal() +
237   theme(legend.position = "None") +
238   xlab("Diastolic Blood Pressure") +
239   ggttitle("Histogram of Diastolic Blood Pressure")
240
241 ggsave(paste0(PATH_IMAGE_DIR, "/hist_dbp.png"))
242
243 df$diaBP <- cut(df$diaBP, c(0, 80, 89, max(df$diaBP)))
244 levels(df$diaBP) <- c('Average', 'Above Average', 'High')
245 table(df$diaBP)
246
247 ggplot(df, aes(x = diaBP, fill = diaBP)) +
248   geom_bar() +
249   theme_minimal() +
250   theme(plot.title = element_text(size = 14),
251         axis.title.x = element_text(size = 14),
252         axis.title.y = element_text(size = 14),
253         axis.text = element_text(size = 14)) +
254   theme(legend.position = "None") +
255   xlab("") +
256   ggttitle("Diastolic Blood Pressure Frequencies")
257
258 ggsave(paste0(PATH_IMAGE_DIR, "/bar_dbp.png"))
259
260
261
262 ## Glucose
263
264
265 ggplot(df, aes(x = glucose, fill = "#F8766D")) +
266   geom_histogram(bins = 20) +
267   theme_minimal() +
268   theme(legend.position = "None") +
269   ggttitle("Histogram of Glucose")
270
271 ggsave(paste0(PATH_IMAGE_DIR, "/hist_glucose.png"))
272
273 df$glucose <- cut(df$glucose, c(0, 98, max(df$glucose)))
274 levels(df$glucose) <- c('Normal', 'Diabetic')
275
276 ggplot(df, aes(x = glucose, fill = glucose)) +
277   geom_bar() +
278   ylim(0, 3500) +
279   theme_minimal() +
280   theme(plot.title = element_text(size = 14),
281         axis.title.x = element_text(size = 14),
282         axis.title.y = element_text(size = 14),
283         axis.text = element_text(size = 14)) +
284   theme(legend.position = "None") +
285   xlab("") +
286   ggttitle("Glucose Frequencies")
287
288 ggsave(paste0(PATH_IMAGE_DIR, "/bar_glucose.png"))
289
290
291
292 ## Age
293
294
295 ggplot(df, aes(x = age, fill = "#F8766D")) +
296   geom_histogram(bins = 8) +
297   theme_minimal() +
298   theme(legend.position = "None") +
299   ggttitle("Histogram of Age")

```

```

300
301 ggsave(paste0(PATH_IMAGE_DIR, "/hist_age.png"))
302
303 df$age <- cut(df$age, c(30, 40, 50, 60, 70))
304 levels(df$age) <- c("30's", "40's", "50's", "Over 60")
305
306 table(df$age)
307
308 df %>%
309   ggplot(aes(x = age, fill = age)) +
310   geom_bar() +
311   theme_minimal() +
312   theme(plot.title = element_text(size = 14),
313         axis.title.x = element_text(size = 14),
314         axis.title.y = element_text(size = 14),
315         axis.text = element_text(size = 14)) +
316   theme(legend.position = "None") +
317   xlab("") +
318   ggtitle("Age Group Frequencies")
319
320 ggsave(paste0(PATH_IMAGE_DIR, "/bar_age.png"))
321
322
323
324 ## SysBP
325
326
327 df$sysBP <- as.numeric(df$sysBP)
328 ggplot(df, aes(x = sysBP, fill = "#F8766D")) +
329   geom_histogram(bins = 15) +
330   theme_minimal() +
331   theme(legend.position = "None") +
332   xlab("Systolic Blood Pressure") +
333   ggtitle("Histogram of Systolic Blood Pressure")
334
335 ggsave(paste0(PATH_IMAGE_DIR, "/hist_sbp.png"))
336
337 df$sysBP <- cut(df$sysBP, c(0, 120, 139, max(df$sysBP)))
338 levels(df$sysBP) <- c('Average', 'Above Average', 'High')
339 table(df$sysBP)
340
341 ggplot(df, aes(x = sysBP, fill = sysBP)) +
342   geom_bar() +
343   theme_minimal() +
344   theme(plot.title = element_text(size = 14),
345         axis.title.x = element_text(size = 14),
346         axis.title.y = element_text(size = 14),
347         axis.text = element_text(size = 14)) +
348   theme(legend.position = "None") +
349   xlab("") +
350   ggtitle("Systolic Blood Pressure Frequencies")
351
352 ggsave(paste0(PATH_IMAGE_DIR, "/bar_sbp.png"))
353
354
355
356 ## Heart Rate
357
358
359 ggplot(df, aes(x = heartRate, fill = "#F8766D")) +
360   geom_histogram(bins = 10) +
361   theme_minimal() +
362   theme(plot.title = element_text(size = 14),
363         axis.title.x = element_text(size = 14),
364         axis.title.y = element_text(size = 14),
365         axis.text = element_text(size = 14)) +
366   theme(legend.position = "None") +
367   ggtitle("Histogram of Heart Rate") +
368   xlab("Heart Rate")
369
370 ggsave(paste0(PATH_IMAGE_DIR, "/hist_heart_rate.png"))
371
372
373
374 ## BMI
375
```

```

376
377 ggplot(df, aes(x = BMI, fill = "#F8766D")) +
378   geom_histogram(bins = 15) +
379   theme_minimal() +
380   theme(plot.title = element_text(size = 14),
381         axis.title.x = element_text(size = 14),
382         axis.title.y = element_text(size = 14),
383         axis.text = element_text(size = 14)) +
384   theme(legend.position = "None") +
385   ggtitle("Histogram of BMI")
386
387 ggsave(paste0(PATH_IMAGE_DIR, "/hist_bmi.png"))
388
389 df$BMI <- cut(df$BMI, c(0, 18.5, 25, 30, max(df$BMI)))
390 levels(df$BMI) <- c('Under Weight', 'Healthy Weight', 'Over Weight', 'Obese')
391
392 ggplot(df, aes(x = BMI, fill = BMI)) +
393   geom_bar() +
394   theme_minimal() +
395   theme(plot.title = element_text(size = 14),
396         axis.title.x = element_text(size = 14),
397         axis.title.y = element_text(size = 14),
398         axis.text = element_text(size = 14)) +
399   theme(legend.position = "None") +
400   xlab("") +
401   ggtitle("BMI Frequencies")
402
403 ggsave(paste0(PATH_IMAGE_DIR, "/bar_bmi.png"))
404
405
406
407 ## Diabetes Status
408
409
410 df$diabetes[df$diabetes == 0] <- "No Diabetes"
411 df$diabetes[df$diabetes == 1] <- "Diabetes"
412 df$diabetes <- factor(df$diabetes)
413
414 ggplot(df, aes(x = diabetes, fill = diabetes)) +
415   geom_bar() +
416   theme_minimal() +
417   theme(plot.title = element_text(size = 14),
418         axis.title.x = element_text(size = 14),
419         axis.title.y = element_text(size = 14),
420         axis.text = element_text(size = 14)) +
421   theme(legend.position = "None") +
422   xlab("") +
423   ggtitle("Diabetes Frequency")
424
425 ggsave(paste0(PATH_IMAGE_DIR, "/bar_diabetes.png"))
426
427
428
429 ## Cholesterol
430
431 ggplot(df, aes(x = totChol, fill = "#F8766D")) +
432   geom_histogram(bins = 20) +
433   theme_minimal() +
434   theme(plot.title = element_text(size = 14),
435         axis.title.x = element_text(size = 14),
436         axis.title.y = element_text(size = 14),
437         axis.text = element_text(size = 14)) +
438   theme(legend.position = "None") +
439   xlab("Total Cholesterol") +
440   ggtitle("Histogram of Total Cholesterol")
441
442 ggsave(paste0(PATH_IMAGE_DIR, "/hist_cholesterol.png"))
443
444
445
446 # Two Way Relationships
447
448 ## Numeric Variables Boxplots
449
450
451 df2 <- read.csv(paste0(PATH_MAIN_DATA, "/framingham.csv")) %>%

```

