DSA 211 – Statistical Learning with R

Professor Kwong Koon Shing

Group Project

G1

Done by:

Sean Wei-De Chang
Isaac Wong Kang Wei
Megan Chong Wen Xi
Charmaine Lee Wai Ying

1. Executive Summary

In the UK, the total annual cost of all heart disease related illnesses is £7 billion. Being able to accurately assess the ten year risk of an individual developing cardiovascular heart disease(CHD) is vital for not only them, but many other developed countries as well, such as Singapore. Our project aims to identify the key variables that increase an individual's Ten-Year CHD risk based on 15 variables in WHO's dataset.

In our analysis, we explored tree based methods and logistic regression models to derive the best prediction for heart disease, under both resampled data and original train data. We applied forward stepwise selection which evaluated models based on deviance, and subsequently conducted a cross-validation analysis on the respective training sets to derive the best models.

Next, we applied decision tree algorithms, random forest algorithms and bagging algorithms on both the training data and upsampled data. Based on the specificity metric and OOB errors, we found that models trained on upsampled data performed better than those trained on the original data set.

Finally, we tested the best models derived earlier on the testing dataset and evaluated each model based on the specificity and sensitivity criterion. We came to a conclusion that our decision tree model trained on upsampled data performed the best, and acts as the final model that we will use.

2. Intro and background

2.1 Key Issue

WHO lists cardiovascular disease as the leading cause of death globally, taking an estimated 17.9 million lives every year, or 31% of all deaths worldwide.[1] In Singapore, coronary heart disease (CHD) accounted for 29.2% of all deaths in 2018, and everyday 17 people die from cardiovascular disease.[2] Early detection of cardiovascular diseases can aid in making lifestyle changes to reduce modifiable risk factors in high risk patients, hence reducing health complications.

Our analysis will identify the most relevant factors that contribute to heart diseases, as well as coming up with a model to predict the probability of an individual having CHD given the other independent variables. This would allow health institutions to forecast which of their patients are at high risk of contracting CHD in order to recommend early treatment. It would also allow individuals at risk of disease to be aware of the risks and take appropriate actions to change their lifestyles in order to prevent it.

2.2 The Dataset

Our dataset was obtained from Kaggle, and is from an ongoing cardiovascular study on residents in the town of Framingham, Massachusetts. In 1948, an original cohort of 5,209 men and women between 30 and 62 years old were recruited. Participant clinic data was collected during three examination periods, approximately 6 years apart, from roughly 1956 to 1968.

The aim is to assess a patient's risk of developing coronary heart disease within the next ten years (TenYearCHD) given 15 attributes. Details of the attributes can be found in Appendix A. Overall, the data set has 557 patients with heart disease, and 3099 observations without heart disease.

2.3 Methodology

To derive the best model for this task, we used 80% of the data as our training set and 20% for our test set. We used this training data to derive the best model through logistic regression and tree-based models, and based on various metrics like cross validation and out-of-bag error estimates. Subsequently, we evaluated the best overall model by considering the prediction ability of the model when shown unseen test data. Once we determined this model, we trained an overall model using the full dataset, for use in practical cases.

3. Data Preparation

3.1 Cleaning of Data

Before exploring the data, we had to reformat categorical variables as factors and remove the missing values. The categorical variables are *currentSmoker, diabetes, prevalentStroke, prevalentHyp, male, BPMeds, TenYearCHD.* We did not encode *education* as a categorical variable, and left it as an integer scaling from levels 1 to 4, where a higher value indicated a higher education level.

[1] https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1
[2] https://www.myheart.org.sg/my-heart/heart-statistics/singapore-statistics/

This new dataset was used for basic data visualization and exploration in Section 3.2. We removed some values based on the assumption that the data is missing at random. Removing the missing values still gave us 3656 observations which is still a sizable number. The *training* data set consists of 2924 observations while the *test* data set consists of 732 observations.

3.2 Data Visualization

Variable	**P-value**
Male	2.9e-8
Age	2.2e-16
Education	8.246e-07
currentSmoker	0.2463
cigsPerDay	0.003256
BP Meds	7.11e-08
Prevalent Stroke	0.003461
Prevalent Hypertension	2.2e-16
Diabetes	1.63e-08
Total Cholesterol	4.199e-07
Sys BP	2.2e-16
Dia BP	2.503e-14
BMI	6.434e-06
heartRate	0.22
Glucose Level	5.059e-06

Fig 1: Result of proportion tests and 2 sample t-test

After preparing the dataset, we proceeded with basic exploration of our variables. We started by visualizing each variable with respect to a person's Ten-year CHD risk. To get a brief idea of the significance of each variable with respect to *TenYearCHD*, we conducted proportion tests for binary / multinary variables and 2-sample t-tests for the rest. It turns out that only two variables, *currentSmoker* and *heartRate* are not significant.

