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*.

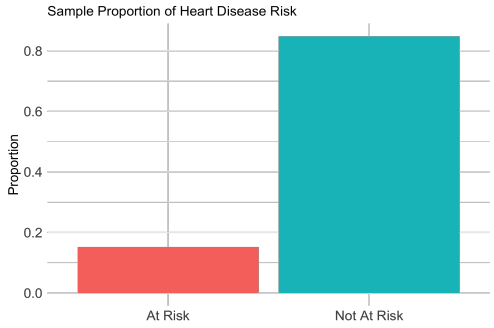
Data

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

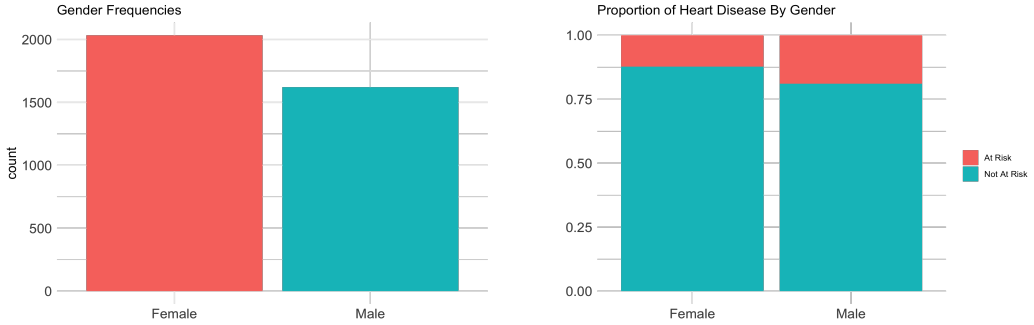
Ten Year Risk of Coronary Heart Disease

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.



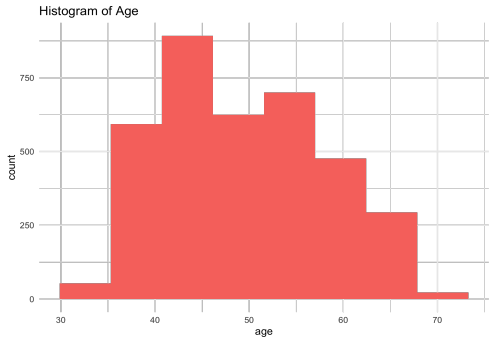
Sex

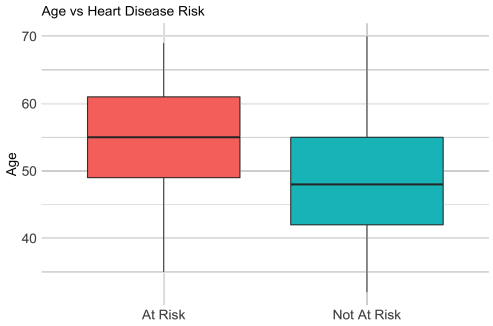
In our study, 55% of the patients are female and 45% of patients are male. According to the two-way proportions plot below, we see that males are slightly more likely to have heart disease compared to the females in this study.



Age

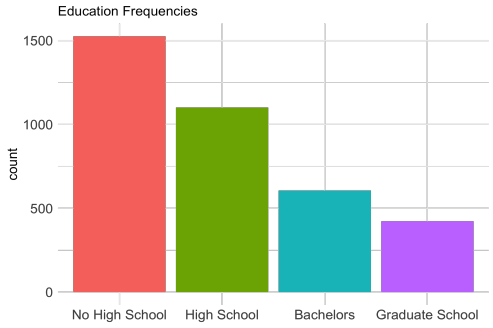
The most common age groups in the data are patients in their 40's and 50's. We can see that older patients' are generally associated with more heart disease risk.

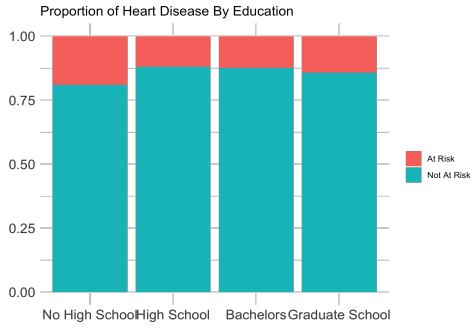




Education

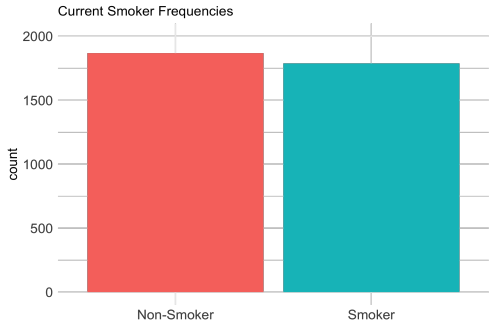
According to the plot below, nearly half of the patients do not have a high school education while one-third of patients have a bachelor's degree or a graduate school degree. The risk of 10-year coronary heart disease among patients with at least a high school education appears less than the proportion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

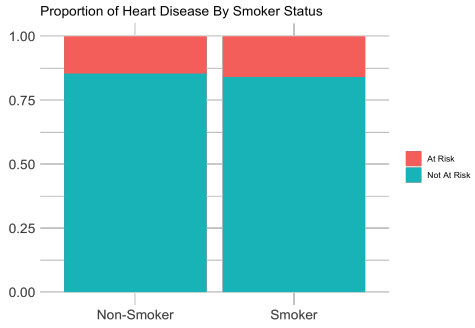




Smoker Status

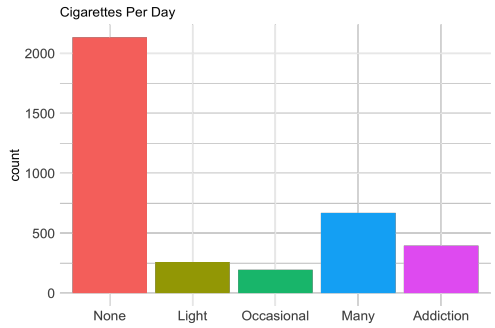
In our dataset, the number of smokers is nearly equivalent to non-smokers. Surprisingly, the risk of 10-year coronary heart disease across both groups is approximately equal. Typically, we would expect smokers to have a slightly larger risk of 10-year coronary heart disease.

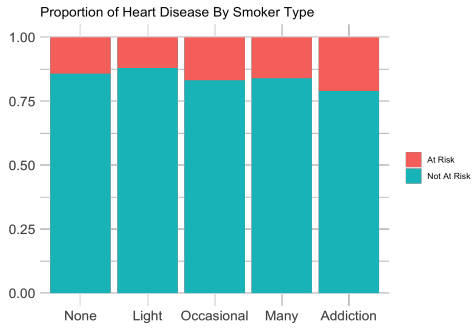




Cigarettes Smoked Per Day

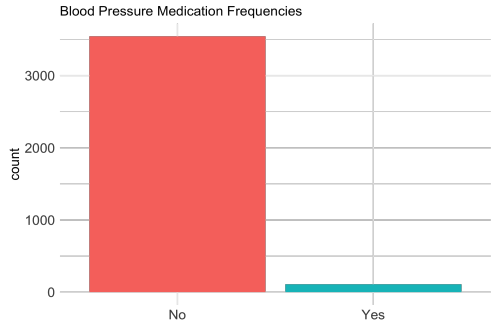
As mentioned above, nearly half our dataset consists of non-smokers. But, for those who do smoke, we see that most patients either smoke many cigarettes per day or have a cigarette addiction. As the smoker type increases from light to addiction, we see the risk of 10-year coronary heart disease increases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

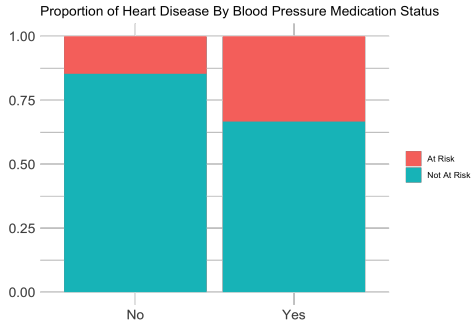




Blood Pressure Medication

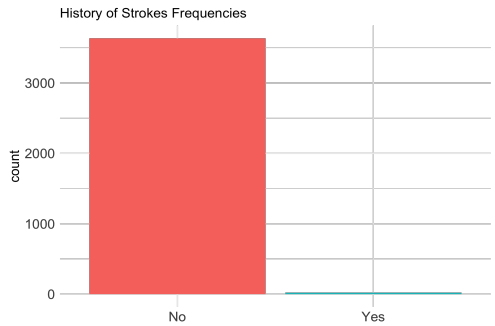
The majority of patients in the study are not on blood pressure medication. For patients on blood pressure medication, over 25% of patients are at risk for 10-year coronary heart disease.

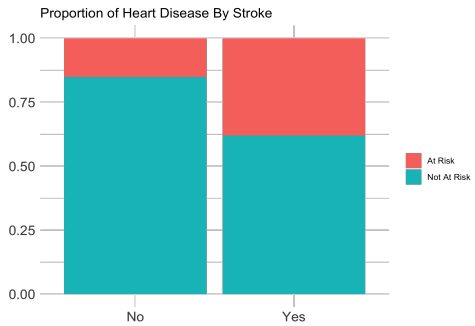




Prevalent Stroke

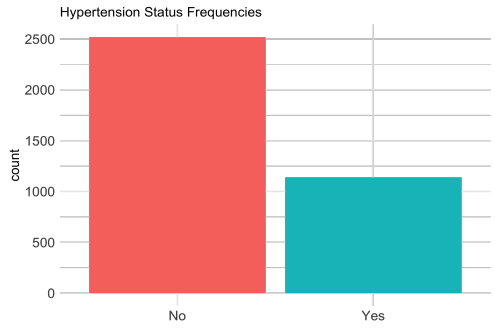
Similarly to blood pressure medication, the majority of patients do not have a history of stroke. As expected, the proportion of patients at risk for 10-year coronary heart disease is greater among patients who have had a stroke. In fact, if a patient in our study has had a stroke, they are twice as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

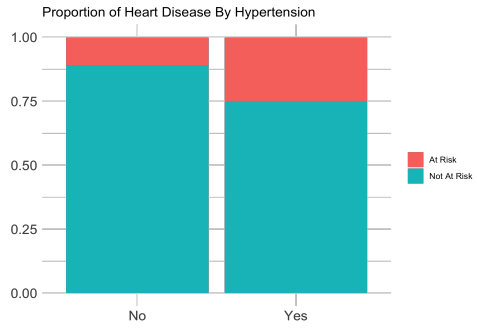




Prevalent Hypertension

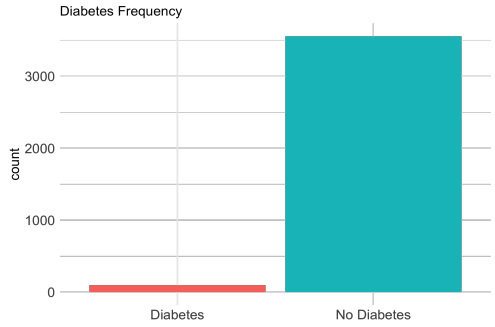
Nearly one-third of patients in the study do not have hypertension. Similarly to prevalent strokes, if a patient in our study has hypertension, they are twice as likely to be at risk for 10-year coronary heart disease.