```

452 drop_na()
453
454 df2$TenYearCHD[df2$TenYearCHD == 0] <- "Not At Risk"
455 df2$TenYearCHD[df2$TenYearCHD == 1] <- "At Risk"
456 df2$TenYearCHD <- factor(df2$TenYearCHD)
457
458 ggplot(df2, aes(x = TenYearCHD, y = diaBP, fill = TenYearCHD)) +
459   geom_boxplot() +
460   theme_minimal() +
461   theme(plot.title = element_text(size = 14),
462         axis.title.x = element_text(size = 14),
463         axis.title.y = element_text(size = 14),
464         axis.text = element_text(size = 14)) +
465   theme(legend.position = "None") +
466   xlab("") +
467   ylab("Diastolic Blood Pressure") +
468   ggtitle("Diastolic Blood Pressure vs Heart Disease Risk")
469
470 ggsave(paste0(PATH_IMAGE_DIR, "/box_dbp.png"))
471
472 ggplot(df2, aes(x = TenYearCHD, y = totChol, fill = TenYearCHD)) +
473   geom_boxplot() +
474   theme_minimal() +
475   theme(plot.title = element_text(size = 14),
476         axis.title.x = element_text(size = 14),
477         axis.title.y = element_text(size = 14),
478         axis.text = element_text(size = 14)) +
479   theme(legend.position = "None") +
480   xlab("") +
481   ylab("Cholesterol") +
482   ggtitle("Cholesterol Levels vs Heart Disease Risk")
483
484 ggsave(paste0(PATH_IMAGE_DIR, "/box_cholesterol.png"))
485
486 ggplot(df2, aes(x = TenYearCHD, y = age, fill = TenYearCHD)) +
487   geom_boxplot() +
488   theme_minimal() +
489   theme(plot.title = element_text(size = 14),
490         axis.title.x = element_text(size = 14),
491         axis.title.y = element_text(size = 14),
492         axis.text = element_text(size = 14)) +
493   theme(legend.position = "None") +
494   xlab("") +
495   ylab("Age") +
496   ggtitle("Age vs Heart Disease Risk")
497
498 ggsave(paste0(PATH_IMAGE_DIR, "/box_age.png"))
499
500 ggplot(df2, aes(x = TenYearCHD, y = glucose, fill = TenYearCHD)) +
501   geom_boxplot() +
502   theme_minimal() +
503   theme(plot.title = element_text(size = 14),
504         axis.title.x = element_text(size = 14),
505         axis.title.y = element_text(size = 14),
506         axis.text = element_text(size = 14)) +
507   theme(legend.position = "None") +
508   xlab("") +
509   ylab("Glucose") +
510   ggtitle("Glucose vs Heart Disease Risk")
511
512 ggsave(paste0(PATH_IMAGE_DIR, "/box_glucose.png"))
513
514 ggplot(df2, aes(x = TenYearCHD, y = sysBP, fill = TenYearCHD)) +
515   geom_boxplot() +
516   theme_minimal() +
517   theme(plot.title = element_text(size = 14),
518         axis.title.x = element_text(size = 14),
519         axis.title.y = element_text(size = 14),
520         axis.text = element_text(size = 14)) +
521   theme(legend.position = "None") +
522   xlab("") +
523   ylab("Systolic Blood Pressure") +
524   ggtitle("Systolic Blood Pressure vs Heart Disease Risk")
525
526 ggsave(paste0(PATH_IMAGE_DIR, "/box_sbp.png"))
527
```

```

528 ggplot(df2, aes(x = TenYearCHD, y = cigsPerDay, fill = TenYearCHD)) +
529   geom_boxplot() +
530   theme_minimal() +
531   theme(plot.title = element_text(size = 14),
532         axis.title.x = element_text(size = 14),
533         axis.title.y = element_text(size = 14),
534         axis.text = element_text(size = 14)) +
535   theme(legend.position = "None") +
536   xlab("") +
537   ylab("Cigarettes Per Day") +
538   ggtitle("Cigarettes Per Day vs Heart Disease Risk")
539
540 ggsave(paste0(PATH_IMAGE_DIR, "/box_cigs_per_day.png"))
541
542 ggplot(df2, aes(x = TenYearCHD, y = BMI, fill = TenYearCHD)) +
543   geom_boxplot() +
544   theme_minimal() +
545   theme(plot.title = element_text(size = 14),
546         axis.title.x = element_text(size = 14),
547         axis.title.y = element_text(size = 14),
548         axis.text = element_text(size = 14)) +
549   theme(legend.position = "None") +
550   xlab("") +
551   ylab("BMI") +
552   ggtitle("BMI vs Heart Disease Risk")
553
554 ggsave(paste0(PATH_IMAGE_DIR, "/box_bmi.png"))
555
556 ggplot(df2, aes(x = TenYearCHD, y = heartRate, fill = TenYearCHD)) +
557   geom_boxplot() +
558   theme_minimal() +
559   theme(plot.title = element_text(size = 14),
560         axis.title.x = element_text(size = 14),
561         axis.title.y = element_text(size = 14),
562         axis.text = element_text(size = 14)) +
563   theme(legend.position = "None") +
564   xlab("") +
565   ylab("Heart Rate") +
566   ggtitle("Heart Rate vs Heart Disease Risk")
567
568 ggsave(paste0(PATH_IMAGE_DIR, "/box_heart_rate.png"))
569
570
571
572 ## Conditional Distributions
573
574
575 ggplot(df2, aes(x = cigsPerDay, fill = "#F8766D")) +
576   geom_density() +
577   facet_wrap(~ TenYearCHD, nrow = 2) +
578   theme_minimal() +
579   theme(legend.position = "None") +
580   xlab("Cigarettes Per Day") +
581   ggtitle("Cigarettes Per Day Conditional Distribution")
582
583 ggsave(paste0(PATH_IMAGE_DIR, "/dens_cigs_per_day.png"))
584
585 ggplot(df2, aes(x = glucose, fill = "#F8766D")) +
586   geom_density() +
587   facet_wrap(~ TenYearCHD, nrow = 2) +
588   theme_minimal() +
589   theme(legend.position = "None") +
590   xlab("Glucose") +
591   ggtitle("Glucose Conditional Distribution")
592
593 ggsave(paste0(PATH_IMAGE_DIR, "/dens_glucose.png"))
594
595 ggplot(df2, aes(x = totChol, fill = "#F8766D")) +
596   geom_density() +
597   facet_wrap(~ TenYearCHD, nrow = 2) +
598   theme_minimal() +
599   theme(legend.position = "None") +
600   xlab("Cholesterol") +
601   ggtitle("Cholesterol Conditional Distribution")
602
603 ggsave(paste0(PATH_IMAGE_DIR, "/dens_cholesterol.png"))

```

```

604
605 ggplot(df2, aes(x = age, fill = "#F8766D")) +
606   geom_density() +
607   facet_wrap(~ TenYearCHD, nrow = 2) +
608   theme_minimal() +
609   theme(legend.position = "None") +
610   xlab("Age") +
611   ggtitle("Age Conditional Distribution")
612
613 ggsave(paste0(PATH_IMAGE_DIR, "/dens_age.png"))
614
615 ggplot(df2, aes(x = sysBP, fill = "#F8766D")) +
616   geom_density() +
617   facet_wrap(~ TenYearCHD, nrow = 2) +
618   theme_minimal() +
619   theme(legend.position = "None") +
620   xlab("Systolic Blood Pressure") +
621   ggtitle("Systolic Blood Pressure Conditional Distribution")
622
623 ggsave(paste0(PATH_IMAGE_DIR, "/dens_sbp.png"))
624
625 ggplot(df2, aes(x = BMI, fill = "#F8766D")) +
626   geom_density() +
627   facet_wrap(~ TenYearCHD, nrow = 2) +
628   theme_minimal() +
629   theme(legend.position = "None") +
630   xlab("BMI") +
631   ggtitle("BMI Conditional Distribution")
632
633 ggsave(paste0(PATH_IMAGE_DIR, "/dens_bmi.png"))
634
635 ggplot(df2, aes(x = diaBP, fill = "#F8766D")) +
636   geom_density() +
637   facet_wrap(~ TenYearCHD, nrow = 2) +
638   theme_minimal() +
639   theme(legend.position = "None") +
640   xlab("Diastolic Blood Pressure") +
641   ggtitle("Diastolic Blood Pressure Conditional Distribution")
642
643 ggsave(paste0(PATH_IMAGE_DIR, "/dens_dbp.png"))
644
645 ggplot(df2, aes(x = heartRate, fill = "#F8766D")) +
646   geom_density() +
647   facet_wrap(~ TenYearCHD, nrow = 2) +
648   theme_minimal() +
649   theme(legend.position = "None") +
650   xlab("Heart Rate") +
651   ggtitle("Heart Rate Conditional Distribution")
652
653 ggsave(paste0(PATH_IMAGE_DIR, "/dens_heart_rate.png"))
654
655 df2 %>%
656   group_by(TenYearCHD) %>%
657   summarise(age_SD = sd(age),
658             cigsPerDay_SD = sd(cigsPerDay),
659             totChol_SD = sd(totChol),
660             sysBP_SD = sd(sysBP),
661             diaBP_SD = sd(diaBP),
662             BMI_SD = sd(BMI),
663             heartRate_SD = sd(heartRate),
664             glucose_SD = sd(glucose))
665
666
667
668 ## Interactions between Strictly Numeric Variables
669
670
671 wilcox.test(sysBP*cigsPerDay ~ TenYearCHD, df2)
672
673 ggplot(df2, aes(x = TenYearCHD, y = cigsPerDay * sysBP, fill = TenYearCHD)) +
674   geom_boxplot() +
675   theme_minimal() +
676   theme(legend.position = "None") +
677   xlab("") +
678   ylab("Systolic BP x Cigarettes Per Day") +
679   ggtitle("(Systolic BP x Cigarettes Per Day) vs Heart Disease Risk")

```