Following this, we visualized the correlation between the independent variables to obtain a brief idea of their pairwise interactions. At a glance, there does not seem to be any strong interaction between any of the variables, which motivated us to exclude this from our model.

3.3 Resampling Technique

Classification algorithms are typically built to minimise classification error, and resultantly do not perform well on imbalanced data. For our given data set, predicting that an individual has no heart disease will still give us 0 sensitivity.

Data set	**Heart Disease**	**No Heart Disease**
Train	443	2481
Upsampled Train	2481	2481

Fig 2: Breakdown of Train and Upsampled Train

As such, we decided to apply resampling techniques to create a new balanced data set, and will compare the results of this methodology to apply statistical learning algorithms to the original train data set. The method we applied is known as upsampling: the algorithm resamples from the minority class to ensure that there is an equal balance between heart disease and no heart disease.

4. Logistic Regression

To derive the best logistic regression model, we utilised the forward stepwise selection methodology for both the *upsampled* data and the *original train* data. The forward selection algorithm is a greedy algorithm that iteratively selects the next variable according to which variable gives the model the lowest deviance. Based on the 15 models derived, we identified the 5 best models with the lowest AIC and BIC each. We decided to only continue our analysis with this subset to save computational time.

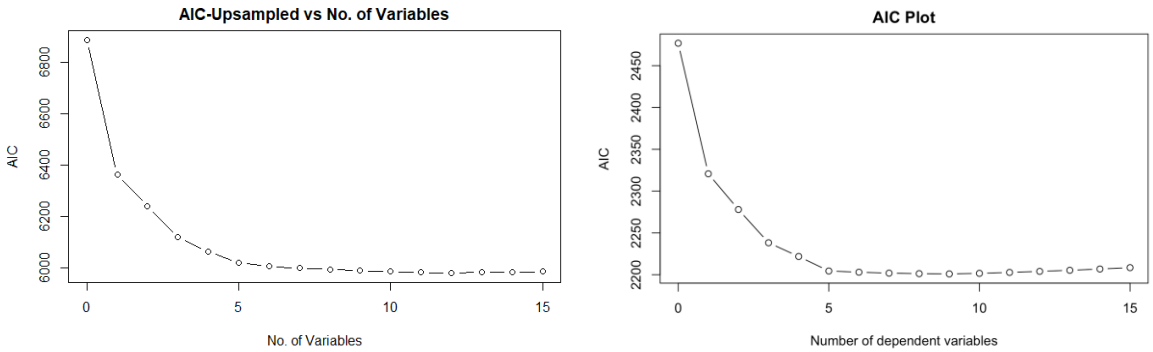


Fig 3: Graphs of AIC on unsampled (left) & original train (right) dataset

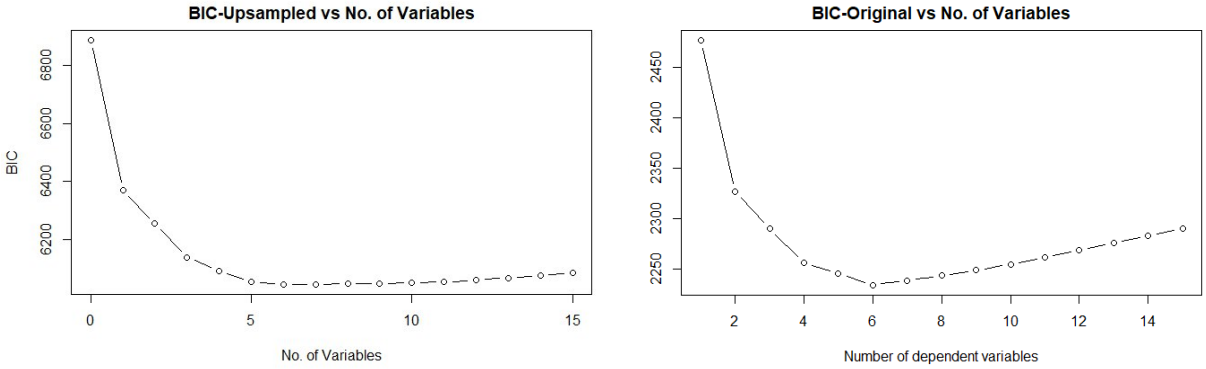


Fig 4: Graphs of BIC on unsampled (left) & original train (right) dataset

Data	**IC**	**5 Best Models (Number of Variables)**
Upsampled	AIC	**M1:** 10, **M2:** 8, **M3:** 12, **M4:** 7, **M5**: 11
Upsampled	BIC	**M1:** 7, **M2:** 6, **M3:** 8, **M4:** 9, **M5:** 10
Original Train	AIC	**M1**: 9, **M2:** 8, **M3:** 10, **M4**: 7, **M5:** 11
Original Train	BIC	**M1:** 5, **M2:** 6, **M3:** 7, **M4:** 4, **M5:** 8