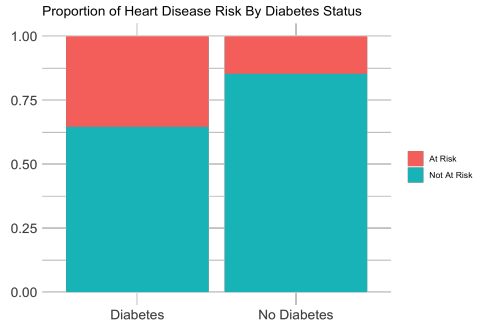




Diabetes

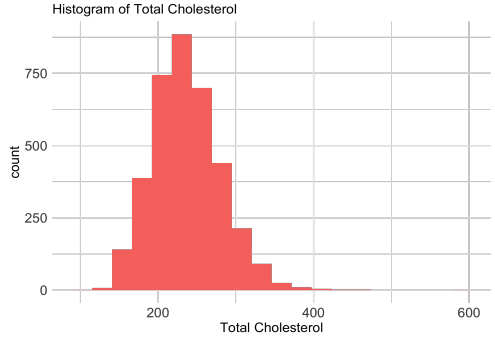
The majority of patients do not have diabetes, and yet, approximately 15% of these patients are at risk for 10-year coronary heart disease. On the other hand, over 30% of patients with diabetes are at risk for 10-year coronary heart disease.

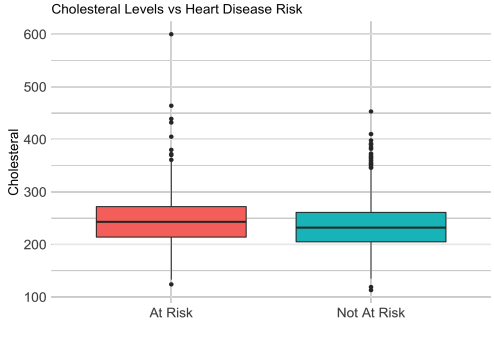




Total Cholesterol

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.

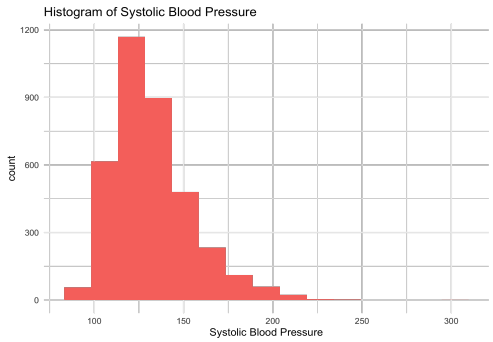


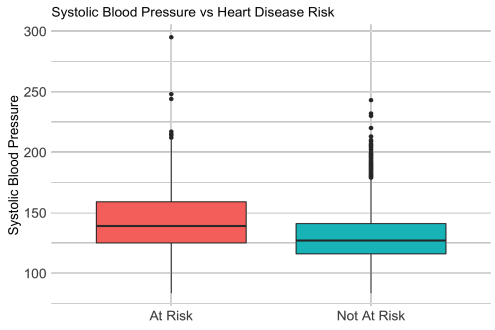


Systolic Blood Pressure

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

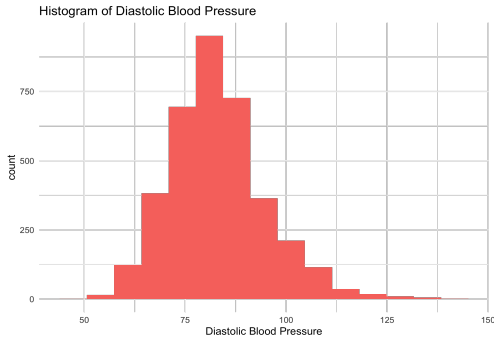
disease.

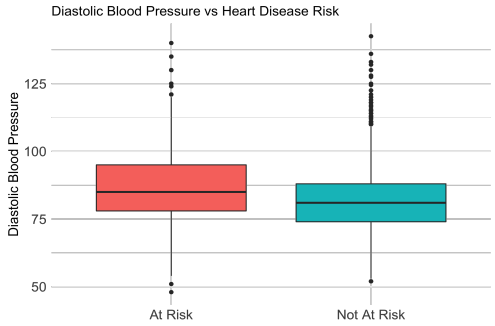




Diastolic Blood Pressure

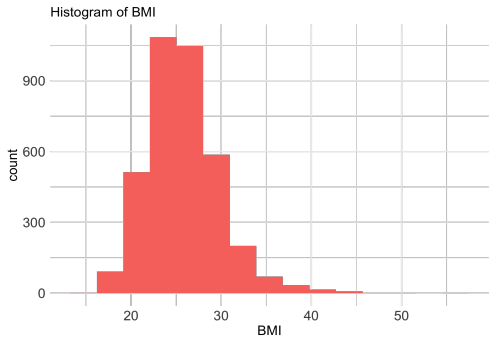
As expected, the distribution of diastolic blood pressure also appears approximately normal, centered around 85 mmHg. Notice that the trend continues as before - an increased level of the diastolic blood pressure is associated with a larger proportion of patients at risk for 10-year coronary heart disease.

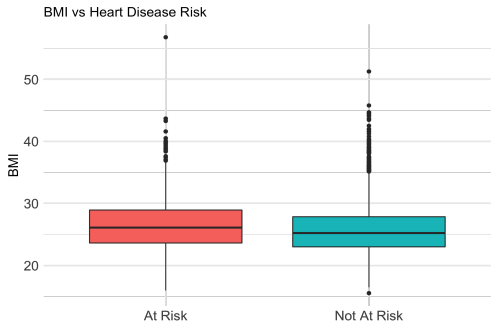




Body Mass Index

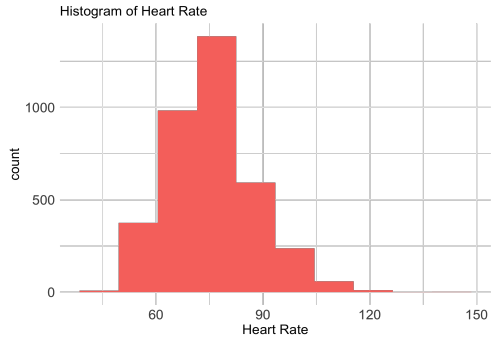
The majority of patients in our study are either at a healthy weight or overweight. Very few patients are underweight in the dataset. Despite the unequal number of patients per group, the proportion of patients at risk for 10-year coronary heart disease is approximately equal across groups.

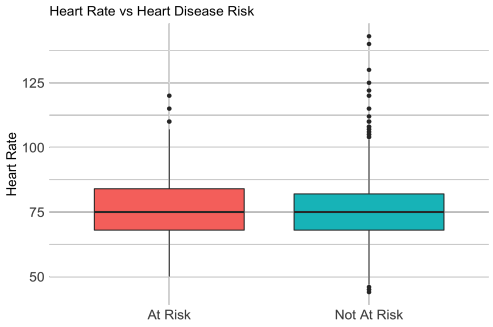




Heart Rate

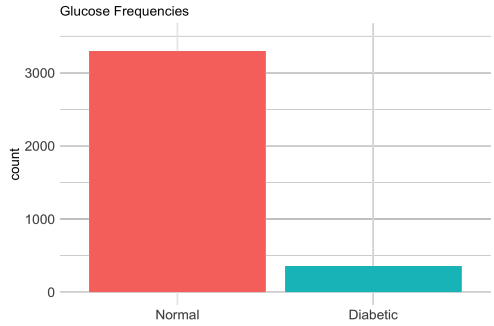
According to the histogram below, patients' heart rates are approximately normally distributed. The average patient heart rate is 75 bpm and the standard deviation is 12 bpm. The distribution of patients' heart rates at risk for 10-year coronary heart disease is nearly identical to the distribution of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

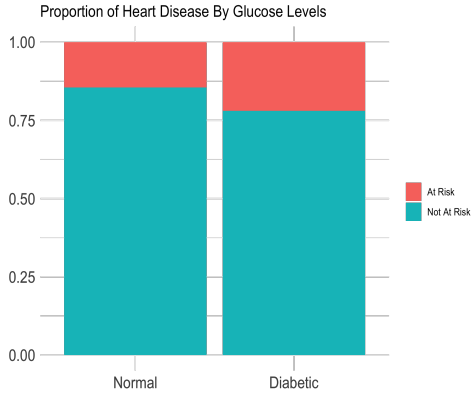




Glucose

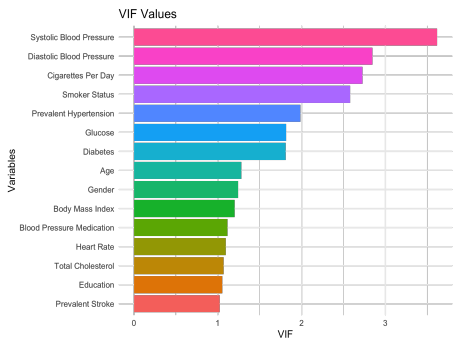
Very few patients in the study have a diabetic glucose level. The majority of patients fall in the normal glucose range. However, a little over 15% of patients in the normal glucose range are at risk for 10-year coronary heart disease. Nearly 25% of patients in the diabetic glucose level are at risk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



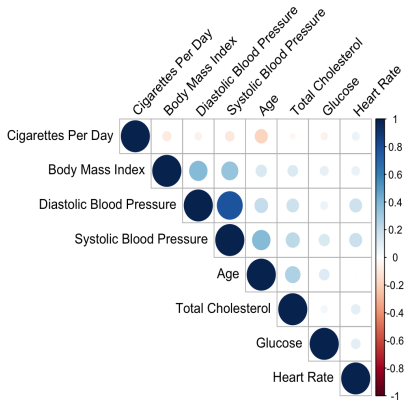


Multicollinearity

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.

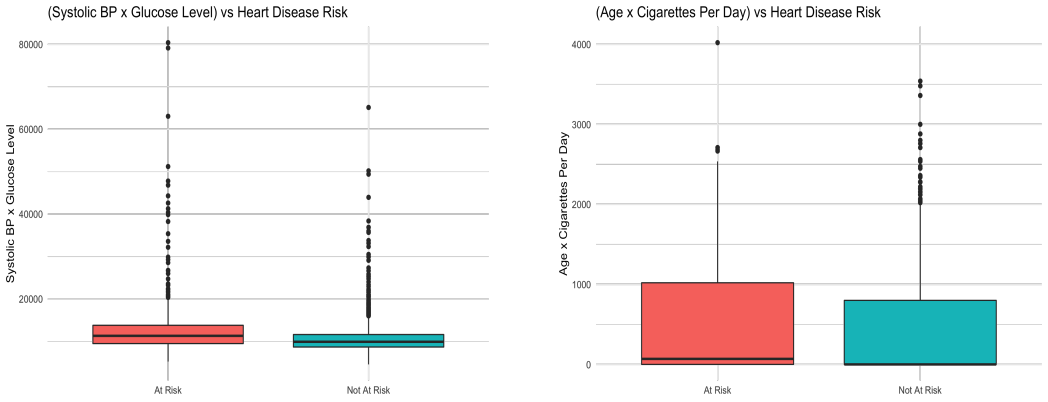
Feature Engineering

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,

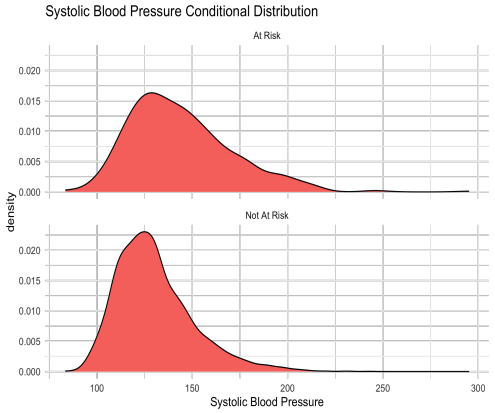
$$\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)}$$

is a quadratic function of X such that,

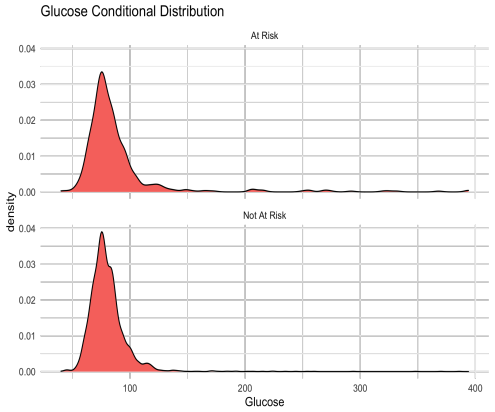
$$\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = \beta_0 + \beta_1 x + \beta_2 x^2$$

for constants β_0, β_1, and $\beta_2 \in \mathbb{R}$ ([**kay**] [**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.