```

680
681 ggsave(paste0(PATH_IMAGE_DIR,"/box_int_sbp_cigs_per_day.png"))
682
683 wilcox.test(age * cigsPerDay ~ TenYearCHD, df2)
684
685 ggplot(df2, aes(x = TenYearCHD, y = cigsPerDay * age, fill = TenYearCHD)) +
686 geom_boxplot() +
687 theme_minimal() +
688 theme(legend.position = "None") +
689 xlab("") +
690 ylab("Age x Cigarettes Per Day") +
691 ggtitle("(Age x Cigarettes Per Day) vs Heart Disease Risk")
692
693 ggsave(paste0(PATH_IMAGE_DIR,"/box_int_age_cigs_per_day.png"))
694
695 wilcox.test(sysBP * glucose ~ TenYearCHD, df2)
696
697 ggplot(df2, aes(x = TenYearCHD, y = glucose * sysBP, fill = TenYearCHD)) +
698 geom_boxplot() +
699 theme_minimal() +
700 theme(legend.position = "None") +
701 xlab("") +
702 ylab("Systolic BP x Glucose Level") +
703 ggtitle("(Systolic BP x Glucose Level) vs Heart Disease Risk")
704
705 ggsave(paste0(PATH_IMAGE_DIR,"/box_int_sbp_glucose.png"))
706
707
708
709 ## Diastolic Blood Pressure Level
710
711
712 table(df$TenYearCHD, df$diaBP)
713
714 df %>%
715 group_by(TenYearCHD, diaBP) %>%
716 summarise(proportion = n() / nrow(df)) %>%
717 ggplot(aes(x = diaBP, y = proportion, fill = TenYearCHD)) +
718 geom_bar(position = "fill", stat = "identity") +
719 theme_minimal() +
720 theme(plot.title = element_text(size = 14),
721 axis.title.x = element_text(size = 14),
722 axis.title.y = element_text(size = 14),
723 axis.text = element_text(size = 14)) +
724 theme(legend.title = element_blank()) +
725 xlab("") +
726 ylab("") +
727 ggtitle("Proportion of Heart Disease Risk By Diastolic Blood Pressure Level")
728
729 ggsave(paste0(PATH_IMAGE_DIR,"/two_way_dbp.png"))
730
731
732
733 ## Diabetes Status
734
735
736 table(df$TenYearCHD, df$diabetes)
737
738 df %>%
739 group_by(TenYearCHD, diabetes) %>%
740 summarise(proportion = n() / nrow(df)) %>%
741 ggplot(aes(x = diabetes, y = proportion, fill = TenYearCHD)) +
742 geom_bar(position = "fill", stat = "identity") +
743 theme_minimal() +
744 theme(plot.title = element_text(size = 14),
745 axis.title.x = element_text(size = 14),
746 axis.title.y = element_text(size = 14),
747 axis.text = element_text(size = 14)) +
748 theme(legend.title = element_blank()) +
749 xlab("") +
750 ylab("") +
751 ggtitle("Proportion of Heart Disease Risk By Diabetes Status")
752
753 ggsave(paste0(PATH_IMAGE_DIR,"/two_way_diabetes.png"))
754
755
```

```

756
757 ## Age
758
759
760 table(df$TenYearCHD, df$age)
761
762 df %>%
763   group_by(TenYearCHD, age) %>%
764   summarise(proportion = n() / nrow(df)) %>%
765   ggplot(aes(x = age, y = proportion, fill = TenYearCHD)) +
766   geom_bar(position = "fill", stat = "identity") +
767   theme_minimal() +
768   theme(plot.title = element_text(size = 14),
769         axis.title.x = element_text(size = 14),
770         axis.title.y = element_text(size = 14),
771         axis.text = element_text(size = 14)) +
772   theme(legend.title = element_blank()) +
773   xlab("") +
774   ylab("") +
775   ggtitle("Proportion of Heart Disease Risk By Age")
776
777 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_age.png"))
778
779
780
781 ## Education Level
782
783
784 table(df$TenYearCHD, df$education)
785
786 df %>%
787   group_by(TenYearCHD, education) %>%
788   summarise(proportion = n() / nrow(df)) %>%
789   ggplot(aes(x = education, y = proportion, fill = TenYearCHD)) +
790   geom_bar(position = "fill", stat = "identity") +
791   theme_minimal() +
792   theme(plot.title = element_text(size = 14),
793         axis.title.x = element_text(size = 14),
794         axis.title.y = element_text(size = 14),
795         axis.text = element_text(size = 14)) +
796   theme(legend.title = element_blank()) +
797   xlab("") +
798   ylab("") +
799   ggtitle("Proportion of Heart Disease By Education")
800
801 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_education.png"))
802
803
804
805 ## Gender
806
807
808 table(df$TenYearCHD, df$male)
809
810 df %>%
811   group_by(TenYearCHD, male) %>%
812   summarise(proportion = n() / nrow(df)) %>%
813   ggplot(aes(x = male, y = proportion, fill = TenYearCHD)) +
814   geom_bar(position = "fill", stat = "identity") +
815   theme_minimal() +
816   theme(plot.title = element_text(size = 14),
817         axis.title.x = element_text(size = 14),
818         axis.title.y = element_text(size = 14),
819         axis.text = element_text(size = 14)) +
820   theme(legend.title = element_blank()) +
821   xlab("") +
822   ylab("") +
823   ggtitle("Proportion of Heart Disease By Gender")
824
825 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_sex.png"))
826
827
828
829 ## Smoker Status
830
831

```

```

832 table(df$TenYearCHD, df$currentSmoker)
833
834 df %>%
835   group_by(TenYearCHD, currentSmoker) %>%
836   summarise(proportion = n() / nrow(df)) %>%
837   ggplot(aes(x = currentSmoker, y = proportion, fill = TenYearCHD)) +
838   geom_bar(position = "fill", stat = "identity") +
839   theme_minimal() +
840   theme(plot.title = element_text(size = 14),
841         axis.title.x = element_text(size = 14),
842         axis.title.y = element_text(size = 14),
843         axis.text = element_text(size = 14)) +
844   theme(legend.title = element_blank()) +
845   xlab("") +
846   ylab("") +
847   ggtitle("Proportion of Heart Disease By Smoker Status")
848
849 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_smoker_status.png"))
850
851
852
853 ## Cigs Per Day
854
855 table(df$TenYearCHD, df$cigsPerDay)
856
857 df %>%
858   group_by(TenYearCHD, cigsPerDay) %>%
859   summarise(proportion = n() / nrow(df)) %>%
860   ggplot(aes(x = cigsPerDay, y = proportion, fill = TenYearCHD)) +
861   geom_bar(position = "fill", stat = "identity") +
862   theme_minimal() +
863   theme(plot.title = element_text(size = 14),
864         axis.title.x = element_text(size = 14),
865         axis.title.y = element_text(size = 14),
866         axis.text = element_text(size = 14)) +
867   theme(legend.title = element_blank()) +
868   xlab("") +
869   ylab("") +
870   ggtitle("Proportion of Heart Disease By Smoker Type")
871
872 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_cigs_per_day.png"))
873
874
875
876 ## Glucose Level
877
878 table(df$TenYearCHD, df$glucose)
879
880 df %>%
881   group_by(TenYearCHD, glucose) %>%
882   summarise(proportion = n() / nrow(df)) %>%
883   ggplot(aes(x = glucose, y = proportion, fill = TenYearCHD)) +
884   geom_bar(position = "fill", stat = "identity") +
885   theme_minimal() +
886   theme(plot.title = element_text(size = 14),
887         axis.title.x = element_text(size = 14),
888         axis.title.y = element_text(size = 14),
889         axis.text = element_text(size = 14)) +
890   theme(legend.title = element_blank()) +
891   xlab("") +
892   ylab("") +
893   ggtitle("Proportion of Heart Disease By Glucose Levels")
894
895 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_glucose.png"))
896
897
898
899 ## BP Meds
900
901
902 table(df$TenYearCHD, df$BPMeds)
903
904 df %>%
905   group_by(TenYearCHD, BPMeds) %>%
906   summarise(proportion = n() / nrow(df)) %>%
907   ggplot(aes(x = BPMeds, y = proportion, fill = TenYearCHD)) +

```

```

908 geom_bar(position = "fill", stat = "identity") +
909 theme_minimal() +
910 theme(plot.title = element_text(size = 14),
911       axis.title.x = element_text(size = 14),
912       axis.title.y = element_text(size = 14),
913       axis.text = element_text(size = 14)) +
914 theme(legend.title = element_blank()) +
915 xlab("") +
916 ylab("") +
917 ggtitle("Proportion of Heart Disease By Blood Pressure Medication Status")
918
919 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_bp_meds.png"))
920
921
922 ## Systolic Blood Pressure Level
923
924
925
926 table(df$TenYearCHD, df$sysBP)
927
928 df %>%
929 group_by(TenYearCHD, sysBP) %>%
930 summarise(proportion = n() / nrow(df)) %>%
931 ggplot(aes(x = sysBP, y = proportion, fill = TenYearCHD)) +
932 geom_bar(position = "fill", stat = "identity") +
933 theme_minimal() +
934 theme(plot.title = element_text(size = 14),
935       axis.title.x = element_text(size = 14),
936       axis.title.y = element_text(size = 14),
937       axis.text = element_text(size = 14)) +
938 theme(legend.title = element_blank()) +
939 xlab("") +
940 ylab("") +
941 ggtitle("Proportion of Heart Disease Risk By Systolic Blood Pressure Level")
942
943 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_sbp.png"))
944
945
946
947 ## BMI
948
949
950 table(df$TenYearCHD, df$BMI)
951
952 df %>%
953 group_by(TenYearCHD, BMI) %>%
954 summarise(proportion = n() / nrow(df)) %>%
955 ggplot(aes(x = BMI, y = proportion, fill = TenYearCHD)) +
956 geom_bar(position = "fill", stat = "identity") +
957 theme_minimal() +
958 theme(plot.title = element_text(size = 14),
959       axis.title.x = element_text(size = 14),
960       axis.title.y = element_text(size = 14),
961       axis.text = element_text(size = 14)) +
962 theme(legend.title = element_blank()) +
963 xlab("") +
964 ylab("") +
965 ggtitle("Proportion of Heart Disease By Body Mass Index")
966
967 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_bmi.png"))
968
969
970
971 ## Hypertension
972
973 table(df$TenYearCHD, df$prevalentHyp)
974
975 df %>%
976 group_by(TenYearCHD, prevalentHyp) %>%
977 summarise(proportion = n() / nrow(df)) %>%
978 ggplot(aes(x = prevalentHyp, y = proportion, fill = TenYearCHD)) +
979 geom_bar(position = "fill", stat = "identity") +
980 theme_minimal() +
981 theme(plot.title = element_text(size = 14),
982       axis.title.x = element_text(size = 14),
983       axis.title.y = element_text(size = 14),

```