Fig 5: Best 5 models chosen with lowest AIC or BIC

From the figures, it can be seen that the AIC plot flattens out as more variables are included. On the other hand, there is more of a U-shaped graph for the BIC plot, because BIC tends to select simpler models. This highlights the difference in the models that are selected for cross validation in BIC and AIC. There is also some difference in the models selected from the upsampled data compared to the normal data, when using the same criterion.

4.1 10-fold CV on Best Models from AIC and BIC

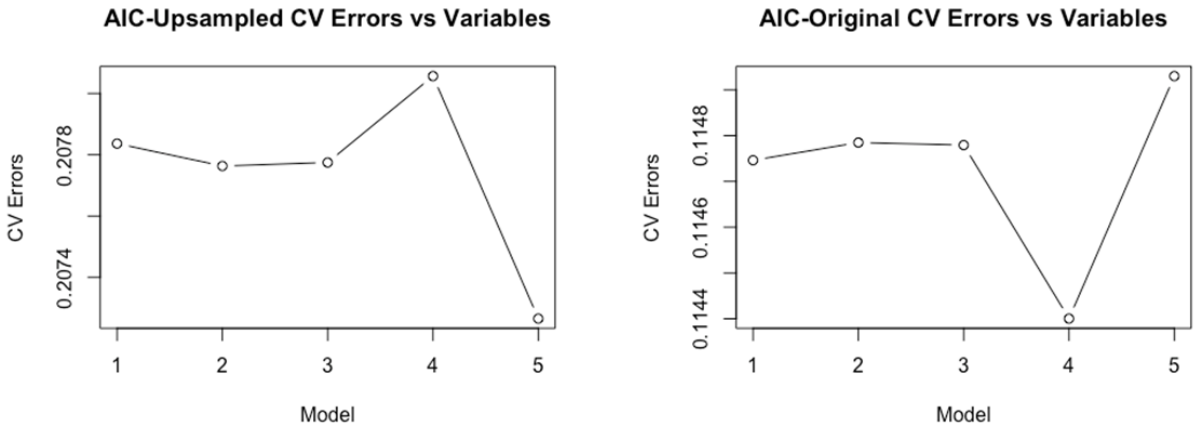


Fig 6: Graphs of AIC on unsampled (left) & original train (right) dataset

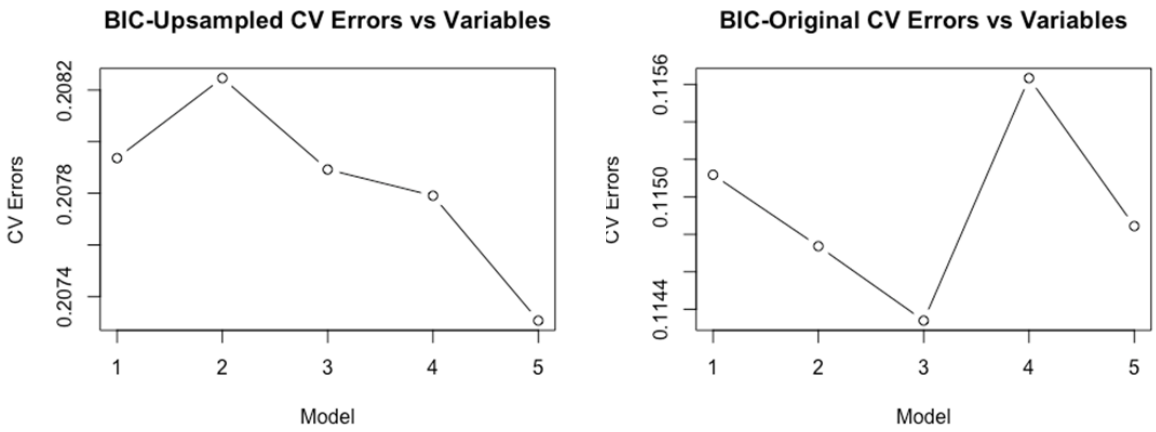


Fig 7: Graphs of BIC on unsampled (left) & original train (right) dataset

Based on these 20 models found under the criterion of AIC and BIC, we used 10-fold cross validation to measure the performance of these model instances. The 10-fold cross validation algorithm computes the deviance, defined as the squared difference between the conditional probability and the actual output (Appendix F). Based on this statistic, we derived our 4 best models, one for each unique dataset and information criterion pair.