(a)



(b)

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:

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

where $l(x_i, y_i; \boldsymbol{\beta})$ is the log-likelihood contribution for the ith 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

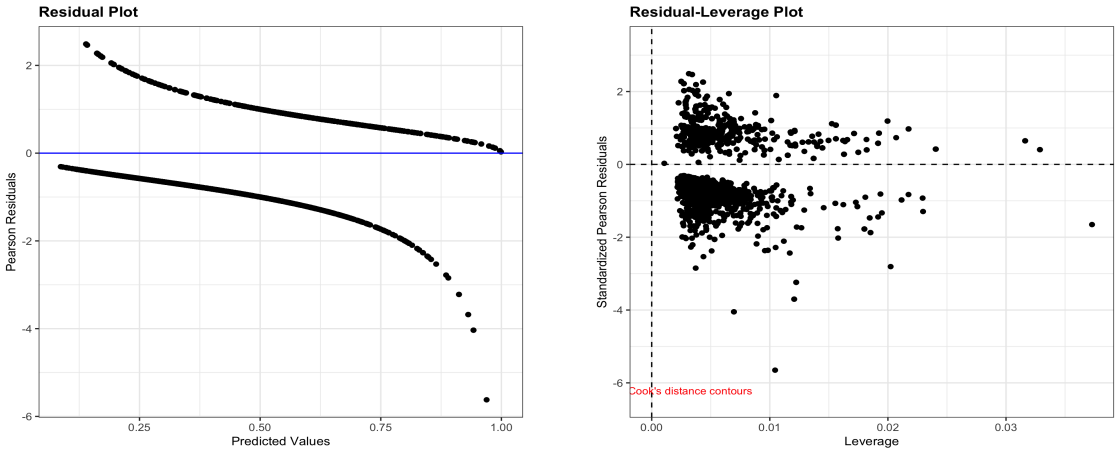
Using backwards elimination, the logistic regression model went from the original set of 23 predictors to 13 predictors. Most of the remaining predictor variables were statistically significant at the $\alpha = 0.05$ level. The backwards elimination dropped the AIC score from 2851.2 with the full model 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.

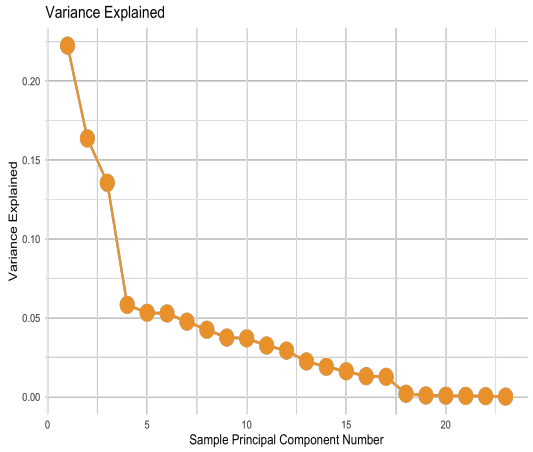
The remaining coefficients $\beta_1, \beta_2, ..., \beta_{13}$ for this logistic regression model are presented in the table below. Since we centered and scaled our data before fitting the logistic regression model, each coefficient from the table below can be interpreted very easily. Recall that for 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*.

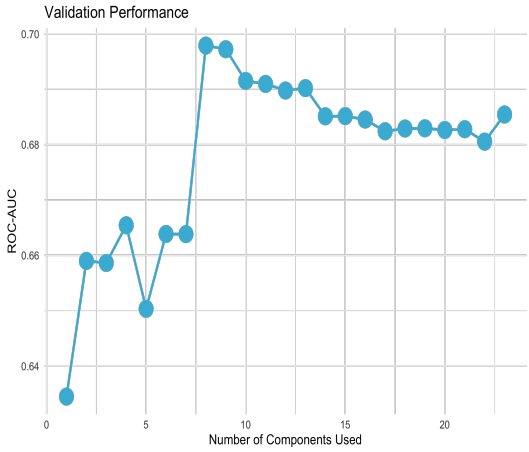
We can exponentiate the coefficients to obtain an equivalent interpretation in terms of the odds. For example, we can see from the table below that one standard deviation increase in age is associated with a 6% ($e^{\hat{\beta}_2} = 1.06$) increase in the odds of a patient having a 10 year risk of heart 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 (Bachelors)	-0.452	0.641	≈ 0
Eucation (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.

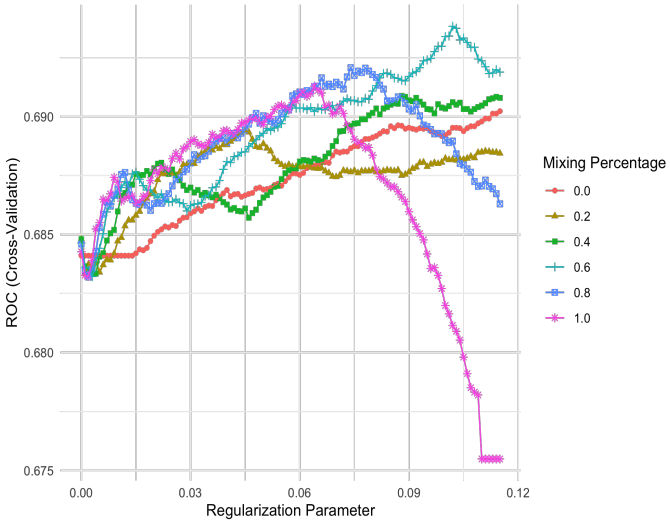
Consequently, using the first 8 principal components reduced the dimension of the feature space from 23 to 8. We found that 6 out of the 8 components were significant at the $\alpha = 0.05$ level. We used residual deviance to test our model against the null model ($H_0 : \beta_i = 0$ for all i). Again, 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 form 23 to 6, and the only remaining predictor variables with non-zero coefficients were: Gender, Age, Systolic Blood Pressure, (Systolic Blood Pressure)^2, (Age$\cdot$Cigarettes Per Day), and (Systolic Blood Pressure $\cdot$ 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.

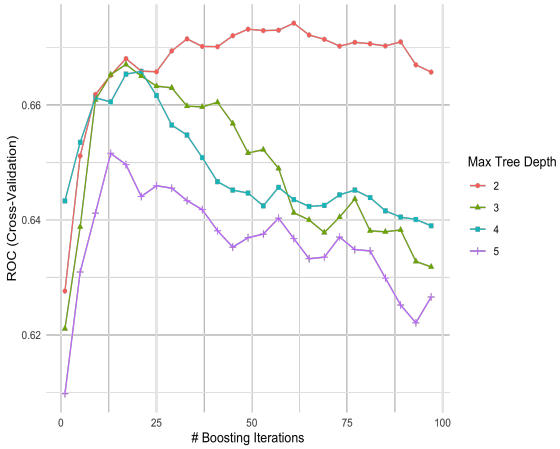
XGBoost

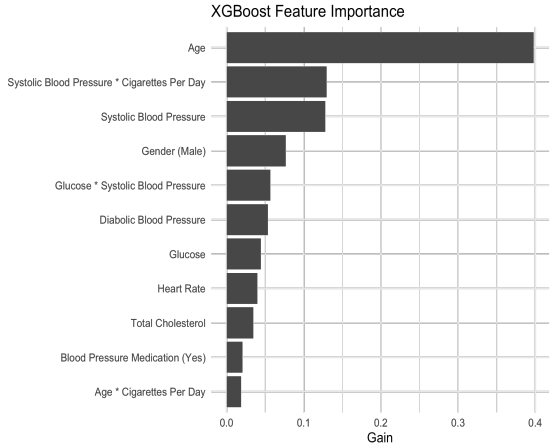
Although this ensemble based model is very different from the previous three models, the approach for training, tuning, and validation remained similar. We used the associated ROC-AUC scores on the validation set to tune the depth of the trees and the number of trees to use in the XGBoost model. Similar 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

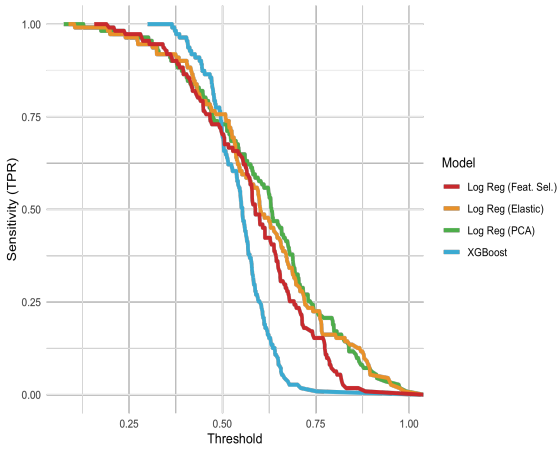
significantly more importance on Age than it does on any other predictor. In fact, the predictor variables Age, Systolic Blood Pressure, and (Systolic Blood Pressure · Cigarettes Per Day) all demonstrated significant importance in our XGBoost model. Notice that these results seem to agree with our elastic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

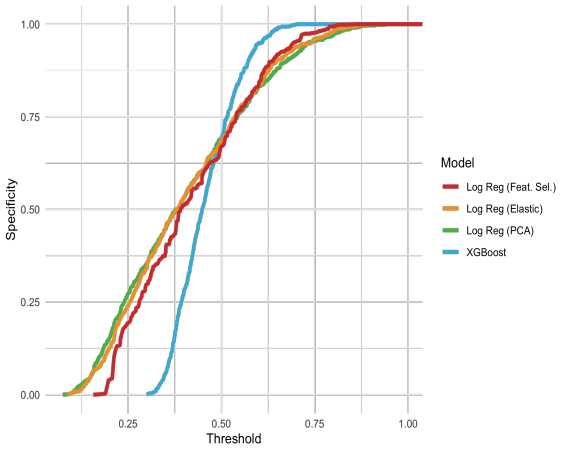




Model Evaluation and Comparison

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

		Predicted		
		No Risk	**Risk**	**Total**
Actual	**No Risk**	377	243	620
	Risk	22	89	111
	Total	399	332	

(a) Log. Reg. w/ Feature Sel.