```

984     axis.text = element_text(size = 14)) +
985   theme(legend.title = element_blank()) +
986   xlab("") +
987   ylab("") +
988   ggtitle("Proportion of Heart Disease By Hypertension")
989
990 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_hypertension.png"))
991
992
993
994 ## Prevalent Stroke
995
996 table(df$TenYearCHD, df$prevalentStroke)
997
998 df %>%
999   group_by(TenYearCHD, prevalentStroke) %>%
1000   summarise(proportion = n() / nrow(df)) %>%
1001   ggplot(aes(x = prevalentStroke, y = proportion, fill = TenYearCHD)) +
1002   geom_bar(position = "fill", stat = "identity") +
1003   theme_minimal() +
1004   theme(plot.title = element_text(size = 14),
1005         axis.title.x = element_text(size = 14),
1006         axis.title.y = element_text(size = 14),
1007         axis.text = element_text(size = 14)) +
1008   theme(legend.title = element_blank()) +
1009   xlab("") +
1010   ylab("") +
1011   ggtitle("Proportion of Heart Disease By Stroke")
1012
1013 ggsave(paste0(PATH_IMAGE_DIR, "/two_way_stroke.png"))
1014
1015
1016
1017 ## Categorical Variables Chisq tests
1018
1019
1020 chisq.test(df$TenYearCHD, df$glucose)
1021 chisq.test(df$TenYearCHD, df$currentSmoker)
1022 chisq.test(df$TenYearCHD, df$education)
1023 chisq.test(df$TenYearCHD, df$cigsPerDay)
1024 chisq.test(df$TenYearCHD, df$prevalentHyp)
1025 chisq.test(df$TenYearCHD, df$diaBP)
1026 chisq.test(df$TenYearCHD, df$male)
1027 chisq.test(df$TenYearCHD, df$BMI)
1028 chisq.test(df$TenYearCHD, df$sysBP)
1029 chisq.test(df$TenYearCHD, df$BPMeds)
1030 chisq.test(df$TenYearCHD, df$prevalentStroke)
1031 chisq.test(df$TenYearCHD, df$diabetes)
1032 chisq.test(df$TenYearCHD, df$age)
1033
1034
1035
1036 # VIF
1037
1038
1039 glmodel1 <- glm(TenYearCHD ~ ., data = df2, family = binomial)
1040
1041 data.frame(DAAG::vif(glmodel1)) %>%
1042   rownames_to_column(var = "var") %>%
1043   ggplot(aes(x = DAAG..vif.glmodel1., y = fct_reorder(var, DAAG..vif.glmodel1.),
1044             fill = fct_reorder(var, DAAG..vif.glmodel1.))) +
1045   geom_col() +
1046   scale_y_discrete(labels = c("Prevalent Stroke", "Education", "Total Cholesterol", "Heart Rate", "Blood
1047   Pressure Medication", "Body Mass Index", "Gender", "Age", "Diabetes", "Glucose", "Prevalent
1048   Hypertension", "Smoker Status", "Cigarettes Per Day", "Diastolic Blood Pressure", "Systolic Blood
1049   Pressure")) +
1050   theme_minimal() +
1051   theme(legend.position = "None") +
1052   xlab("VIF") +
1053   ylab("Variables") +
1054   ggtitle("VIF Values")
1055
1056 ggsave(paste0(PATH_IMAGE_DIR, "/vif.png"))

```

```
1057 # Correlation Plot
1058
1059
1060 corrplot::corrplot(
1061   cor(df2 %>% select(diaBP, BMI, cigsPerDay, age, totChol, sysBP, glucose, heartRate) %>%
1062     rename('Diastolic Blood Pressure' = diaBP,
1063           'Body Mass Index' = BMI,
1064           'Cigarettes Per Day' = cigsPerDay,
1065           'Age' = age,
1066           'Total Cholesterol' = totChol,
1067           'Systolic Blood Pressure' = sysBP,
1068           'Glucose' = glucose,
1069           'Heart Rate' = heartRate)),
1070   type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)
1071
1072 ggsave(paste0(PATH_IMAGE_DIR, "/cor.png"))
```

We include the code used for our modeling and testing below:

```
1 # Set Environment Variables For Input/Output Paths
2
3
4 ## define input path
5 PATH_MAIN_DATA    <- Sys.getenv("STATS_412_PROJECT_DATA")
6
7 ## define output path
8 PATH_IMAGE_DIR   <- Sys.getenv("STATS_412_PROJECT_IMAGE_DIR")
9
10
11 # Loading the Data & Libraries
12
13 library(caTools)
14 library(ROCR)
15 library(pROC)
16 library(dplyr)
17 library(AMR)
18 library(ggplot2)
19 library(car)
20 library(ggResidpanel)
21 library(ggfortify)
22 library(xgboost)
23 library(caret)
24 library(glmnet)
25 library(alr4)
26 library(ROSE)
27 library(sjPlot)
28
29
30
31 # Load original data, add additional variables, make encode dummy variables (treatment), and then scale
#      the data for non-PCA models
32
33
34 orig_df <- na.omit(read.csv(paste0(PATH_MAIN_DATA, "/framingham.csv")))
35 facts <- c('male', 'TenYearCHD', 'education', 'currentSmoker',
36           'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes')
37 orig_df[, facts] <- lapply(orig_df[, facts], factor)
38
39 levels(orig_df$education) <- c("no high school", "high school", "bachelors", "graduate school")
40
41 # squared terms
42 orig_df$diaBP.SQ <- (orig_df$diaBP) ^ 2
43 orig_df$glucose.SQ <- (orig_df$glucose) ^ 2
44 orig_df$sysBP.SQ <- (orig_df$sysBP) ^ 2
45
46 # interactions
47 orig_df$ageXcigsPerDay <- orig_df$age*orig_df$cigsPerDay
48 orig_df$glucoseXsysBP <- orig_df$glucose*orig_df$sysBP
49 orig_df$sysBPXcigsPerDay <- orig_df$sysBP*orig_df$cigsPerDay
50
51 # make dummy variables
52 dummy <- dummyVars(~ ., data = orig_df, fullRank = TRUE)
53 orig_df <- data.frame(predict(dummy, newdata = orig_df))
54
55 # set seed for consistency
56 set.seed(4)
57
58 # Split data, then apply ROSE to the training data to bootstrap the training set
59 train.index <- createDataPartition(orig_df$TenYearCHD.1, p = 0.8, list = F)
60 holdout_data <- orig_df[train.index[1:ceiling(length(train.index)*0.2)], ]
61 train_data <- orig_df[train.index[ceiling(length(train.index)*0.2):length(train.index)], ]
62 train_data <- ovun.sample(TenYearCHD.1 ~ ., data = train_data, seed = 4)$data
63 test_data <- orig_df[-train.index, ]
64
65 # scale the data for model 1 and model 3
66 train_data_scaled <- train_data
67 holdout_data_scaled <- holdout_data
68 test_data_scaled <- test_data
69
70
71 # scale data with no data leakage
72 normParam <- preProcess(train_data[, -18])
```