Best Models (Variables)	**IC**	**Dataset**	**CV Errors**
Male, age, cigsPerDay, prevalentHyp, totChol, sysBP, glucose	AIC (M4)	Original Training Set	0.1147461
Male, age, education, cigsPerDay, prevalentStroke, prevalentHyp, totChol, sysBP, BMI, heartRate, glucose	AIC (M5)	Upsampled Training Set	0.2072655

Male, age, cigsPerDay, prevalentHyp, totChol, sysBP, glucose	BIC (M3)	Original Training Set	0.1143396
Male, age, cigsPerDay, prevalentStroke, prevalentHyp, totChol, sysBP, BMI, heartRate, glucose	BIC (M5)	Upsampled Training Set	0.2073066

Fig 8: Summary of best 4 models from AIC and BIC on Original train and unsampled datasets

From this cursory analysis, we found out that the overall best models chosen are very different, owing to the upsampling performed earlier. These 4 best models will be introduced to unseen test data as a basis of comparison for the effectiveness of each of these models.

5. Tree Based Methods

5.1 Decision Trees

Beyond fitting a logistic regression model, we used tree based algorithms to attempt to derive a better classification. The first algorithm we applied is a decision tree, with deviance as the criteria for splitting.

Algorithm Used	**Data Trained on**	**Residual Mean Deviance**	**Misclassification Error Rate**
Decision Tree	Train Data	0.789	0.1505
Decision Tree	Upsampled Train Data	1.253	0.3508

Fig 9: Result of Decision Tree Algorithms

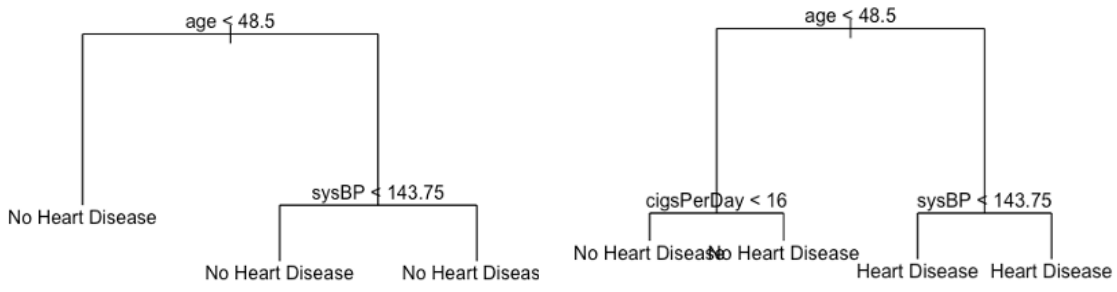


Fig 10: Decision Tree Diagram for Original Data (Left) and Upsampled Data (Right)

While the initial results of the decision tree look optimistic with a misclassification of 0.1505, closer inspection of the tree showed that all 3 terminal nodes predicted “No Heart Disease”,

and resultantly misclassified every individual who had heart disease. Given that deviance is a function of the proportion of a node and the number of observations, the decision tree algorithm was able to minimise residual mean deviance by predicting “No Heart Disease” for all patients.

Upon applying the same decision tree algorithm to both the *upsampled* and *downsampled* data, we were able to derive 4 terminal nodes, 2 of which classified the data as “Heart Disease”, showing a clear improvement over training on the *original* data set.

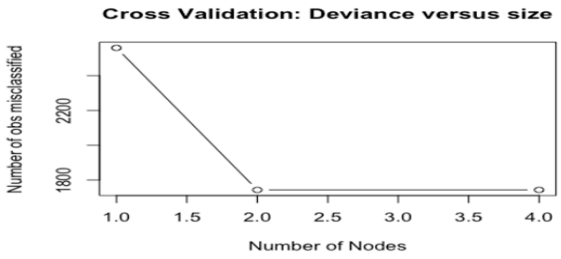


Fig 11: Cross Validation of Upsampled (left)

Given the small number of terminal nodes, pruning of the decision tree does not seem necessary, as the simplicity of the model suggests that we are unlikely to be overfitting the model to the training data. 10-fold cross validation on the *upsampled* data showed that 4 terminal nodes gave the lowest misclassification error, and thus we did not reduce the number of nodes in our tree. Overall, we will use the tree trained with the *upsampled* train data as the best decision tree model to feed into the test data, since the other decision tree is completely unable to predict Heart Disease.