		Predicted		
		No Risk	**Risk**	**Total**
Actual	**No Risk**	410	210	620
	Risk	25	86	111
	Total	435	296	

(b) Log. Reg. w/ Elastic Net

		Predicted		
		No Risk	**Risk**	**Total**
Actual	**No Risk**	434	186	620
	Risk	29	82	111
	Total	463	268	

(c) Log. Reg. w/ PCA

		Predicted		
		No Risk	**Risk**	**Total**
Actual	**No Risk**	417	203	620
	Risk	33	78	111
	Total	450	281	

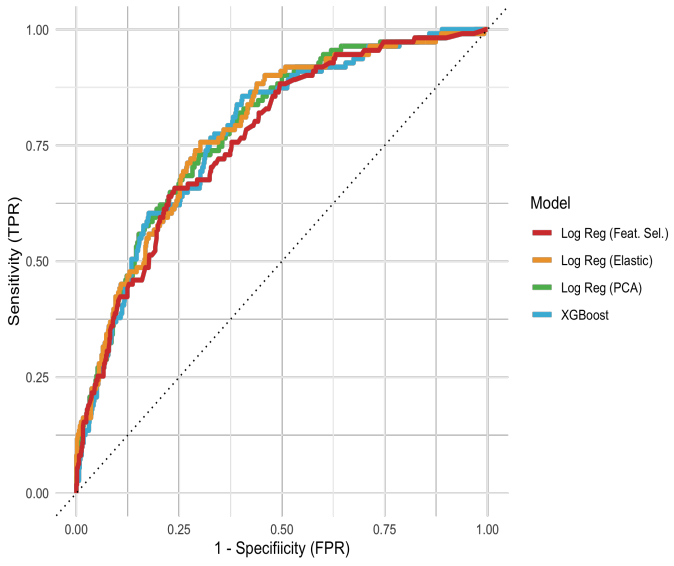
(d) XGBoost

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 XBGoost'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:

```

# Set Environment Variables For Input/Output Paths

## define input path
PATH_MAIN_DATA   <- Sys.getenv("STATS_412_PROJECT_DATA")

## define output path
PATH_IMAGE_DIR  <- Sys.getenv("STATS_412_PROJECT_IMAGE_DIR")


# Load the necessary libraries and read the dataset


library(tidyverse)

rawdf <- read.csv(paste0(PATH_MAIN_DATA,"/framingham.csv"))

df <- rawdf %>%
  drop_na()

# proportion missing
1 - nrow(df) / nrow(rawdf)



# Summarize the features in the Framingham data


summary(df)



# Exploratory Data Analysis

## Response


df$TenYearCHD[df$TenYearCHD == 0] <- "Not At Risk"
df$TenYearCHD[df$TenYearCHD == 1] <- "At Risk"
df$TenYearCHD <- factor(df$TenYearCHD)

table(df$TenYearCHD)

df %>%
  group_by(TenYearCHD) %>%
  summarise(Proportion = n() / nrow(df)) %>%
  ggplot(aes(x = TenYearCHD, y = Proportion, fill = TenYearCHD)) +
  geom_col() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Sample Proportion of Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_chd.png"))



## Education Level


df$education[df$education == 1] <- "No High School"
df$education[df$education == 2] <- "High School"
df$education[df$education == 3] <- "Bachelors"
df$education[df$education == 4] <- "Graduate School"

education_levels <- c("No High School", "High School", "Bachelors", "Graduate School")
df$education <- factor(df$education, levels = education_levels)
```

```r

table(df$education)

ggplot(df, aes(x = education, fill = education)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Education Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_education.png"))



## Current Smoker


df$currentSmoker[df$currentSmoker == 0] <- "Non-Smoker"
df$currentSmoker[df$currentSmoker == 1] <- "Smoker"
df$currentSmoker <- factor(df$currentSmoker)

table(df$currentSmoker)
ggplot(df, aes(x = currentSmoker, fill = currentSmoker)) +
  geom_bar() +
  ylim(0, 2000) +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Current Smoker Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_smoker_status.png"))



## Gender Frequency


df$male[df$male == 1] <- "Male"
df$male[df$male == 0] <- "Female"

table(df$male)
ggplot(df, aes(x = factor(male), fill = factor(male))) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Gender Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_sex.png"))



## Cigs Per Day


df$cigsPerDay <- cut(df$cigsPerDay, c(-1, 5, 10, 15, 20, 90))
levels(df$cigsPerDay) <- c("None", "Light", "Occasional", "Many", "Addiction")

table(df$cigsPerDay)
ggplot(df, aes(x = cigsPerDay, fill = cigsPerDay)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
```

```
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Cigarettes Per Day")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_cigs_per_day.png"))



## BP Meds


df$BPMeds[df$BPMeds == 0] <- "No"
df$BPMeds[df$BPMeds == 1] <- "Yes"
df$BPMeds <- factor(df$BPMeds)

table(df$BPMeds)

ggplot(df, aes(x = BPMeds, fill = BPMeds)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Blood Pressure Medication Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_bp_meds.png"))



## Prevalent Strokes


df$prevalentStroke[df$prevalentStroke == 0] <- "No"
df$prevalentStroke[df$prevalentStroke == 1] <- "Yes"
df$prevalentStroke <- factor(df$prevalentStroke)

table(df$prevalentStroke)

ggplot(df, aes(x = prevalentStroke, fill = prevalentStroke)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("History of Strokes Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_strokes.png"))



## Hypertension


df$prevalentHyp[df$prevalentHyp == 0] <- "No"
df$prevalentHyp[df$prevalentHyp == 1] <- "Yes"
df$prevalentHyp <- factor(df$prevalentHyp)

table(df$prevalentHyp)

ggplot(df, aes(x = prevalentHyp, fill = prevalentHyp)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
```

```r
  ggtitle("Hypertension Status Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_hypertension.png"))



## DiaBP


df$diaBP <- as.numeric(df$diaBP)
ggplot(df, aes(x = diaBP, fill = "#F8766D")) +
  geom_histogram(bins = 15) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Diastolic Blood Pressure") +
  ggtitle("Histogram of Diastolic Blood Pressure")

ggsave(paste0(PATH_IMAGE_DIR,"/hist_dbp.png"))

df$diaBP <- cut(df$diaBP, c(0, 80, 89, max(df$diaBP)))
levels(df$diaBP) <- c('Average', 'Above Average', 'High')
table(df$diaBP)

ggplot(df, aes(x = diaBP, fill = diaBP)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Diastolic Blood Pressure Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_dbp.png"))



## Glucose


ggplot(df, aes(x = glucose, fill = "#F8766D")) +
  geom_histogram(bins = 20) +
  theme_minimal() +
  theme(legend.position = "None") +
  ggtitle("Histogram of Glucose")

ggsave(paste0(PATH_IMAGE_DIR,"/hist_glucose.png"))

df$glucose <- cut(df$glucose, c(0, 98, max(df$glucose)))
levels(df$glucose) <- c('Normal', 'Diabetic')

ggplot(df, aes(x = glucose, fill = glucose)) +
  geom_bar() +
  ylim(0, 3500) +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Glucose Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_glucose.png"))



## Age


ggplot(df, aes(x = age, fill = "#F8766D")) +
  geom_histogram(bins = 8) +
  theme_minimal() +
  theme(legend.position = "None") +
  ggtitle("Histogram of Age")
```

```r

ggsave(paste0(PATH_IMAGE_DIR,"/hist_age.png"))

df$age <- cut(df$age, c(30, 40, 50, 60, 70))
levels(df$age) <- c("30's", "40's", "50's", "Over 60")

table(df$age)

df %>%
  ggplot(aes(x = age, fill = age)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Age Group Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_age.png"))



## SysBP


df$sysBP <- as.numeric(df$sysBP)
ggplot(df, aes(x = sysBP, fill = "#F8766D")) +
  geom_histogram(bins = 15) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Systolic Blood Pressure") +
  ggtitle("Histogram of Systolic Blood Pressure")

ggsave(paste0(PATH_IMAGE_DIR,"/hist_sbp.png"))

df$sysBP <- cut(df$sysBP, c(0, 120, 139, max(df$sysBP)))
levels(df$sysBP) <- c('Average', 'Above Average', 'High')
table(df$sysBP)

ggplot(df, aes(x = sysBP, fill = sysBP)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Systolic Blood Pressure Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_sbp.png"))



## Heart Rate


ggplot(df, aes(x = heartRate, fill = "#F8766D")) +
  geom_histogram(bins = 10) +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  ggtitle("Histogram of Heart Rate") +
  xlab("Heart Rate")

ggsave(paste0(PATH_IMAGE_DIR,"/hist_heart_rate.png"))



## BMI