```

73 train_data_scaled[, -18] <- as.data.frame(predict(normParam, train_data[, -18]))
74 holdout_data_scaled[, -18] <- as.data.frame(predict(normParam, holdout_data[, -18]))
75 test_data_scaled[, -18] <- as.data.frame(predict(normParam, test_data[, -18]))
76
77
78
79 # Load original data, add additional variables, make encode dummy variables (treatment), and then scale
    the data for PCA
81
82
83 orig_df_PCA <- na.omit(read.csv(paste0(PATH_MAIN_DATA, "/framingham.csv")))
84 facts <- c('male', 'TenYearCHD', 'education', 'currentSmoker',
85           'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes')
86 orig_df_PCA[, facts] <- lapply(orig_df_PCA[, facts], factor)
87
88 # squared terms
89 orig_df_PCA$diaBP.SQ <- (orig_df_PCA$diaBP) ^ 2
90 orig_df_PCA$glucose.SQ <- (orig_df_PCA$glucose) ^ 2
91 orig_df_PCA$sysBP.SQ <- (orig_df_PCA$sysBP) ^ 2
92
93 # interactions
94 orig_df_PCA$ageXcigsPerDay <- orig_df_PCA$age*orig_df_PCA$cigsPerDay
95 orig_df_PCA$glucoseXsysBP <- orig_df_PCA$glucose*orig_df_PCA$sysBP
96 orig_df_PCA$sysBPXcigsPerDay <- orig_df_PCA$sysBP*orig_df_PCA$cigsPerDay
97
98 # make dummy variables for PCA
99 dummy <- dummyVars(~ ., data = orig_df_PCA, fullRank = TRUE)
100 orig_df_PCA <- data.frame(predict(dummy, newdata = orig_df_PCA))
101
102 # split data
103 set.seed(4)
104 train_index_PCA <- createDataPartition(orig_df_PCA$TenYearCHD, p = 0.8, list = F)
105 holdout_data_PCA <- orig_df_PCA[train_index_PCA[1:ceiling(length(train_index_PCA)*0.2)], ]
106 train_data_PCA <- orig_df_PCA[train_index_PCA[ceiling(length(train_index_PCA)*0.2):length(train_index_PCA)
    ], ]
107 test_data_PCA <- orig_df_PCA[-train_index_PCA, ]
108
109 # then apply ROSE to the training data to bootstrap the training set
110 train_data_PCA <- ovun.sample(TenYearCHD.1 ~ ., data = train_data_PCA, seed = 4)$data
111
112 # scale data with no data leakage
113 normParam <- preProcess(train_data_PCA[, -18])
114 train_data_PCA[, -18] <- predict(normParam, train_data_PCA[, -18])
115 holdout_data_PCA[, -18] <- predict(normParam, holdout_data_PCA[, -18])
116
117 # remove response, apply pca
118 PCs <- prcomp(train_data_PCA[, -18], scale = FALSE, center = FALSE)
119 train_data_PCA = cbind(as.data.frame(PCs$x), TenYearCHD = train_data_PCA[, 18])
120
121 # adjust new holdout and test data with training loadings (new data leakage)
122 x <- as.matrix(holdout_data_PCA[, -18])
123 x <- x %*% PCs$rotation
124 holdout_data_PCA <- data.frame(x, TenYearCHD = holdout_data_PCA[, 18])
125
126
127
128 # Logistic Regression Modeling (Using Feature Subset Selection)
129
130
131 ## Backward Stepwise Feature Selection
132
133
134 glm_bs1 <- glm(TenYearCHD.1 ~ ., data = train_data_scaled, family = binomial)
135 summary(glm_bs1)
136
137 # use backward selection
138 glm_bs2 <- step(glm_bs1, direction = "backward", trace = 0)
139 summary(glm_bs2)
140
141 # difference in deviance can be used to compare the backward selection model
142 # and the original
143 anova(glm_bs2, glm_bs1, test = "Chisq")
144
145
146
```

```

147 ## Goodness of fit
148
149
150 subset_LR <- glm(TenYearCHD.1 ~ male.1 + age + education.high.school + education.bachelors +
151   education.graduate.school + cigsPerDay + BPMeds.1 + prevalentHyp.1 +
152   totChol + sysBP + diaBP + diaBP.SQ + glucose.SQ,
153   family = binomial,
154   data = train_data_scaled)
155
156 summary(subset_LR)
157
158 # test deviance against null model
159 null_LR <- glm(TenYearCHD.1 ~ 1, data = train_data_scaled, family = binomial)
160
161 anova(null_LR, subset_LR, test = "Chisq")
162
163 # could equivalently be done
164 # pchisq(3245.3 - 2808.6, 2340-2327, lower=FALSE)
165
166 # chisq goodness of fit test
167 ts <- sum(residuals(subset_LR, type = "pearson") ^ 2)
168 cat("P-Value for the Pearson Goodness of Fit Test = ", pchisq(ts, subset_LR$df.residual, lower=FALSE), "\n")
169
170
171 ## Leverage, Influence, and Residual Analysis
172
173
174 par(mfrow = c(2, 2))
175 resid_panel(subset_LR, plots = c("resid", "qq"), theme = "minimal", type = "pearson")
176 ggsave(paste0(PATH_IMAGE_DIR, "/subset_LR_pearson_residuals.png"))
177
178 resid_panel(subset_LR, plots = c("resid", "qq"), theme = "minimal", type = "deviance")
179 ggsave(paste0(PATH_IMAGE_DIR, "/subset_LR_dev_residuals.png"))
180
181 par(mfrow = c(1, 2))
182 autoplot(subset_LR, which = c(4, 5)) + theme_minimal()
183 ggsave(paste0(PATH_IMAGE_DIR, "/subset_LR_diagnostics.png"))
184
185
186
187
188
189
190 ## Holdout Set CV
191
192
193 # defines holdout set usage for no data leakage
194 cv_train_data_scaled <- rbind(holdout_data_scaled, train_data_scaled)
195
196 cv_train_data_scaled$TenYearCHD.1 <- factor(cv_train_data_scaled$TenYearCHD.1)
197 levels(cv_train_data_scaled$TenYearCHD.1) <- c("no", "yes")
198
199 fit_on <- list(ts = nrow(holdout_data_scaled):nrow(cv_train_data_scaled))
200 pred_on <- list(hs = 1:nrow(holdout_data_scaled))
201
202 train.control <- trainControl(method = "cv", classProbs = TRUE,
203   index = fit_on, indexOut = pred_on,
204   summaryFunction = twoClassSummary)
205
206 cv_subset_glm <- train(TenYearCHD.1 ~ male.1 + age + education.high.school + education.bachelors +
207   education.graduate.school + cigsPerDay + BPMeds.1 + prevalentHyp.1 +
208   totChol + sysBP + diaBP + diaBP.SQ + glucose.SQ,
209   data = cv_train_data_scaled,
210   method = "glm",
211   family = "binomial",
212   metric = "ROC",
213   trControl = train.control)
214
215 print(cv_subset_glm)
216
217 levels(cv_train_data_scaled$TenYearCHD.1) <- c(0, 1)
218
219
220
221 # Logistic Regression (Using Lasso + Ridge = Elastic)

```

```

222
223 ## Finding Optimal Regularization Parameter
224
225
226 # defines holdout set usage for no data leakage
227 cv_train_data_scaled <- rbind(holdout_data_scaled, train_data_scaled)
228
229 cv_train_data_scaled$TenYearCHD.1 <- factor(cv_train_data_scaled$TenYearCHD.1)
230 levels(cv_train_data_scaled$TenYearCHD.1) <- c("no", "yes")
231
232 # predictor variables
233 cv_train_x <- model.matrix(TenYearCHD.1 ~ ., cv_train_data_scaled)[, -1]
234
235 # Convert the outcome (class) to a numerical variable
236 cv_train_y <- cv_train_data_scaled$TenYearCHD.1
237
238 fit_on <- list(ts = nrow(holdout_data_scaled):nrow(cv_train_data_scaled))
239 pred_on <- list(hs = 1:nrow(holdout_data_scaled))
240
241 train.control <- trainControl(method = "cv",
242                                 index = fit_on, indexOut = pred_on,
243                                 summaryFunction = twoClassSummary)
244
245 # large grid
246 regGrid <- expand.grid(alpha = seq(0, 1, by = 0.2), lambda = seq(0, 0.115, by = 0.001))
247
248 cv_elastic_glm <- train(cv_train_x, cv_train_y,
249                           method = "glmnet",
250                           family = "binomial",
251                           metric = "ROC",
252                           tuneGrid = regGrid,
253                           trControl = train.control)
254
255 ggplot(cv_elastic_glm) + theme_minimal()
256 ggsave(paste0(PATH_IMAGE_DIR, "/elastic_LR_tuning.png"))
257
258 levels(cv_train_data_scaled$TenYearCHD.1) <- c(0, 1)
259
260 # fit "optimal" model on smaller training set
261 train_x <- model.matrix(TenYearCHD.1 ~ ., train_data_scaled)[, -1]
262 train_y <- train_data_scaled$TenYearCHD.1
263
264 elastic_lr <- glmnet(train_data_scaled[, -18], train_data_scaled[, 18], alpha = 0.6, family = "binomial",
265                         lambda = 0.098)
266
267 # print coeffs
268 coef(elastic_lr)
269
270
271 ## Manual Residual Analysis
272
273
274 # formulas from lecture
275 fittedvals <- predict(elastic_lr, newx = train_x, type = "response")
276
277 pearsonResiduals <- (train_y - fittedvals) / sqrt(fittedvals*(1-fittedvals))
278
279 devianceResiduals <- ifelse(as.logical(train_y), 1, -1) *
280   sqrt(-2*(train_y * log(fittedvals) + (1-train_y)*log(1-fittedvals)))
281
282 par(mfrow = c(1, 2))
283
284 ggplot(data.frame(fittedvals, devianceResiduals), aes(x = fittedvals, y = devianceResiduals)) +
285   geom_point() +
286   geom_hline(yintercept = 0, color = "blue") +
287   ggtitle("Residuals Plot") +
288   xlab("Fitted Values") +
289   ylab("Pearson Residuals") +
290   theme_minimal()
291 ggsave(paste0(PATH_IMAGE_DIR, "/elastic_LR_pearson_residuals.png"))
292
293 ggplot(data.frame(fittedvals, devianceResiduals), aes(x = fittedvals, y = devianceResiduals)) +
294   geom_point() +
295   geom_hline(yintercept = 0, color = "blue") +
296   ggtitle("Residuals Plot") +

```