5.2 Ensemble Learning (Random Forest and Bagging)

We applied Ensemble Learning methods hoping to improve prediction of our model. Growing many different trees each with bootstrapped data sets helps to reduce the bias of the model, by aggregating the results from these different data sets. Moreover, random forest approach will also decorrelate and reduce the variance of our model, and allow our model to perform better on unseen data.

With ensemble learning, we are able to use out-of-bag error estimates as an approximation for the prediction error, and an alternative to cross-validation, because these observations were not used in the bootstrapped dataset that was used to grow the tree.

Algorithm	Data	OOB Estimate of Error Rate	Sensitivity (Train)	Specificity (Train)
Bagging (150 trees)	Train Data	15.83%	0.3963	0.8592
Bagging (200 trees)	Upsampled Data	2.94%	0.9458	0.9982
RF (200 trees, mtry = 6)	Train Data	14.91%	0.5357	0.8570
RF (200 trees, mtry = 6)	Upsampled Data	2.23%	0.9593	0.9974

Fig 12: Results of Bagging and Random Forest

Initially, we started bagging and random forest algorithms with 1000 trees, and proceeded to tune the number of trees grown. We observed that OOB errors and prediction errors for each category on the training data set flattened out when approximately 150 trees were grown for the *original* data and 200 trees were grown for the *upsampled* data. Similarly, for the Random Forest models, we found that OOB error flattened out at approximately 200 trees grown (Appendix).

The train sensitivity of the bagging algorithm on the *training* set was low (0.3963) compared to the bagging algorithm trained on *upsampled* data (0.9458). This is also the case for the specificity. Given that these metrics are low on the *train* data, we can reasonably believe this is also the case for the *test* data. When comparing the OOB Error Estimates of the models, the bagging algorithm trained on *normal* data has much higher OOB error (15.83%) compared to training on *upsampled* data (2.94%). It is evident that ensemble learning algorithms also suffer when we train the model with the *imbalanced* data, due to the same reasons we explained earlier. As such, the model trained on the *original* data would not be useful in predicting if patients had heart disease.

From the results, it is clear that the random forest model trained with *upsampled* data performed the best, as it produced the lowest OOB estimate of error rate, with extremely high sensitivity (0.9593) and specificity (0.9974).

6. Evaluation of Important Variables

6.1 Significant Variables

Model	Important Variables
Upsampled Tree	*Age, cigsPerDay, sysBP*
Original AIC	*Male, age, cigsPerDay, prevalentHyp, totChol, sysBP, glucose*
Original BIC	*Male, age, cigsPerDay, prevalentHyp, totChol, sysBP, glucose*

Upsampled AIC	*Male, age, education, cigsPerDay, prevalentStroke, prevalentHyp, totChol, sysBP, BMI, heartRate, glucose*
Upsampled BIC	*Male, age, cigsPerDay, prevalentStroke, prevalentHyp, totChol, sysBP, BMI, heartRate, glucose*

Fig 13: Variables used for splitting

Universally, all models agree that *cigsPerDay, Age,* and *SysBP* are important factors in predicting heart disease. In addition, from subset selection, we see that *male, age, cigsPerDay, prevalentHyp, totChol* and *glucose* are present in the models with the lowest AIC and BIC under the *original train* dataset.

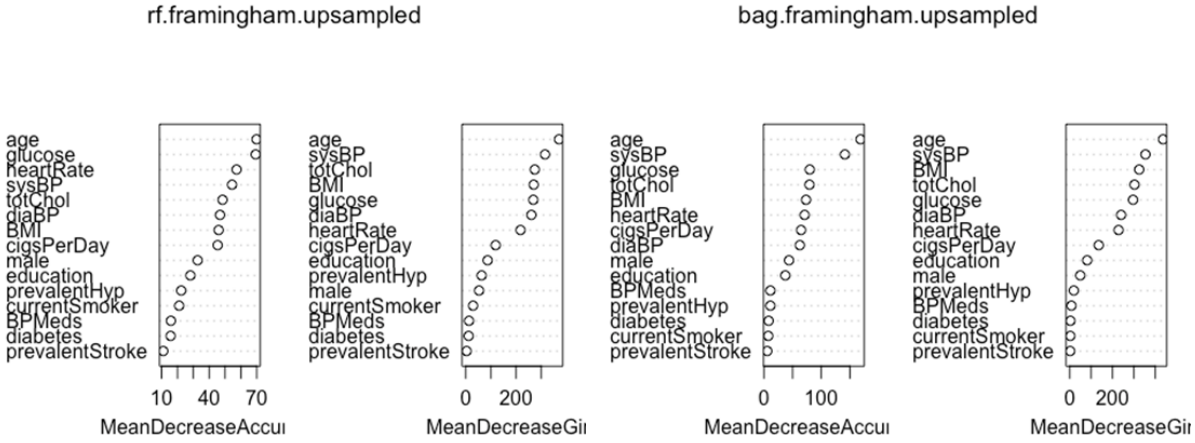


Fig 14: Importance plot for best Random Forest model and Bagging model (upsampled data)