```

```r

ggplot(df, aes(x = BMI, fill = "#F8766D")) +
  geom_histogram(bins = 15) +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  ggtitle("Histogram of BMI")

ggsave(paste0(PATH_IMAGE_DIR,"/hist_bmi.png"))

df$BMI <- cut(df$BMI, c(0, 18.5, 25, 30, max(df$BMI)))
levels(df$BMI) <- c('Under Weight', 'Healthy Weight', 'Over Weight', 'Obese')

ggplot(df, aes(x = BMI, fill = BMI)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("BMI Frequencies")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_bmi.png"))



## Diabetes Status


df$diabetes[df$diabetes == 0] <- "No Diabetes"
df$diabetes[df$diabetes == 1] <- "Diabetes"
df$diabetes <- factor(df$diabetes)

ggplot(df, aes(x = diabetes, fill = diabetes)) +
  geom_bar() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ggtitle("Diabetes Frequency")

ggsave(paste0(PATH_IMAGE_DIR,"/bar_diabetes.png"))



## Cholesterol

ggplot(df, aes(x = totChol, fill = "#F8766D")) +
  geom_histogram(bins = 20) +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("Total Cholesterol") +
  ggtitle("Histogram of Total Cholesterol")

ggsave(paste0(PATH_IMAGE_DIR,"/hist_cholesterol.png"))



# Two Way Relationships

## Numeric Variables Boxplots


df2 <- read.csv(paste0(PATH_MAIN_DATA, "/framingham.csv")) %>%
```

```r
  drop_na()

df2$TenYearCHD[df2$TenYearCHD == 0] <- "Not At Risk"
df2$TenYearCHD[df2$TenYearCHD == 1] <- "At Risk"
df2$TenYearCHD <- factor(df2$TenYearCHD)

ggplot(df2, aes(x = TenYearCHD, y = diaBP, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Diastolic Blood Pressure") +
  ggtitle("Diastolic Blood Pressure vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_dbp.png"))

ggplot(df2, aes(x = TenYearCHD, y = totChol, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Cholesteral") +
  ggtitle("Cholesteral Levels vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_cholesterol.png"))

ggplot(df2, aes(x = TenYearCHD, y = age, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Age") +
  ggtitle("Age vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_age.png"))

ggplot(df2, aes(x = TenYearCHD, y = glucose, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Glucose") +
  ggtitle("Glucose vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_glucose.png"))

ggplot(df2, aes(x = TenYearCHD, y = sysBP, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Systolic Blood Pressure") +
  ggtitle("Systolic Blood Pressure vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_sbp.png"))
```

```r
ggplot(df2, aes(x = TenYearCHD, y = cigsPerDay, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Cigarettes Per Day") +
  ggtitle("Cigarettes Per Day vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_cigs_per_day.png"))

ggplot(df2, aes(x = TenYearCHD, y = BMI, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("BMI") +
  ggtitle("BMI vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_bmi.png"))

ggplot(df2, aes(x = TenYearCHD, y = heartRate, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Heart Rate") +
  ggtitle("Heart Rate vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_heart_rate.png"))



## Conditional Distributions


ggplot(df2, aes(x = cigsPerDay, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Cigarettes Per Day") +
  ggtitle("Cigarettes Per Day Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_cigs_per_day.png"))

ggplot(df2, aes(x = glucose, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Glucose") +
  ggtitle("Glucose Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_glucose.png"))

ggplot(df2, aes(x = totChol, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Cholesterol") +
  ggtitle("Cholesterol Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_cholesterol.png"))
```

```r

ggplot(df2, aes(x = age, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Age") +
  ggtitle("Age Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_age.png"))

ggplot(df2, aes(x = sysBP, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Systolic Blood Pressure") +
  ggtitle("Systolic Blood Pressure Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_sbp.png"))

ggplot(df2, aes(x = BMI, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("BMI") +
  ggtitle("BMI Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_bmi.png"))

ggplot(df2, aes(x = diaBP, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Diastolic Blood Pressure") +
  ggtitle("Diastolic Blood Pressure Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_dbp.png"))

ggplot(df2, aes(x = heartRate, fill = "#F8766D")) +
  geom_density() +
  facet_wrap(~ TenYearCHD, nrow = 2) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("Heart Rate") +
  ggtitle("Heart Rate Conditional Distribution")

ggsave(paste0(PATH_IMAGE_DIR,"/dens_heart_rate.png"))

df2 %>%
  group_by(TenYearCHD) %>%
  summarise(age_SD = sd(age),
            cigsPerDay_SD = sd(cigsPerDay),
            totChol_SD = sd(totChol),
            sysBP_SD = sd(sysBP),
            diaBP_SD = sd(diaBP),
            BMI_SD = sd(BMI),
            heartRate_SD = sd(heartRate),
            glucose_SD = sd(glucose))



## Interactions between Strictly Numeric Variables


wilcox.test(sysBP*cigsPerDay ~ TenYearCHD, df2)

ggplot(df2, aes(x = TenYearCHD, y = cigsPerDay * sysBP, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Systolic BP x Cigarettes Per Day") +
  ggtitle("(Systolic BP x Cigarettes Per Day) vs Heart Disease Risk")
```

```r

ggsave(paste0(PATH_IMAGE_DIR,"/box_int_sbp_cigs_per_day.png"))

wilcox.test(age * cigsPerDay ~ TenYearCHD, df2)

ggplot(df2, aes(x = TenYearCHD, y = cigsPerDay * age, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Age x Cigarettes Per Day") +
  ggtitle("(Age x Cigarettes Per Day) vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_int_age_cigs_per_day.png"))

wilcox.test(sysBP * glucose ~ TenYearCHD, df2)

ggplot(df2, aes(x = TenYearCHD, y = glucose * sysBP, fill = TenYearCHD)) +
  geom_boxplot() +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("") +
  ylab("Systolic BP x Glucose Level") +
  ggtitle("(Systolic BP x Glucose Level) vs Heart Disease Risk")

ggsave(paste0(PATH_IMAGE_DIR,"/box_int_sbp_glucose.png"))



## Diastolic Blood Pressure Level


table(df$TenYearCHD, df$diaBP)

df %>%
  group_by(TenYearCHD, diaBP) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = diaBP, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease Risk By Diastolic Blood Pressure Level")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_dbp.png"))



## Diabetes Status


table(df$TenYearCHD, df$diabetes)

df %>%
  group_by(TenYearCHD, diabetes) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = diabetes, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease Risk By Diabetes Status")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_diabetes.png"))


```

```r

## Age


table(df$TenYearCHD, df$age)

df %>%
  group_by(TenYearCHD, age) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = age, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease Risk By Age")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_age.png"))



## Education Level


table(df$TenYearCHD, df$education)

df %>%
  group_by(TenYearCHD, education) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = education, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Education")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_education.png"))



## Gender


table(df$TenYearCHD, df$male)

df %>%
  group_by(TenYearCHD, male) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = male, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Gender")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_sex.png"))



## Smoker Status


```

```r
table(df$TenYearCHD, df$currentSmoker)

df %>%
  group_by(TenYearCHD, currentSmoker) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = currentSmoker, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Smoker Status")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_smoker_status.png"))


## Cigs Per Day

table(df$TenYearCHD, df$cigsPerDay)

df %>%
  group_by(TenYearCHD, cigsPerDay) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = cigsPerDay, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Smoker Type")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_cigs_per_day.png"))


## Glucose Level

table(df$TenYearCHD, df$glucose)

df %>%
  group_by(TenYearCHD, glucose) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = glucose, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Glucose Levels")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_glucose.png"))


## BP Meds


table(df$TenYearCHD, df$BPMeds)

df %>%
  group_by(TenYearCHD, BPMeds) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = BPMeds, y = proportion, fill = TenYearCHD)) +
```

```
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Blood Pressure Medication Status")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_bp_meds.png"))



## Systolic Blood Pressure Level


table(df$TenYearCHD, df$sysBP)

df %>%
  group_by(TenYearCHD, sysBP) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = sysBP, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease Risk By Systolic Blood Pressure Level")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_sbp.png"))



## BMI


table(df$TenYearCHD, df$BMI)

df %>%
  group_by(TenYearCHD, BMI) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = BMI, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Body Mass Index")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_bmi.png"))



## Hypertension

table(df$TenYearCHD, df$prevalentHyp)

df %>%
  group_by(TenYearCHD, prevalentHyp) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = prevalentHyp, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
```

```
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Hypertension")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_hypertension.png"))



## Prevalent Stroke

table(df$TenYearCHD, df$prevalentStroke)

df %>%
  group_by(TenYearCHD, prevalentStroke) %>%
  summarise(proportion = n() / nrow(df)) %>%
  ggplot(aes(x = prevalentStroke, y = proportion, fill = TenYearCHD)) +
  geom_bar(position = "fill", stat = "identity") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text = element_text(size = 14)) +
  theme(legend.title = element_blank()) +
  xlab("") +
  ylab("") +
  ggtitle("Proportion of Heart Disease By Stroke")

ggsave(paste0(PATH_IMAGE_DIR,"/two_way_stroke.png"))



## Categorical Variables Chisq tests


chisq.test(df$TenYearCHD, df$glucose)
chisq.test(df$TenYearCHD, df$currentSmoker)
chisq.test(df$TenYearCHD, df$education)
chisq.test(df$TenYearCHD, df$cigsPerDay)
chisq.test(df$TenYearCHD, df$prevalentHyp)
chisq.test(df$TenYearCHD, df$diaBP)
chisq.test(df$TenYearCHD, df$male)
chisq.test(df$TenYearCHD, df$BMI)
chisq.test(df$TenYearCHD, df$sysBP)
chisq.test(df$TenYearCHD, df$BPMeds)
chisq.test(df$TenYearCHD, df$prevalentStroke)
chisq.test(df$TenYearCHD, df$diabetes)
chisq.test(df$TenYearCHD, df$age)



# VIF


glmodel1 <- glm(TenYearCHD ~ ., data = df2, family = binomial)

data.frame(DAAG::vif(glmodel1)) %>%
  rownames_to_column(var = "var") %>%
  ggplot(aes(x = DAAG..vif.glmodel1., y = fct_reorder(var, DAAG..vif.glmodel1.),
             fill = fct_reorder(var, DAAG..vif.glmodel1.))) +
  geom_col() +
  scale_y_discrete(labels = c("Prevalent Stroke", "Education", "Total Cholesterol", "Heart Rate", "Blood
    Pressure Medication", "Body Mass Index", "Gender", "Age", "Diabetes", "Glucose", "Prevalent
    Hypertension", "Smoker Status", "Cigarettes Per Day", "Diastolic Blood Pressure", "Systolic Blood
    Pressure")) +
  theme_minimal() +
  theme(legend.position = "None") +
  xlab("VIF") +
  ylab("Variables") +
  ggtitle("VIF Values")

ggsave(paste0(PATH_IMAGE_DIR,"/vif.png"))
```

```r
# Correlation Plot


corrplot::corrplot(
  cor(df2 %>% select(diaBP, BMI, cigsPerDay, age, totChol, sysBP, glucose,heartRate) %>%
        rename(`Diastolic Blood Pressure` = diaBP,
               `Body Mass Index` = BMI,
               `Cigarettes Per Day` = cigsPerDay,
               `Age` = age,
               `Total Cholesterol` = totChol,
               `Systolic Blood Pressure` = sysBP,
               `Glucose` = glucose,
               `Heart Rate` = heartRate)),
  type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)

ggsave(paste0(PATH_IMAGE_DIR,"/cor.png"))
```