```

297 xlab("Fitted Values") +
298 ylab("Deviance Residuals") +
299 theme_minimal()
300 ggsave(paste0(PATH_IMAGE_DIR,"/elastic_LR_dev_residuals.png"))
301
302
303
304
305 ## Deviance
306
307
308 # can compare with feature selection model
309 deviance(elastic.lr)
310
311
312
313 # Logistic Regression Modeling (Using PCA)
314
315 ## Choosing the number of Principal Components
316
317
318 # summary of the sample principal components
319 summary(PCs)
320
321 # plots
322 var_explained = PCs$sdev ^ 2 / sum(PCs$sdev ^ 2)
323
324 ggplot(data.frame(x = 1:length(PCs$sdev), y = var_explained)) +
325   geom_line(aes(x = x, y = y, size=1), color = "#EEA236", show.legend = F) +
326   geom_point(aes(x = x, y = y, size=2), color = "#EEA236", show.legend = F) +
327
328   ggtitle("Variance Explained") +
329   xlab("Sample Principal Component Number") +
330   ylab("Variance Explained") +
331   theme_minimal()
332 ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_scree.png"))
333
334 tot_var_explained = cumsum(PCs$sdev ^ 2) / sum(PCs$sdev ^ 2)
335
336 ggplot(data.frame(x = 1:length(PCs$sdev), y = tot_var_explained), aes(x = x, y = y)) +
337   geom_line() +
338   ggtitle("Cumulative Variance Explained") +
339   xlab("Sample Principal Component Number") +
340   ylab("Cumulative Variance Explained") +
341   theme_minimal()
342 ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_cumul_scree.png"))
343
344 # create biplot to visualize results of PCA
345 ggplot_pca(PCs, arrows_colour = "red", arrows_size = 1) + theme_minimal()
346 ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_biplot.png"))
347
348 ggplot(train_data_PCA, aes(x = PC1, y = PC2, col = factor(TenYearCHD))) +
349   geom_point() +
350   ggtitle("PC Separability") +
351   xlab("PC1") +
352   ylab("PC2") +
353   theme_minimal() +
354   guides(color = guide_legend(title = "Has Heart Disease?")) +
355   scale_color_discrete(labels = c("No", "Yes"))
356 ggsave(paste0(PATH_IMAGE_DIR,"/pca_seperability.png"))
357
358 # not really seeing much separability
359
360 # Holdout Set CV
361
362 # defines holdout set usage for no data leakage
363 cv_train_data_PCA <- rbind(holdout_data_PCA, train_data_PCA)
364
365 cv_train_data_PCA$TenYearCHD <- factor(cv_train_data_PCA$TenYearCHD)
366 levels(cv_train_data_scaled$TenYearCHD) <- c("no", "yes")
367
368 fit_on <- list(ts = nrow(holdout_data_PCA):nrow(cv_train_data_PCA))
369 pred_on <- list(hs = 1:nrow(holdout_data_PCA))
370
371 train.control <- trainControl(method = "cv", classProbs = TRUE,
372                                 index = fit_on, indexOut = pred_on,

```

```

373                                         summaryFunction = twoClassSummary)
374
375 cv_train_data_PCA$TenYearCHD <- factor(cv_train_data_PCA$TenYearCHD)
376 levels(cv_train_data_PCA$TenYearCHD) <- c("no", "yes")
377
378 # examine the AUC performance for different number of PCs
379 vAUC <- c()
380 for (k in 1:23){
381   print(paste("***** Number of PCs = ", k, "*****"))
382   cv_pca_glm <- train(TenYearCHD ~ .,
383                         data = cv_train_data_PCA[, c(1:k, 24)],
384                         method = "glm",
385                         family = "binomial",
386                         metric = "ROC",
387                         trControl = train.control)
388   print(cv_pca_glm)
389
390   vAUC <- c(vAUC, cv_pca_glm$results$ROC)
391 }
392
393 levels(cv_train_data_PCA$TenYearCHD) <- c(0, 1)
394
395
396 ggplot(data.frame(x=1:23, vAUC)) +
397   geom_line(aes(x = 1:23, y = vAUC, size=1), color = "#46B8DA", show.legend = F) +
398   geom_point(aes(x = 1:23, y = vAUC, size=2), color = "#46B8DA", show.legend = F) +
399   ggtitle("Validation Performance") +
400   xlab("Number of Components Used") +
401   ylab("ROC-AUC") +
402   theme_minimal()
403 ggsave(paste0(PATH_IMAGE_DIR,"/pca_validation.png"))
404
405
406
407
408 ## Goodness of fit
409
410
411 pca_LR <- glm(TenYearCHD ~ ., data = train_data_PCA[, c(1:8, 24)], family = binomial)
412 summary(pca_LR)
413
414 # test deviance against null model
415 null_LR <- glm(TenYearCHD ~ 1, data = train_data_PCA[, c(1:8, 24)], family = binomial)
416 anova(null_LR, pca_LR, test = "Chisq")
417
418 # could equivalently be done
419 # pchisq(3245.3 - 2841.1, 2340-2329, lower=FALSE)
420
421 # chisq goodness of fit test
422 ts <- sum(residuals(pca_LR, type = "pearson") ^ 2)
423 cat("P-Value for the Pearson Goodness of Fit Test = ", pchisq(ts, pca_LR$df.residual, lower=FALSE), "\n")
424
425
426
427 ## Leverage, Influence, and Residual Analysis
428
429
430 par(mfrow = c(2, 2))
431 resid_panel(pca_LR, plots = c("resid", "qq"), theme = "minimal", type = "pearson")
432 ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_pearson_residuals.png"))
433
434 resid_panel(pca_LR, plots = c("resid", "qq"), theme = "minimal", type = "deviance")
435 ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_dev_residuals.png"))
436
437 par(mfrow = c(1, 2))
438 autoplot(pca_LR, which = c(4, 5)) + theme_minimal()
439
440 ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_diagnostics.png"))
441
442
443
444 # Extreme Gradient Boosted Classifier
445
446
447 train_x = data.matrix(train_data_scaled[, -18])

```

```

448 train_y = train_data_scaled[, 18]
449
450 # define final training and testing sets
451 xgb_train = xgb.DMatrix(data = train_x, label = as.numeric(train_y))
452
453 xgb.model <- xgboost(data = xgb_train,
454                         max_depth = 2,
455                         nrounds = 5,
456                         objective = 'binary:logistic',
457                         lambda = 2,
458                         gamma = 0,
459                         colsample_bytree = 1,
460                         verbose = F)
461
462
463
464 ## Holdout Set CV
465
466
467 # defines holdout set usage for no data leakage
468 cv_train_data_scaled <- rbind(holdout_data_scaled, train_data_scaled)
469
470 train_y <- factor(cv_train_data_scaled[, 18])
471 levels(train_y) <- c("no", "yes")
472
473 fit_on <- list(ts = nrow(holdout_data_scaled):nrow(cv_train_data_scaled))
474 pred_on <- list(hs = 1:nrow(holdout_data_scaled))
475
476 xgbGrid <- expand.grid(nrounds = seq(1, 100, by = 4),
477                           max_depth = c(2, 3, 4, 5),
478                           eta = 0.1, # default
479                           gamma = 0, # default
480                           colsample_bytree = 1, # default
481                           min_child_weight = 1, # default
482                           subsample = 1 # default
483 )
484
485 xgb_trcontrol <- trainControl(method = "cv", classProbs = TRUE,
486                                 index = fit_on, indexOut = pred_on,
487                                 summaryFunction = twoClassSummary)
488
489 xgb_model = train(as.matrix(cv_train_data_scaled[, -18]), train_y,
490                     trControl = xgb_trcontrol,
491                     tuneGrid = xgbGrid,
492                     method = "xgbTree",
493                     metric = "ROC",
494                     verbosity = 0
495 )
496
497 #print(xgb_model)
498 ggplot(xgb_model) + theme_minimal()
499 ggsave(paste0(PATH_IMAGE_DIR, "/xgb_tuning.png"))
500
501 levels(cv_train_data_scaled$TenYearCHD.1) <- c(0, 1)
502
503
504
505 # Retrain Best Models Using Entire Training Set
506
507 ## Update new training and test sets (no more holdout set)
508
509
510 # use original split, then apply over sampling again to the training data to
511 # bootstrap the larger training set
512 train_data <- orig_df[train.index, ]
513 train_data <- ovun.sample(TenYearCHD.1 ~ ., data = train_data, seed = 4)$data
514 test_data <- orig_df[-train.index, ]
515
516 # scale the data for model 1 and model 3
517 train_data_scaled <- train_data
518 test_data_scaled <- test_data
519
520 normParam <- preProcess(train_data[, -18])
521 train_data_scaled[, -18] <- as.data.frame(predict(normParam, train_data[, -18]))
522 test_data_scaled[, -18] <- as.data.frame(predict(normParam, test_data[, -18]))
523
```