The importance plots from both the bagging algorithm and the Random Forest are fairly similar. They agree that *sysBP, glucose, totChol* are important, but place less emphasis on the rest of the variables. Interestingly, these algorithms did not feature *cigsPerDay* very highly in terms of importance for increasing purity of nodes, or increasing accuracy, as compared to the other algorithms.

Looking back at our original 2 sample t-tests and proportion tests, these results seem consistent with what we had deduced earlier. *Smoker* and *heartRate* did not show a significant difference between individuals with and without heart disease, as both of these variables were not shown to be important in our best models.

7. Evaluation of Models

7.1 Appropriateness of Selection Criteria

For predicting heart disease, medical institutions typically use a rule-out diagnosis rather than a rule-in diagnosis. This means that it is more important to correctly rule-out individuals not at risk of CHD, in order to better address the needs of individuals that may potentially have the disease. There are several metrics used to evaluate models, mainly sensitivity, specificity, and error rate. In this case, sensitivity, or true positive rate tells us what percentage of people with

CHD were correctly identified, while specificity, or true negative rate tells us what percentage of people without CHD were correctly identified.

Among the evaluation metrics, sensitivity is most appropriate because the cost of a false positive is lower than that of a false negative. Specifically, the cost of identifying someone at low risk of CHD as a high risk individual (more invasive tests to confirm diagnosis) is lower than classifying someone at high risk as a low risk of developing the disease (potentially not addressing the issue), which is why we focused on minimising false negative rate. For this reason, misclassification error rate is also not appropriate since it fails to consider the differing costs of false positives and false negatives. At the same time, specificity must also be taken into consideration as it ensures that a high sensitivity does not come from simply predicting all to be positive.

7.2 Modifications with Interaction Term for Best Logistic Regression Model

Noting that our best models for AIC and BIC include the variables *male, age, cigsPerDay, prevalentStroke, prevalentHyp, totChol, sysBP, BMI, heartRate, glucose*, we did a pairwise plot zooming in on only these variables. It does not seem like there are any significant interactions between the independent variables. Hence, we did not explore any further modifications to our model.

7.3 Assessing All Models with *Test* Set

Finally, we tested all the models using the *test* set, to evaluate how each model performs when shown unseen data.

Model	**Sensitivity**	**Specificity**	**Error Rate**
tree.framingham.upsampled	0.7436	0.5366	0.4303
bag.framingham.upsampled	0.1709	0.9480	0.1762
rf.framingham.upsampled	0.1453	0.9707	0.1612
aic.upsampled.final	0.3077	0.3496	0.6571
aic.original.final	0.9487	0.0032	0.8456
bic.upsampled.final	0.3162	0.3577	0.6489
bic.original.final	0.9487	0.0033	0.8456

Fig. 15: Classification Analysis

Based on the results, *aic.original.final* and *bic.original.final* performed well in terms of sensitivity. However, we noted that they performed extremely poorly in terms of specificity, meaning that these 2 models were classifying most low risk individuals as high risk, making the prediction inaccurate. This is also reflected in their high error rates of about 85%.
On the other hand, the bagging and the random forest models have extremely high specificity (0.9480 and 0.9707), but have extremely low sensitivity (0.1709 and 0.1453). As mentioned

earlier, low sensitivity is extremely costly in this scenario, and we resultantly exclude these models. Interestingly, on the *training* data, these models showed very high sensitivity (0.9537 and 0.9870). This suggests that our model is overfitted to our data set, and there is high variance.

As a result, we looked at the next best model, which is the upsampled decision tree. Despite the lower sensitivity, specificity is significantly better, minimising the tradeoff between sensitivity and specificity. Evidently, this analysis also highlights the effectiveness of implementing upsampling as a solution to dealing with imbalanced data. Upsampled logistic regression models performed significantly better than the logistic regression models trained on *normal* data, based on error rate and specificity. Moreover, the best tree-based models were derived from *upsampled* data.