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

```
# Set Environment Variables For Input/Output Paths


## define input path
PATH_MAIN_DATA   <- Sys.getenv("STATS_412_PROJECT_DATA")

## define output path
PATH_IMAGE_DIR  <- Sys.getenv("STATS_412_PROJECT_IMAGE_DIR")


# Loading the Data & Libraries

library(caTools)
library(ROCR)
library(pROC)
library(dplyr)
library(AMR)
library(ggplot2)
library(car)
library(ggResidpanel)
library(ggfortify)
library(xgboost)
library(caret)
library(glmnet)
library(alr4)
library(ROSE)
library(sjPlot)



# Load original data, add additional variables, make encode dummy variables (treatment), and then scale
    the data for non-PCA models


orig_df <- na.omit(read.csv(paste0(PATH_MAIN_DATA, "/framingham.csv")))
facts <- c('male', 'TenYearCHD', 'education', 'currentSmoker',
           'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes')
orig_df[, facts] <- lapply(orig_df[, facts], factor)

levels(orig_df$education) <- c("no high school", "high school", "bachelors", "graduate school")

# squared terms
orig_df$diaBP.SQ <- (orig_df$diaBP) ^ 2
orig_df$glucose.SQ <- (orig_df$glucose) ^ 2
orig_df$sysBP.SQ <- (orig_df$sysBP) ^ 2

# interactions
orig_df$ageXcigsPerDay <- orig_df$age*orig_df$cigsPerDay
orig_df$glucoseXsysBP <- orig_df$glucose*orig_df$sysBP
orig_df$sysBPXcigsPerDay <- orig_df$sysBP*orig_df$cigsPerDay

# make dummy variables
dummy <- dummyVars(" ~ .", data = orig_df, fullRank = TRUE)
orig_df <- data.frame(predict(dummy, newdata = orig_df))

# set seed for consistency
set.seed(4)

# Split data, then apply ROSE to the training data to bootstrap the training set
train.index <- createDataPartition(orig_df$TenYearCHD.1, p = 0.8, list = F)
holdout_data <- orig_df[train.index[1:ceiling(length(train.index)*0.2)], ]
train_data <- orig_df[train.index[ceiling(length(train.index)*0.2):length(train.index)], ]
train_data <- ovun.sample(TenYearCHD.1 ~ ., data = train_data, seed = 4)$data
test_data <- orig_df[-train.index, ]

# scale the data for model 1 and model 3
train_data_scaled <- train_data
holdout_data_scaled <- holdout_data
test_data_scaled <- test_data


# scale data with no data leakage
normParam <- preProcess(train_data[, -18])
```

```r
train_data_scaled[, -18] <- as.data.frame(predict(normParam, train_data[, -18]))
holdout_data_scaled[, -18] <- as.data.frame(predict(normParam, holdout_data[, -18]))
test_data_scaled[, -18] <- as.data.frame(predict(normParam, test_data[, -18]))



# Load original data, add additional variables, make encode dummy variables (treatment), and then scale the data for PCA


orig_df_PCA <- na.omit(read.csv(paste0(PATH_MAIN_DATA, "/framingham.csv")))
facts <- c('male', 'TenYearCHD', 'education', 'currentSmoker',
           'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes')
orig_df_PCA[, facts] <- lapply(orig_df_PCA[, facts], factor)

# squared terms
orig_df_PCA$diaBP.SQ <- (orig_df_PCA$diaBP) ^ 2
orig_df_PCA$glucose.SQ <- (orig_df_PCA$glucose) ^ 2
orig_df_PCA$sysBP.SQ <- (orig_df_PCA$sysBP) ^ 2

# interactions
orig_df_PCA$ageXcigsPerDay <- orig_df_PCA$age*orig_df_PCA$cigsPerDay
orig_df_PCA$glucoseXsysBP <- orig_df_PCA$glucose*orig_df_PCA$sysBP
orig_df_PCA$sysBPXcigsPerDay <- orig_df_PCA$sysBP*orig_df_PCA$cigsPerDay

# make dummy variables for PCA
dummy <- dummyVars(" ~ .", data = orig_df_PCA, fullRank = TRUE)
orig_df_PCA <- data.frame(predict(dummy, newdata = orig_df_PCA))

# split data
set.seed(4)
train.index_PCA <- createDataPartition(orig_df_PCA$TenYearCHD, p = 0.8, list = F)
holdout_data_PCA <- orig_df_PCA[train.index_PCA[1:ceiling(length(train.index_PCA)*0.2)], ]
train_data_PCA <- orig_df_PCA[train.index_PCA[ceiling(length(train.index_PCA)*0.2):length(train.index_PCA)], ]
test_data_PCA <- orig_df_PCA[-train.index_PCA, ]

# then apply ROSE to the training data to bootstrap the training set
train_data_PCA <- ovun.sample(TenYearCHD.1 ~ ., data = train_data_PCA, seed = 4)$data

# scale data with no data leakage
normParam <- preProcess(train_data_PCA[, -18])
train_data_PCA[, -18] <- predict(normParam, train_data_PCA[, -18])
holdout_data_PCA[, -18] <- predict(normParam, holdout_data_PCA[, -18])

# remove response, apply pca
PCs <- prcomp(train_data_PCA[, -18], scale = FALSE, center = FALSE)
train_data_PCA = cbind(as.data.frame(PCs$x), TenYearCHD = train_data_PCA[, 18])

# adjust new holdout and test data with training loadings (new data leakage)
x <- as.matrix(holdout_data_PCA[, -18])
x <- x %*% PCs$rotation
holdout_data_PCA <- data.frame(x, TenYearCHD = holdout_data_PCA[, 18])



# Logistic Regression Modeling (Using Feature Subset Selection)

## Backward Stepwise Feature Selection


glm_bs1 <- glm(TenYearCHD.1 ~ ., data = train_data_scaled, family = binomial)
summary(glm_bs1)

# use backward selection
glm_bs2 <- step(glm_bs1, direction = "backward", trace = 0)
summary(glm_bs2)

# difference in deviance can be used to compare the backward selection model
# and the original
anova(glm_bs2, glm_bs1, test = "Chisq")



```

```r
## Goodness of fit


subset_LR <- glm(TenYearCHD.1 ~ male.1 + age + education.high.school + education.bachelors +
                   education.graduate.school + cigsPerDay + BPMeds.1 + prevalentHyp.1 +
                   totChol + sysBP + diaBP + diaBP.SQ + glucose.SQ,
                 family = binomial,
                 data = train_data_scaled)

summary(subset_LR)

# test deviance against null model
null_LR <- glm(TenYearCHD.1 ~ 1, data = train_data_scaled, family = binomial)

anova(null_LR, subset_LR, test = "Chisq")

# could equivalently be done
# pchisq(3245.3 - 2808.6, 2340-2327, lower=FALSE)

# chisq goodness of fit test
ts <- sum(residuals(subset_LR, type = "pearson") ^ 2)
cat("P-Value for the Pearson Goodness of Fit Test  = ", pchisq(ts, subset_LR$df.residual, lower=FALSE), "\
    n")



## Leverage, Influence, and Residual Analysis


par(mfrow = c(2, 2))
resid_panel(subset_LR, plots = c("resid", "qq"), theme = "minimal", type = "pearson")
ggsave(paste0(PATH_IMAGE_DIR,"/subset_LR_pearson_residuals.png"))

resid_panel(subset_LR, plots = c("resid", "qq"), theme = "minimal", type = "deviance")
ggsave(paste0(PATH_IMAGE_DIR,"/subset_LR_dev_residuals.png"))

par(mfrow = c(1, 2))
autoplot(subset_LR, which = c(4, 5)) + theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/subset_LR_diagnostics.png"))




## Holdout Set CV


# defines holdout set usage for no data leakage
cv_train_data_scaled <- rbind(holdout_data_scaled, train_data_scaled)

cv_train_data_scaled$TenYearCHD.1 <- factor(cv_train_data_scaled$TenYearCHD.1)
levels(cv_train_data_scaled$TenYearCHD.1) <- c("no", "yes")

fit_on <- list(ts = nrow(holdout_data_scaled):nrow(cv_train_data_scaled))
pred_on <- list(hs = 1:nrow(holdout_data_scaled))

train.control <- trainControl(method = "cv", classProbs = TRUE,
                              index =  fit_on, indexOut = pred_on,
                              summaryFunction = twoClassSummary)

cv_subset_glm <- train(TenYearCHD.1 ~ male.1 + age + education.high.school + education.bachelors +
                         education.graduate.school + cigsPerDay + BPMeds.1 + prevalentHyp.1 +
                         totChol + sysBP + diaBP + diaBP.SQ + glucose.SQ,
                       data = cv_train_data_scaled,
                       method = "glm",
                       family = "binomial",
                       metric = "ROC",
                       trControl = train.control)

print(cv_subset_glm)

levels(cv_train_data_scaled$TenYearCHD.1) <- c(0, 1)



# Logistic Regression (Using Lasso + Ridge = Elastic)
```

```r

## Finding Optimal Regularization Parameter


# defines holdout set usage for no data leakage
cv_train_data_scaled <- rbind(holdout_data_scaled, train_data_scaled)

cv_train_data_scaled$TenYearCHD.1 <- factor(cv_train_data_scaled$TenYearCHD.1)
levels(cv_train_data_scaled$TenYearCHD.1) <- c("no", "yes")

# predictor variables
cv_train_x <- model.matrix(TenYearCHD.1 ~ ., cv_train_data_scaled)[, -1]

# Convert the outcome (class) to a numerical variable
cv_train_y <- cv_train_data_scaled$TenYearCHD.1

fit_on <- list(ts = nrow(holdout_data_scaled):nrow(cv_train_data_scaled))
pred_on <- list(hs = 1:nrow(holdout_data_scaled))

train.control <- trainControl(method = "cv", classProbs = TRUE,
                              index =  fit_on, indexOut = pred_on,
                              summaryFunction = twoClassSummary)

# large grid
regGrid <- expand.grid(alpha = seq(0, 1, by = 0.2), lambda = seq(0, 0.115, by = 0.001))

cv_elastic_glm <- train(cv_train_x, cv_train_y,
                        method = "glmnet",