```

524 # for PCA datasets
525 # use original split, then apply over sampling again to the training data to
526 # bootstrap the larger training set
527
528 train_data_PCA <- orig_df_PCA[train.index_PCA, ]
529 train_data_PCA <- ovun.sample(TenYearCHD.1 ~ ., data = train_data_PCA, seed = 4)$data
530 test_data_PCA <- orig_df_PCA[-train.index_PCA, ]
531
532
533 # scale data with no data leakage
534 normParam <- preProcess(train_data_PCA[, -18])
535 train_data_PCA[, -18] <- predict(normParam, train_data_PCA[, -18])
536 test_data_PCA[, -18] <- predict(normParam, test_data_PCA[, -18])
537
538 # remove response, apply pca
539 PCs <- prcomp(train_data_PCA[, -18], scale = FALSE, center = FALSE)
540 train_data_PCA = cbind(as.data.frame(PCs$x), TenYearCHD = train_data_PCA[, 18])
541
542 # adjust new holdout and test data with training loadings (new data leakage)
543 x <- as.matrix(test_data_PCA[, -18])
544 x <- x %*% PCs$rotation
545 test_data_PCA <- data.frame(x, TenYearCHD = test_data_PCA[, 18])
546
547
548
549
550 ## Logistic Regression with Feature Selection
551
552
553 best_subset.lr <- glm(TenYearCHD.1 ~ male.1 + age + education.high.school + education.bachelors +
554                         education.graduate.school + cigsPerDay + BPMeds.1 + prevalentHyp.1 +
555                         totChol + sysBP + diaBP + diaBP.SQ + glucose.SQ,
556                         family = binomial,
557                         data = train_data)
558
559 summary(best_subset.lr)
560 plot_model(best_subset.lr) +
561   theme_minimal() +
562   ggtitle("Ten Year CHD") +
563   scale_x_discrete(labels = c(expression("(Glucose)"^"2"),
564                             expression("(Diabolic Blood Pressure)"^"2"),
565                             "Diabolic Blood Pressure",
566                             "Systolic Blood Pressure",
567                             "Total Cholesterol",
568                             "Prevalent Hypertension (Yes)",
569                             "Blood Pressure Medication (Yes)",
570                             "Cigarettes Per Day",
571                             "Education (Graduate School)",
572                             "Education (Bachelor's)",
573                             "Education (High School)",
574                             "Age",
575                             "Gender (Male)"))
576
577 ggsave(paste0(PATH_IMAGE_DIR, "/subset_LR_coefs.png"))
578
579
580
581
582 ## Logistic Regression with Elastic Net
583
584
585 train_x <- model.matrix(TenYearCHD.1 ~ ., train_data)[, -1]
586 train_y <- train_data$TenYearCHD.1
587
588 test_x <- model.matrix(TenYearCHD.1 ~ ., test_data)[, -1]
589 test_y <- train_data$TenYearCHD.1
590
591 best_elastic.lr <- glmnet(train_x, train_y, alpha = 0.6, family = "binomial", lambda = 0.098)
592 coef(best_elastic.lr)
593
594
595
596 ## Logistic Regression with PCA
597
598
599 best_pca.lr <- glm(TenYearCHD ~ ., data = train_data_PCA[, c(1:8, 24)], family = binomial)

```

```

600 summary(best.pca.lr)
601
602
603
604 ## XGBoost
605
606
607 # setup data for xgboost again
608 trainx = data.matrix(train_data_scaled[, -18])
609 trainy = train_data_scaled[, 18]
610
611 # define predictor and response variables in testing set
612 testx = data.matrix(test_data_scaled[, -18])
613 testy = test_data_scaled[, 18]
614
615 # define final training and testing sets
616 xgb_train = xgb.DMatrix(data = trainx, label = as.numeric(trainy))
617 xgb_test = xgb.DMatrix(data = testx, label = as.numeric(testy))
618
619 # create best model from CV above, using entire training set
620 best.xgb.model <- xgboost(data = xgb_train,
621                           max.depth = 2,
622                           nrounds = 13,
623                           objective = 'binary:logistic',
624                           lambda = 2,
625                           gamma = 0,
626                           colsample_bytree = 1,
627                           verbose = F)
628
629 summary(best.xgb.model)
630 importance_matrix = xgb.importance(colnames(xgb_train), model = best.xgb.model)
631 importance_matrix
632
633 ggplot(importance_matrix, aes(x = Gain, y =forcats::fct_reorder(Feature, Gain))) +
634   geom_col() +
635   theme_minimal() +
636   ylab("") +
637   ggtitle("XGBoost Feature Importance") +
638   scale_y_discrete(labels = c("Age * Cigarettes Per Day",
639                             "Blood Pressure Medication (Yes)",
640                             "Total Cholesterol",
641                             "Heart Rate",
642                             "Glucose",
643                             "Diabolic Blood Pressure",
644                             "Glucose * Systolic Blood Pressure",
645                             "Gender (Male)",
646                             "Systolic Blood Pressure",
647                             "Systolic Blood Pressure * Cigarettes Per Day",
648                             "Age"))
649 ggsave(paste0(PATH_IMAGE_DIR,"/xgb_importance.png"))
650
651
652
653 # Finding Optimal Thresholds
654
655
656 train_result_subset_lr <- predict(best.subset.lr, newdata = train_data, type = "response")
657 xtrain <- model.matrix(TenYearCHD.1 ~ ., train_data)[, -1]
658 train_result_elastic_lr <- predict(best.elastic.lr, newx = train_x, type = "response")
659 train_result_pca_lr <- predict(best.pca.lr, newdata = train_data_PCA, type = "response")
660 train_result_xgb <- predict(best.xgb.model, newdata = xgb_train)
661
662 train_metrics_subset <- data.frame()
663 for (threshold in seq(0.41, 0.6, 0.01)){
664   train_subset_lr_threshold <- as.factor(as.numeric(train_result_subset_lr > threshold))
665   train_subset_lr_table <- table(train_subset_lr_threshold, factor(train_data$TenYearCHD))
666   temp <- confusionMatrix(train_subset_lr_table, positive = "1")
667   model <- "subset"
668   temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
669   train_metrics_subset <- rbind(train_metrics_subset, temp)
670 }
671
672 train_metrics_elastic <- data.frame()
673 for (threshold in seq(0.41, 0.6, 0.01)){
674   train_elastic_lr_threshold <- as.factor(as.numeric(train_result_elastic_lr > threshold))
675   train_elastic_lr_table <- table(train_elastic_lr_threshold, factor(train_y))

```

```

676 temp <- confusionMatrix(train_elastic_lr_table, positive = "1")
677 model <- "elastic"
678 temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
679 train_metrics_elastic <- rbind(train_metrics_elastic, temp)
680 }
681
682 train_metrics_pca <- data.frame()
683 for (threshold in seq(0.41, 0.6, 0.01)){
684   train_pca_lr_threshold <- as.factor(as.numeric(train_result_pca_lr > threshold))
685   train_pca_lr_table <- table(train_pca_lr_threshold, factor(train_data$TenYearCHD))
686   temp <- confusionMatrix(train_pca_lr_table, positive = "1")
687   model <- "pca"
688   temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
689   train_metrics_pca <- rbind(train_metrics_pca, temp)
690 }
691
692 train_metrics_xgb <- data.frame()
693 for (threshold in seq(0.41, 0.6, 0.01)){
694   train_xgb_threshold <- as.factor(as.numeric(train_result_xgb > threshold))
695   train_xgb_table <- table(train_xgb_threshold, factor(train_data$TenYearCHD))
696   temp <- confusionMatrix(train_xgb_table, positive = "1")
697   model <- "xgb"
698   temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
699   train_metrics_xgb <- rbind(train_metrics_xgb, temp)
700 }
701
702 train_metrics_subset %>%
703   select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
704   mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
705   mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
706   arrange(desc(Gmean))
707
708 train_metrics_elastic %>%
709   select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
710   mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
711   mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
712   arrange(desc(Gmean))
713
714 train_metrics_pca %>%
715   select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
716   mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
717   mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
718   arrange(desc(Gmean))
719
720 train_metrics_xgb %>%
721   select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
722   mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
723   mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
724   arrange(desc(Gmean))
725
726
727
728 # Testing Results
729
730
731 # store predicted probabilities for all four models
732 result_subset_lr <- predict(best_subset.lr, newdata = test_data, type = "response")
733 result_elastic_lr <- predict(best.elastic.lr, newx = test_x, type = "response")
734 result_pca_lr <- predict(best.pca.lr, newdata = test_data_PCA, type = "response")
735 result_xgb <- predict(best.xgb.model, newdata = xgb_test)
736
737 # confusion matrices
738 cat("Log Reg with Feature Selection", "\n\n")
739 subset_lr_threshold <- as.factor(as.numeric(result_subset_lr>0.45))
740 subset_lr_table <- table(subset_lr_threshold, factor(test_data$TenYearCHD))
741 confusionMatrix(subset_lr_table, positive = "1")
742
743 cat("Log Reg with Elastic", "\n\n")
744 elastic_lr_threshold <- as.factor(as.numeric(result_elastic_lr>0.49))
745 elastic_lr_table <- table(elastic_lr_threshold, factor(test_data$TenYearCHD))
746 confusionMatrix(elastic_lr_table, positive = "1")
747
748 cat("Log Reg with PCA", "\n\n")
749 pca_lr_threshold <- as.factor(as.numeric(result_pca_lr>0.51))
750 pca_lr_table <- table(pca_lr_threshold, factor(test_data_PCA$TenYearCHD))
751 confusionMatrix(pca_lr_table, positive = "1")

```