7.4 Final Model: Decision Tree

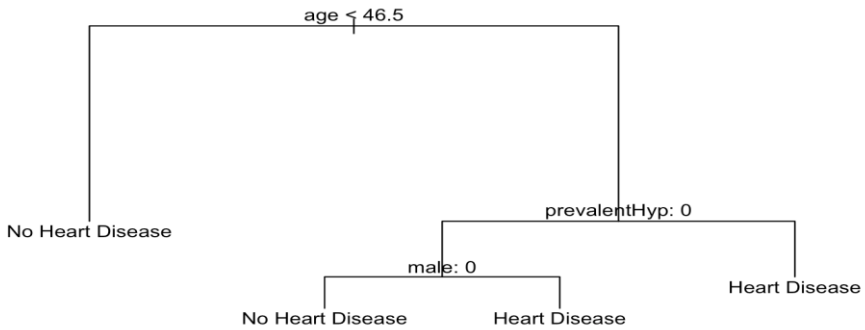


Fig 16: Final Decision Tree Grown

To derive a final model for application, we upsampled the entire framingham dataset and trained our decision tree model on it. Below shows the final variables that our model selected. As we can see, the variables used for splitting were *age, prevalentHyp,* and *gender*, which differed from the set of variables used for the upsampled tree, which was *age, cigsPerDay, sysBP*. This difference highlights the disadvantage of decision trees: they are prone to overfitting to the training data, resulting in highly variable trees being grown when introduced to new data.

8. Application

Having a decision tree as our final model is appropriate for this situation. It helps the general population assess their own risk more easily without much complications. Unlike a random forest model or a logistic regression model, this decision tree is more easily understood and applied by the layman.

In general, predicting CHD has many practical direct and indirect applications. These key indicators that we have found suggest an emphasis on preventative medicine. On the ground level, GPs and other healthcare professionals can advise high-risk individuals on preventive measures to manage chronic diseases well to keep their risk low. For example, patients can

be taught to adopt a new emphasis on certain diets which help in weight loss and lowering their cholesterol levels, something which many plant-based clinics in the USA are already trying to do. Also, more studies can be conducted based on these indicators to see if preemptive medication can effectively reduce risk. At a higher level, public health service providers and the government can find out how they expect CHD to progress in the country and the associated implications. This helps them take preventive measures such as implementing certain policies to reduce the potential risks, or allocating funds to tackle the issue.

9. Limitations: Comparison of Decision Tree with Conventional Wisdom

Our decision tree model did not include other variables typically seen as good predictors of CHD. Typical indicators also include smoking, gender and hypertension, which were not included in our model. This could mean that the model generated from our dataset may be incomplete in identifying the best predictors of CHD. These may affect the accuracy of the direct application of the chosen best model in predicting risk of heart disease. However, these factors may work through other major risk factors such as cholesterol levels, blood pressure and BMI.

10. Conclusion

In conclusion, the best model chosen to predict an individual's risk of developing CHD in the next ten years is our decision tree model. This model was chosen among others based on our choice of selection criteria, which is specificity, since it is the most appropriate in this situation. However, we must note that the methodology explored here is not extensive and there are many other algorithms that have potential to produce better results.

For example, cost-sensitive learning is a general approach to penalising false negatives more than a false positive, pushing algorithms to train a model that is better able to predict rare events. A key disadvantage of the upsampling technique that we employed is that it may result in the model overfitting to our current data set. This was seen by the low

Moreover, upsampling methodology may not be sufficient, there are other oversampling methods such as Random Over-Sampling Examples (ROSE) and Synthetic Minority Over-Sampling Technique (SMOTE) that offer alternatives for creating more realistic balanced data sets. We could also look to explore other methods to fill in the missing values, instead of simply removing any observation with missing data. The assumption that we made that the values are missing at random is unlikely. Being able to fill this data would give us additional observations that may enable us to train a better model.

Despite the limitations, there are still many practical applications of predicting *TenYearCHD* using the selected model, especially due to its accuracy in correctly identifying individuals not at risk of CHD. In this way, more attention can be paid to those not identified as low risk (potentially at risk of CHD).