                        family = "binomial",
                        metric = "ROC",
                        tuneGrid = regGrid,
                        trControl = train.control)

ggplot(cv_elastic_glm) + theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/elastic_LR_tuning.png"))

levels(cv_train_data_scaled$TenYearCHD.1) <- c(0, 1)

# fit "optimal" model on smaller training set
train_x <- model.matrix(TenYearCHD.1 ~ ., train_data_scaled)[, -1]
train_y <- train_data_scaled$TenYearCHD.1

elastic.lr <- glmnet(train_data_scaled[, -18], train_data_scaled[, 18], alpha = 0.6, family = "binomial",
    lambda = 0.098)

# print coefs
coef(elastic.lr)



## Manual Residual Analysis


# formulas from lecture
fittedvals <- predict(elastic.lr, newx = train_x, type = "response")

pearsonResiduals <- (train_y - fittedvals) / sqrt(fittedvals*(1-fittedvals))

devianceResiduals <- ifelse(as.logical(train_y), 1, -1) *
  sqrt(-2*(train_y *log(fittedvals) + (1-train_y)*log(1-fittedvals)))

par(mfrow = c(1, 2))

ggplot(data.frame(fittedvals, devianceResiduals), aes(x = fittedvals, y = devianceResiduals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "blue") +
  ggtitle("Residuals Plot") +
  xlab("Fitted Values") +
  ylab("Pearson Residuals") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/elastic_LR_peason_residuals.png"))

ggplot(data.frame(fittedvals, devianceResiduals), aes(x = fittedvals, y = devianceResiduals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "blue") +
  ggtitle("Residuals Plot") +
```

```
  xlab("Fitted Values") +
  ylab("Deviance Residuals") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/elastic_LR_dev_residuals.png"))



## Deviance

# can compare with feature selection model
deviance(elastic.lr)



# Logistic Regression Modeling (Using PCA)

## Choosing the number of Principal Components

# sumamry of the sample principal components
summary(PCs)

# plots
var_explained = PCs$sdev ^ 2 / sum(PCs$sdev ^ 2)

ggplot(data.frame(x = 1:length(PCs$sdev), y = var_explained)) +
  geom_line(aes(x = x, y = y, size=1), color = "#EEA236", show.legend = F) +
  geom_point(aes(x = x, y = y, size=2), color = "#EEA236", show.legend = F) +

  ggtitle("Variance Explained") +
  xlab("Sample Principal Component Number") +
  ylab("Variance Explained") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_scree.png"))

tot_var_explained = cumsum(PCs$sdev ^ 2) / sum(PCs$sdev ^ 2)

ggplot(data.frame(x = 1:length(PCs$sdev), y = tot_var_explained), aes(x = x, y = y)) +
  geom_line() +
  ggtitle("Cumulative Variance Explained") +
  xlab("Sample Principal Component Number") +
  ylab("Cumulative Variance Explained") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_cumul_scree.png"))

# create biplot to visualize results of PCA
ggplot_pca(PCs, arrows_colour = "red", arrows_size = 1) + theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_biplot.png"))

ggplot(train_data_PCA, aes(x = PC1, y = PC2, col = factor(TenYearCHD))) +
  geom_point() +
  ggtitle("PC Separability") +
  xlab("PC1") +
  ylab("PC2") +
  theme_minimal() +
  guides(color = guide_legend(title = "Has Heart Disease?")) +
  scale_color_discrete(labels = c("No", "Yes"))
ggsave(paste0(PATH_IMAGE_DIR,"/pca_seperability.png"))

# not really seeing much separability

# Holdout Set CV

# defines holdout set usage for no data leakage
cv_train_data_PCA <- rbind(holdout_data_PCA, train_data_PCA)

cv_train_data_PCA$TenYearCHD <- factor(cv_train_data_PCA$TenYearCHD)
levels(cv_train_data_scaled$TenYearCHD) <- c("no", "yes")

fit_on <- list(ts = nrow(holdout_data_PCA):nrow(cv_train_data_PCA))
pred_on <- list(hs = 1:nrow(holdout_data_PCA))

train.control <- trainControl(method = "cv", classProbs = TRUE,
                              index =  fit_on, indexOut = pred_on,
```

```
                              summaryFunction = twoClassSummary)

cv_train_data_PCA$TenYearCHD <- factor(cv_train_data_PCA$TenYearCHD)
levels(cv_train_data_PCA$TenYearCHD) <- c("no", "yes")

# examine the AUC performance for different number of PCs
vAUC <- c()
for (k in 1:23){
  print(paste("********************** Number of PCs  = ", k, "**********************"))
  cv_pca_glm <- train(TenYearCHD ~ .,
                      data = cv_train_data_PCA[, c(1:k, 24)],
                      method = "glm",
                      family = "binomial",
                      metric = "ROC",
                      trControl = train.control)
  print(cv_pca_glm)

  vAUC <- c(vAUC, cv_pca_glm$results$ROC)

}

levels(cv_train_data_PCA$TenYearCHD) <- c(0, 1)

ggplot(data.frame(x=1:23, vAUC)) +
  geom_line(aes(x = 1:23, y = vAUC, size=1), color = "#46B8DA", show.legend = F) +
  geom_point(aes(x = 1:23, y = vAUC, size=2), color = "#46B8DA", show.legend = F) +
  ggtitle("Validation Performance") +
  xlab("Number of Components Used") +
  ylab("ROC-AUC") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/pca_validation.png"))



## Goodness of fit


pca_LR <- glm(TenYearCHD ~ ., data = train_data_PCA[, c(1:8, 24)], family = binomial)
summary(pca_LR)

# test deviance against null model
null_LR <- glm(TenYearCHD ~ 1, data = train_data_PCA[, c(1:8, 24)], family = binomial)
anova(null_LR, pca_LR, test = "Chisq")

# could equivalently be done
# pchisq(3245.3 - 2841.1, 2340-2329, lower=FALSE)

# chisq goodness of fit test
ts <- sum(residuals(pca_LR, type = "pearson") ^ 2)
cat("P-Value for the Pearson Goodness of Fit Test  = ",  pchisq(ts, pca_LR$df.residual, lower=FALSE), "\n"
    )



## Leverage, Influence, and Residual Analysis


par(mfrow = c(2, 2))
resid_panel(pca_LR, plots = c("resid", "qq"), theme = "minimal", type = "pearson")
ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_pearson_residuals.png"))

resid_panel(pca_LR, plots = c("resid", "qq"), theme = "minimal", type = "deviance")
ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_dev_residuals.png"))

par(mfrow = c(1, 2))
autoplot(pca_LR, which = c(4, 5)) + theme_minimal()

ggsave(paste0(PATH_IMAGE_DIR,"/pca_LR_diagnostics.png"))



# Extreme Gradient Boosted Classifier


train_x = data.matrix(train_data_scaled[, -18])
```

```r
train_y = train_data_scaled[, 18]

# define final training and testing sets
xgb_train = xgb.DMatrix(data = train_x, label = as.numeric(train_y))

xgb.model <- xgboost(data = xgb_train,
                     max.depth = 2,
                     nrounds = 5,
                     objective = 'binary:logistic',
                     lambda = 2,
                     gamma = 0,
                     colsample_bytree = 1,
                     verbose = F)



## Holdout Set CV


# defines holdout set usage for no data leakage
cv_train_data_scaled <- rbind(holdout_data_scaled, train_data_scaled)

train_y <- factor(cv_train_data_scaled[, 18])
levels(train_y) <- c("no", "yes")

fit_on <- list(ts = nrow(holdout_data_scaled):nrow(cv_train_data_scaled))
pred_on <- list(hs = 1:nrow(holdout_data_scaled))

xgbGrid <- expand.grid(nrounds = seq(1, 100, by = 4),
                       max_depth = c(2, 3, 4, 5),
                       eta = 0.1, # default
                       gamma = 0, #default
                       colsample_bytree = 1, # default
                       min_child_weight = 1, # default
                       subsample = 1 # default
)

xgb_trcontrol <- trainControl(method = "cv", classProbs = TRUE,
                              index =  fit_on, indexOut = pred_on,
                              summaryFunction = twoClassSummary)

xgb_model = train(as.matrix(cv_train_data_scaled[, -18]), train_y,
                  trControl = xgb_trcontrol,
                  tuneGrid = xgbGrid,
                  method = "xgbTree",
                  metric = "ROC",
                  verbosity = 0
)

#print(xgb_model)
ggplot(xgb_model) + theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/xgb_tuning.png"))

levels(cv_train_data_scaled$TenYearCHD.1) <- c(0, 1)



# Retrain Best Models Using Entire Training Set

## Update new training and test sets (no more holdout set)


# use original split, then apply over sampling again to the training data to
# bootstrap the larger training set
train_data <- orig_df[train.index, ]
train_data <- ovun.sample(TenYearCHD.1 ~ ., data = train_data, seed = 4)$data
test_data <- orig_df[-train.index, ]

# scale the data for model 1 and model 3
train_data_scaled <- train_data
test_data_scaled <- test_data

normParam <- preProcess(train_data[, -18])
train_data_scaled[, -18] <- as.data.frame(predict(normParam, train_data[, -18]))
test_data_scaled[, -18] <- as.data.frame(predict(normParam, test_data[, -18]))