```

752
753 cat("XGBOOST", "\n\n")
754 xgb_threshold <- as.factor(as.numeric(result_xgb > 0.5))
755 xgb_table <- table(xgb_threshold, factor(test_data_scaled$TenYearCHD.1))
756 confusionMatrix(xgb_table, positive = "1")
757
758
759
760
761 # store prediction results
762 pred_subset_lr <- prediction(result_subset_lr, test_data$TenYearCHD)
763 pred_elastic_lr <- prediction(result_elastic_lr, test_data$TenYearCHD)
764 pred_pca_lr <- prediction(result_pca_lr, test_data$TenYearCHD)
765 pred_xgb <- prediction(result_xgb, test_data$TenYearCHD)
766
767 # plot different useful performance metrics
768 measures <- c("acc", "sens", "spec", "fpr")
769 meas_names <- c("Accuracy", "Sensitivity (TPR)", "Specificity", "FPR")
770 cols <- c("#D43F3A", "#EEA236", "#5CB85C", "#46B8DA")
771
772 measure <- "acc"
773 meas_name <- "Accuracy"
774 subset_df <- data.frame(x = performance(pred_subset_lr, measure)$x.values[[1]],
775                           y = performance(pred_subset_lr, measure)$y.values[[1]])
776 elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)$x.values[[1]],
777                           y = performance(pred_elastic_lr, measure)$y.values[[1]])
778 pca_df <- data.frame(x = performance(pred_pca_lr, measure)$x.values[[1]],
779                           y = performance(pred_pca_lr, measure)$y.values[[1]])
780 xgb_df <- data.frame(x = performance(pred_xgb, measure)$x.values[[1]],
781                           y = performance(pred_xgb, measure)$y.values[[1]])
782
783 ggplot() +
784   geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
785   geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
786   geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
787   geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
788   labs(color = "Model") +
789   xlab("Threshold") +
790   ylab(meas_name) +
791   scale_color_manual(values = cols,
792                      labels = c("Log Reg (Feat. Sel.)",
793                                 "Log Reg (Elastic)",
794                                 "Log Reg (PCA)",
795                                 "XGBoost")) +
796   geom_hline(yintercept = 0.85, linetype = "dotted", color = "black", size = 0.8) +
797   annotate("text", x=0.15, y=0.83, label = "Natural Class Balance", size = 2.5) +
798   theme(legend.position = "right") +
799   theme_minimal()
800 ggsave(paste0(PATH_IMAGE_DIR, "/results_acc.png"))
801
802 measure <- "sens"
803 meas_name <- "Sensitivity (TPR)"
804 subset_df <- data.frame(x = performance(pred_subset_lr, measure)$x.values[[1]],
805                           y = performance(pred_subset_lr, measure)$y.values[[1]])
806 elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)$x.values[[1]],
807                           y = performance(pred_elastic_lr, measure)$y.values[[1]])
808 pca_df <- data.frame(x = performance(pred_pca_lr, measure)$x.values[[1]],
809                           y = performance(pred_pca_lr, measure)$y.values[[1]])
810 xgb_df <- data.frame(x = performance(pred_xgb, measure)$x.values[[1]],
811                           y = performance(pred_xgb, measure)$y.values[[1]])
812
813 ggplot() +
814   geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
815   geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
816   geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
817   geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
818   labs(color = "Model") +
819   xlab("Threshold") +
820   ylab(meas_name) +
821   scale_color_manual(values = cols,
822                      labels = c("Log Reg (Feat. Sel.)",
823                                 "Log Reg (Elastic)",
824                                 "Log Reg (PCA)",
825                                 "XGBoost")) +
826   theme(legend.position = "right") +
827   theme_minimal()

```

```

828 ggsave(paste0(PATH_IMAGE_DIR,"/results_sens.png"))
829
830 measure <- "spec"
831 meas_name <- "Specificity"
832 subset_df <- data.frame(x = performance(pred_subset_lr, measure)@x.values[[1]],
833                           y = performance(pred_subset_lr, measure)@y.values[[1]])
834 elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)@x.values[[1]],
835                           y = performance(pred_elastic_lr, measure)@y.values[[1]])
836 pca_df <- data.frame(x = performance(pred_pca_lr, measure)@x.values[[1]],
837                        y = performance(pred_pca_lr, measure)@y.values[[1]])
838 xgb_df <- data.frame(x = performance(pred_xgb, measure)@x.values[[1]],
839                        y = performance(pred_xgb, measure)@y.values[[1]])
840
841 ggplot() +
842   geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
843   geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
844   geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
845   geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
846   labs(color = "Model") +
847   xlab("Threshold") +
848   ylab(meas_name) +
849   scale_color_manual(values = cols,
850                      labels = c("Log Reg (Feat. Sel.)",
851                                 "Log Reg (Elastic)",
852                                 "Log Reg (PCA)",
853                                 "XGBoost")) +
854   theme(legend.position = "right") +
855   theme_minimal()
856 ggsave(paste0(PATH_IMAGE_DIR,"/results_spec.png"))
857
858 measure <- "fpr"
859 meas_name <- "FPR"
860 subset_df <- data.frame(x = performance(pred_subset_lr, measure)@x.values[[1]],
861                           y = performance(pred_subset_lr, measure)@y.values[[1]])
862 elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)@x.values[[1]],
863                           y = performance(pred_elastic_lr, measure)@y.values[[1]])
864 pca_df <- data.frame(x = performance(pred_pca_lr, measure)@x.values[[1]],
865                        y = performance(pred_pca_lr, measure)@y.values[[1]])
866 xgb_df <- data.frame(x = performance(pred_xgb, measure)@x.values[[1]],
867                        y = performance(pred_xgb, measure)@y.values[[1]])
868
869 ggplot() +
870   geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
871   geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
872   geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
873   geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
874   labs(color = "Model") +
875   xlab("Threshold") +
876   ylab(meas_name) +
877   scale_color_manual(values = cols,
878                      labels = c("Log Reg (Feat. Sel.)",
879                                 "Log Reg (Elastic)",
880                                 "Log Reg (PCA)",
881                                 "XGBoost")) +
882   theme(legend.position = "right") +
883   theme_minimal()
884 ggsave(paste0(PATH_IMAGE_DIR,"/results_fpr.png"))
885
886 subset_df <- data.frame(x = performance(pred_subset_lr, "sens", "fpr")@x.values[[1]],
887                           y = performance(pred_subset_lr, "sens", "fpr")@y.values[[1]])
888 elastic_df <- data.frame(x = performance(pred_elastic_lr, "sens", "fpr")@x.values[[1]],
889                           y = performance(pred_elastic_lr, "sens", "fpr")@y.values[[1]])
890 pca_df <- data.frame(x = performance(pred_pca_lr, "sens", "fpr")@x.values[[1]],
891                        y = performance(pred_pca_lr, "sens", "fpr")@y.values[[1]])
892 xgb_df <- data.frame(x = performance(pred_xgb, "sens", "fpr")@x.values[[1]],
893                        y = performance(pred_xgb, "sens", "fpr")@y.values[[1]])
894
895 ggplot() +
896   geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
897   geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
898   geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
899   geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
900   labs(color = "Model") +
901   xlab("1 - Specificity (FPR)") +
902   ylab("Sensitivity (TPR)") +
903   scale_color_manual(values = cols,

```

```

904     labels  = c("Log Reg (Feat. Sel.)",
905                 "Log Reg (Elastic)",
906                 "Log Reg (PCA)",
907                 "XGBoost")) +
908     geom_abline(intercept = 0, slope = 1, linetype = "dotted", color = "black", size = 0.5) +
909     theme(legend.position = "right") +
910     theme_minimal()
911 ggsave(paste0(PATH_IMAGE_DIR,"/results_roc.png"))
912
913
914
915 roc_subset_lr <- roc(test_data$TenYearCHD, result_subset_lr)
916 auc_subset_lr <- roc_subset_lr$auc
917 cat("ROC-AUC from Log Reg w/ Feature Selection: ", auc_subset_lr, "\n")
918
919 roc_elastic_lr <- roc(test_data$TenYearCHD, result_elastic_lr)
920 auc_elastic_lr <- roc_elastic_lr$auc
921 cat("ROC-AUC from Log Reg w/ ElasticNet: ", auc_elastic_lr, "\n")
922
923 roc_pca_lr <- roc(test_data$TenYearCHD, result_pca_lr)
924 auc_pca_lr <- roc_pca_lr$auc
925 cat("ROC-AUC from Log Reg w/ PCA: ", auc_pca_lr, "\n")
926
927 roc_xgb <- roc(as.numeric(testy), result_xgb)
928 auc_xgb <- roc_xgb$auc
929 cat("ROC-AUC from XGBoost: ", auc_xgb, "\n")

```