```

```r
# for PCA datasets
# use original split, then apply over sampling again to the training data to
# bootstrap the larger training set

train_data_PCA <- orig_df_PCA[train.index_PCA, ]
train_data_PCA <- ovun.sample(TenYearCHD.1 ~ ., data = train_data_PCA, seed = 4)$data
test_data_PCA <- orig_df_PCA[-train.index_PCA, ]


# scale data with no data leakage
normParam <- preProcess(train_data_PCA[, -18])
train_data_PCA[, -18] <- predict(normParam, train_data_PCA[, -18])
test_data_PCA[, -18] <- predict(normParam, test_data_PCA[, -18])

# remove response, apply pca
PCs <- prcomp(train_data_PCA[, -18], scale = FALSE, center = FALSE)
train_data_PCA = cbind(as.data.frame(PCs$x), TenYearCHD = train_data_PCA[, 18])

# adjust new holdout and test data with training loadings (new data leakage)
x <- as.matrix(test_data_PCA[, -18])
x <- x %*% PCs$rotation
test_data_PCA <- data.frame(x, TenYearCHD = test_data_PCA[, 18])



## Logistic Regression with Feature Selection


best.subset.lr <- glm(TenYearCHD.1 ~ male.1 + age + education.high.school + education.bachelors +
                        education.graduate.school + cigsPerDay + BPMeds.1 + prevalentHyp.1 +
                        totChol + sysBP + diaBP + diaBP.SQ + glucose.SQ,
                      family = binomial,
                      data = train_data)

summary(best.subset.lr)
plot_model(best.subset.lr) +
  theme_minimal() +
  ggtitle("Ten Year CHD") +
  scale_x_discrete(labels = c(expression("(Glucose)"^"2"),
                              expression("(Diabolic Blood Pressure)"^"2"),
                              "Diabolic Blood Pressure",
                              "Systolic Blood Pressure",
                              "Total Cholesterol",
                              "Prevalent Hypertension (Yes)",
                              "Blood Pressure Medication (Yes)",
                              "Cigarettes Per Day",
                              "Education (Graduate School)",
                              "Education (Bachelor's)",
                              "Education (High School)",
                              "Age",
                              "Gender (Male)"))

ggsave(paste0(PATH_IMAGE_DIR,"/subset_LR_coefs.png"))



## Logistic Regression with Elastic Net


train_x <- model.matrix(TenYearCHD.1 ~ ., train_data)[, -1]
train_y <- train_data$TenYearCHD.1

test_x <- model.matrix(TenYearCHD.1 ~ ., test_data)[, -1]
test_y <- train_data$TenYearCHD.1

best.elastic.lr <- glmnet(train_x, train_y, alpha = 0.6, family = "binomial", lambda = 0.098)
coef(best.elastic.lr)



## Logistic Regression with PCA


best.pca.lr <- glm(TenYearCHD ~ ., data = train_data_PCA[, c(1:8, 24)], family = binomial)
```

```r
summary(best.pca.lr)



## XGBoost


# setup data for xgboost again
trainx = data.matrix(train_data_scaled[, -18])
trainy = train_data_scaled[, 18]

# define predictor and response variables in testing set
testx = data.matrix(test_data_scaled[, -18])
testy = test_data_scaled[, 18]

# define final training and testing sets
xgb_train = xgb.DMatrix(data = trainx, label = as.numeric(trainy))
xgb_test = xgb.DMatrix(data = testx, label = as.numeric(testy))

# create best model from CV above, using entire training set
best.xgb.model <- xgboost(data = xgb_train,
                          max.depth = 2,
                          nrounds = 13,
                          objective = 'binary:logistic',
                          lambda = 2,
                          gamma = 0,
                          colsample_bytree = 1,
                          verbose = F)

summary(best.xgb.model)
importance_matrix = xgb.importance(colnames(xgb_train), model = best.xgb.model)
importance_matrix

ggplot(importance_matrix, aes(x = Gain, y = forcats::fct_reorder(Feature, Gain))) +
  geom_col() +
  theme_minimal() +
  ylab("") +
  ggtitle("XGBoost Feature Importance") +
  scale_y_discrete(labels = c("Age * Cigarettes Per Day",
                              "Blood Pressure Medication (Yes)",
                              "Total Cholesterol",
                              "Heart Rate",
                              "Glucose",
                              "Diabolic Blood Pressure",
                              "Glucose * Systolic Blood Pressure",
                              "Gender (Male)",
                              "Systolic Blood Pressure",
                              "Systolic Blood Pressure * Cigarettes Per Day",
                              "Age"))
ggsave(paste0(PATH_IMAGE_DIR,"/xgb_importance.png"))



# Finding Optimal Thresholds


train_result_subset_lr <- predict(best.subset.lr, newdata = train_data, type = "response")
xtrain <- model.matrix(TenYearCHD.1 ~ ., train_data)[, -1]
train_result_elastic_lr <- predict(best.elastic.lr, newx = train_x, type = "response")
train_result_pca_lr <- predict(best.pca.lr, newdata = train_data_PCA, type = "response")
train_result_xgb <- predict(best.xgb.model, newdata = xgb_train)

train_metrics_subset <- data.frame()
for (threshold in seq(0.41, 0.6, 0.01)){
  train_subset_lr_threshold <- as.factor(as.numeric(train_result_subset_lr > threshold))
  train_subset_lr_table <- table(train_subset_lr_threshold, factor(train_data$TenYearCHD))
  temp <- confusionMatrix(train_subset_lr_table, positive = "1")
  model <- "subset"
  temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
  train_metrics_subset <- rbind(train_metrics_subset, temp)
}

train_metrics_elastic <- data.frame()
for (threshold in seq(0.41, 0.6, 0.01)){
  train_elastic_lr_threshold <- as.factor(as.numeric(train_result_elastic_lr > threshold))
  train_elastic_lr_table <- table(train_elastic_lr_threshold, factor(train_y))
```

```
  temp <- confusionMatrix(train_elastic_lr_table, positive = "1")
  model <- "elastic"
  temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
  train_metrics_elastic <- rbind(train_metrics_elastic, temp)
}

train_metrics_pca <- data.frame()
for (threshold in seq(0.41, 0.6, 0.01)){
  train_pca_lr_threshold <- as.factor(as.numeric(train_result_pca_lr > threshold))
  train_pca_lr_table <- table(train_pca_lr_threshold, factor(train_data$TenYearCHD))
  temp <- confusionMatrix(train_pca_lr_table, positive = "1")
  model <- "pca"
  temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
  train_metrics_pca <- rbind(train_metrics_pca, temp)
}

train_metrics_xgb <- data.frame()
for (threshold in seq(0.41, 0.6, 0.01)){
  train_xgb_threshold <- as.factor(as.numeric(train_result_xgb > threshold))
  train_xgb_table <- table(train_xgb_threshold, factor(train_data$TenYearCHD))
  temp <- confusionMatrix(train_xgb_table, positive = "1")
  model <- "xgb"
  temp <- data.frame(cbind(model, cbind(threshold, t(temp$byClass))))
  train_metrics_xgb <- rbind(train_metrics_xgb, temp)
}

train_metrics_subset %>%
  select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
  mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
  mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
  arrange(desc(Gmean))

train_metrics_elastic %>%
  select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
  mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
  mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
  arrange(desc(Gmean))

train_metrics_pca %>%
  select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
  mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
  mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
  arrange(desc(Gmean))

train_metrics_xgb %>%
  select(model, threshold, Sensitivity, Specificity, Balanced.Accuracy) %>%
  mutate_at(c("Sensitivity", "Specificity", "Balanced.Accuracy"), as.numeric) %>%
  mutate(Gmean = sqrt(Sensitivity * Specificity)) %>%
  arrange(desc(Gmean))



# Testing Results


# store predicted probabilities for all four models
result_subset_lr <- predict(best.subset.lr, newdata = test_data, type = "response")
result_elastic_lr <- predict(best.elastic.lr, newx = test_x, type = "response")
result_pca_lr <- predict(best.pca.lr, newdata = test_data_PCA, type = "response")
result_xgb <- predict(best.xgb.model, newdata = xgb_test)

# confusion matrices
cat("Log Reg with Feature Selection", "\n\n")
subset_lr_threshold <- as.factor(as.numeric(result_subset_lr>0.45))
subset_lr_table <- table(subset_lr_threshold, factor(test_data$TenYearCHD))
confusionMatrix(subset_lr_table, positive = "1")

cat("Log Reg with Elastic", "\n\n")
elastic_lr_threshold <- as.factor(as.numeric(result_elastic_lr>0.49))
elastic_lr_table <- table(elastic_lr_threshold, factor(test_data$TenYearCHD))
confusionMatrix(elastic_lr_table, positive = "1")

cat("Log Reg with PCA", "\n\n")
pca_lr_threshold <- as.factor(as.numeric(result_pca_lr>0.51))
pca_lr_table <- table(pca_lr_threshold, factor(test_data_PCA$TenYearCHD))
confusionMatrix(pca_lr_table, positive = "1")
```

```r

cat("XGBOOST", "\n\n")
xgb_threshold <- as.factor(as.numeric(result_xgb > 0.5))
xgb_table <- table(xgb_threshold, factor(test_data_scaled$TenYearCHD.1))
confusionMatrix(xgb_table, positive = "1")




# store prediction results
pred_subset_lr <- prediction(result_subset_lr, test_data$TenYearCHD)
pred_elastic_lr <- prediction(result_elastic_lr, test_data$TenYearCHD)
pred_pca_lr <- prediction(result_pca_lr, test_data$TenYearCHD)
pred_xgb <- prediction(result_xgb, test_data$TenYearCHD)

# plot different useful performance metrics
measures <- c("acc", "sens", "spec", "fpr")
meas_names <- c("Accuracy", "Sensitivity (TPR)", "Specificity", "FPR")
cols <- c("#D43F3A", "#EEA236", "#5CB85C", "#46B8DA")

measure <- "acc"
meas_name <- "Accuracy"
subset_df <- data.frame(x = performance(pred_subset_lr, measure)@x.values[[1]],
                        y = performance(pred_subset_lr, measure)@y.values[[1]])
elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)@x.values[[1]],
                         y = performance(pred_elastic_lr, measure)@y.values[[1]])
pca_df <- data.frame(x = performance(pred_pca_lr, measure)@x.values[[1]],
                     y = performance(pred_pca_lr, measure)@y.values[[1]])
xgb_df <- data.frame(x = performance(pred_xgb, measure)@x.values[[1]],
                     y = performance(pred_xgb, measure)@y.values[[1]])

ggplot() +
  geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
  geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
  geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
  geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
  labs(color = "Model") +
  xlab("Threshold") +
  ylab(meas_name) +
  scale_color_manual(values = cols,
                     labels  = c("Log Reg (Feat. Sel.)",
                                 "Log Reg (Elastic)",
                                 "Log Reg (PCA)",
                                 "XGBoost")) +
  geom_hline(yintercept = 0.85, linetype = "dotted", color = "black", size = 0.8) +
  annotate("text", x=0.15, y=0.83, label= "Natural Class Balance", size = 2.5) +
  theme(legend.position = "right") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/results_acc.png"))

measure <- "sens"
meas_name <- "Sensitivity (TPR)"
subset_df <- data.frame(x = performance(pred_subset_lr, measure)@x.values[[1]],
                        y = performance(pred_subset_lr, measure)@y.values[[1]])
elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)@x.values[[1]],
                         y = performance(pred_elastic_lr, measure)@y.values[[1]])
pca_df <- data.frame(x = performance(pred_pca_lr, measure)@x.values[[1]],
                     y = performance(pred_pca_lr, measure)@y.values[[1]])
xgb_df <- data.frame(x = performance(pred_xgb, measure)@x.values[[1]],
                     y = performance(pred_xgb, measure)@y.values[[1]])

ggplot() +
  geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
  geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
  geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
  geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
  labs(color = "Model") +
  xlab("Threshold") +
  ylab(meas_name) +
  scale_color_manual(values = cols,
                     labels  = c("Log Reg (Feat. Sel.)",
                                 "Log Reg (Elastic)",
                                 "Log Reg (PCA)",
                                 "XGBoost")) +
  theme(legend.position = "right") +
  theme_minimal()
```

```r
ggsave(paste0(PATH_IMAGE_DIR,"/results_sens.png"))

measure <- "spec"
meas_name <- "Specificity"
subset_df <- data.frame(x = performance(pred_subset_lr, measure)@x.values[[1]],
                        y = performance(pred_subset_lr, measure)@y.values[[1]])
elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)@x.values[[1]],
                         y = performance(pred_elastic_lr, measure)@y.values[[1]])
pca_df <- data.frame(x = performance(pred_pca_lr, measure)@x.values[[1]],
                     y = performance(pred_pca_lr, measure)@y.values[[1]])
xgb_df <- data.frame(x = performance(pred_xgb, measure)@x.values[[1]],
                     y = performance(pred_xgb, measure)@y.values[[1]])

ggplot() +
  geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
  geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
  geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
  geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
  labs(color = "Model") +
  xlab("Threshold") +
  ylab(meas_name) +
  scale_color_manual(values = cols,
                     labels  = c("Log Reg (Feat. Sel.)",
                                 "Log Reg (Elastic)",
                                 "Log Reg (PCA)",
                                 "XGBoost")) +
  theme(legend.position = "right") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/results_spec.png"))

measure <- "fpr"
meas_name <- "FPR"
subset_df <- data.frame(x = performance(pred_subset_lr, measure)@x.values[[1]],
                        y = performance(pred_subset_lr, measure)@y.values[[1]])
elastic_df <- data.frame(x = performance(pred_elastic_lr, measure)@x.values[[1]],
                         y = performance(pred_elastic_lr, measure)@y.values[[1]])
pca_df <- data.frame(x = performance(pred_pca_lr, measure)@x.values[[1]],
                     y = performance(pred_pca_lr, measure)@y.values[[1]])
xgb_df <- data.frame(x = performance(pred_xgb, measure)@x.values[[1]],
                     y = performance(pred_xgb, measure)@y.values[[1]])

ggplot() +
  geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
  geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
  geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
  geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
  labs(color = "Model") +
  xlab("Threshold") +
  ylab(meas_name) +
  scale_color_manual(values = cols,
                     labels  = c("Log Reg (Feat. Sel.)",
                                 "Log Reg (Elastic)",
                                 "Log Reg (PCA)",
                                 "XGBoost")) +
  theme(legend.position = "right") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/results_fpr.png"))

subset_df <- data.frame(x = performance(pred_subset_lr, "sens", "fpr")@x.values[[1]],
                        y = performance(pred_subset_lr, "sens", "fpr")@y.values[[1]])
elastic_df <- data.frame(x = performance(pred_elastic_lr, "sens", "fpr")@x.values[[1]],
                         y = performance(pred_elastic_lr, "sens", "fpr")@y.values[[1]])
pca_df <- data.frame(x = performance(pred_pca_lr, "sens", "fpr")@x.values[[1]],
                     y = performance(pred_pca_lr, "sens", "fpr")@y.values[[1]])
xgb_df <- data.frame(x = performance(pred_xgb, "sens", "fpr")@x.values[[1]],
                     y = performance(pred_xgb, "sens", "fpr")@y.values[[1]])

ggplot() +
  geom_line(data = subset_df, aes(x = x, y = y, color = cols[1]), lwd=1.5) +
  geom_line(data = elastic_df, aes(x = x, y = y, color = cols[2]), lwd=1.5) +
  geom_line(data = pca_df, aes(x = x, y = y, color = cols[3]), lwd=1.5) +
  geom_line(data = xgb_df, aes(x = x, y = y, color = cols[4]), lwd=1.5) +
  labs(color = "Model") +
  xlab("1 - Specifiicity (FPR)") +
  ylab("Sensitivity (TPR)") +
  scale_color_manual(values = cols,
```

```
                      labels  = c("Log Reg (Feat. Sel.)",
                                  "Log Reg (Elastic)",
                                  "Log Reg (PCA)",
                                  "XGBoost")) +
  geom_abline(intercept = 0, slope = 1, linetype = "dotted", color = "black", size = 0.5) +
  theme(legend.position = "right") +
  theme_minimal()
ggsave(paste0(PATH_IMAGE_DIR,"/results_roc.png"))



roc_subset_lr <- roc(test_data$TenYearCHD, result_subset_lr)
auc_subset_lr <- roc_subset_lr$auc
cat("ROC-AUC from Log Reg w/ Feature Selection: ", auc_subset_lr, "\n")

roc_elastic_lr <- roc(test_data$TenYearCHD, result_elastic_lr)
auc_elastic_lr <- roc_elastic_lr$auc
cat("ROC-AUC from Log Reg w/ ElasticNet: ", auc_elastic_lr, "\n")

roc_pca_lr <- roc(test_data$TenYearCHD, result_pca_lr)
auc_pca_lr <- roc_pca_lr$auc
cat("ROC-AUC from Log Reg w/ PCA: ", auc_pca_lr, "\n")

roc_xgb <- roc(as.numeric(testy), result_xgb)
auc_xgb <- roc_xgb$auc
cat("ROC-AUC from XGBoost: ", auc_xgb, "\n")
```