References

Framingham Heart Study dataset. (n.d.). Retrieved from https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset

Cardiovascular diseases. (n.d.). Retrieved from https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1

Singapore Statistics. (n.d.). Retrieved from https://www.myheart.org.sg/my-heart/heart-statistics/singapore-statistics/

Risk factors for Cardiovascular diseases. (n.d.). Retrieved from https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

Appendix A: Description of Variables

Description of variables:

1. Male: gender of patient (1 if male)
2. Age: Age of the patient
3. Education: 1 = Some High School; 2 = High School or GED; 3 = Some College or Vocational School; 4 = college
4. currentSmoker: 0 = nonsmoker; 1 = smoker
5. cigsPerDay: number of cigarettes smoked per day (estimated average)
6. BPMeds: 0 = Not on Blood Pressure medications; 1 = Is on Blood Pressure medications
7. prevalentStroke: whether or not the patient had previously had a stroke
8. prevalentHyp: whether or not the patient was hypertensive
9. diabetes: 0 = No; 1 = Yes
10. totChol: mg/dL
11. sysBP: mmHg
12. diaBP: mmHg
13. BMI: Body Mass Index calculated as: Weight (kg) / Height (meter-squared)
14. heartRate: Beats/Min (Ventricular)
15. glucose: mg/dL

Appendix B: Hypothesis Testing Results for Classification Variables

	Variable: Male	
10 Year CHD	0	1
Heart Disease	250	307
No Heart Disease	1784	1315
P value	2.9e-8	

	Variable: Education			
10 Year CHD	1	2	3	4
Heart Disease	291	231	75	60
No Heart Disease	1235	970	531	363
P-value	8.246e-07			

	Variable: Current Smoker	
10 Year CHD	0	1
Heart Disease	272	285
No Heart Disease	1596	1503

P-value	0.2463

	Variable: BPMeds	
10 Year CHD	0	1
Heart Disease	520	37
No Heart Disease	3025	74
P-value	7.11e-08	

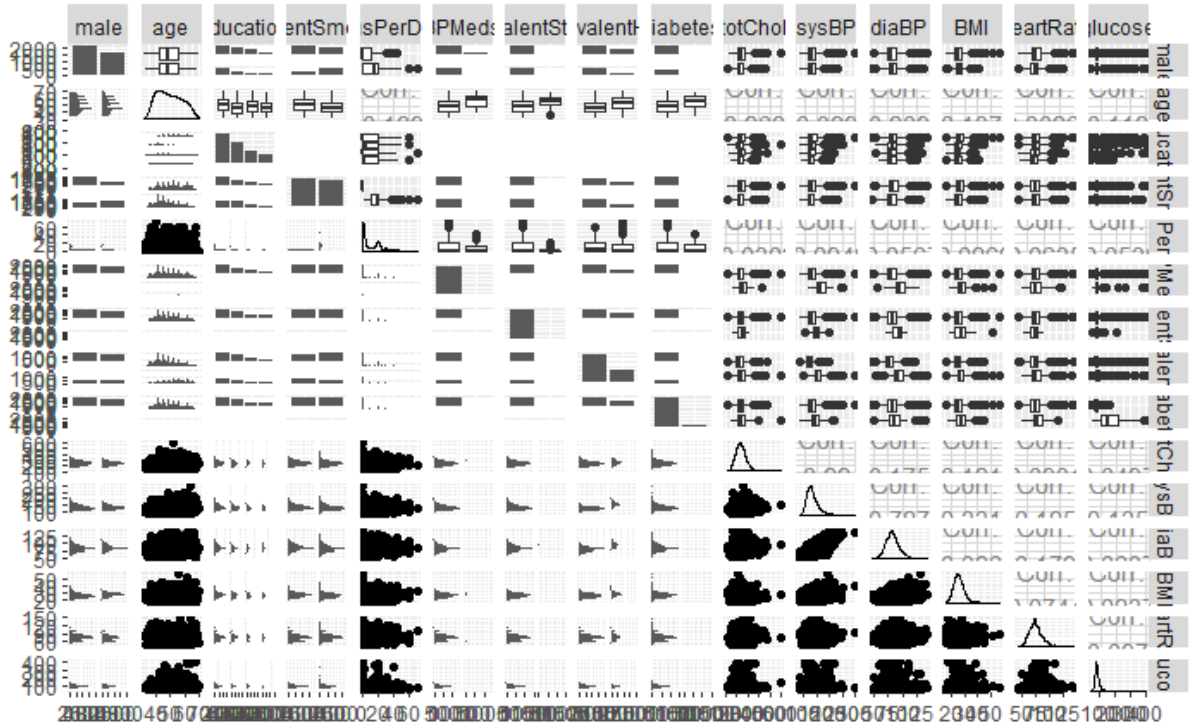
	Variable: PrevalentStroke	
10 Year CHD	0	1
Heart Disease	549	8
No Heart Disease	3086	13
P-value	0.003461	

	Variable: PrevalentHyp	
10 Year CHD	0	1
Heart Disease	273	284
No Heart Disease	2244	855
P-value	2.2e-16	

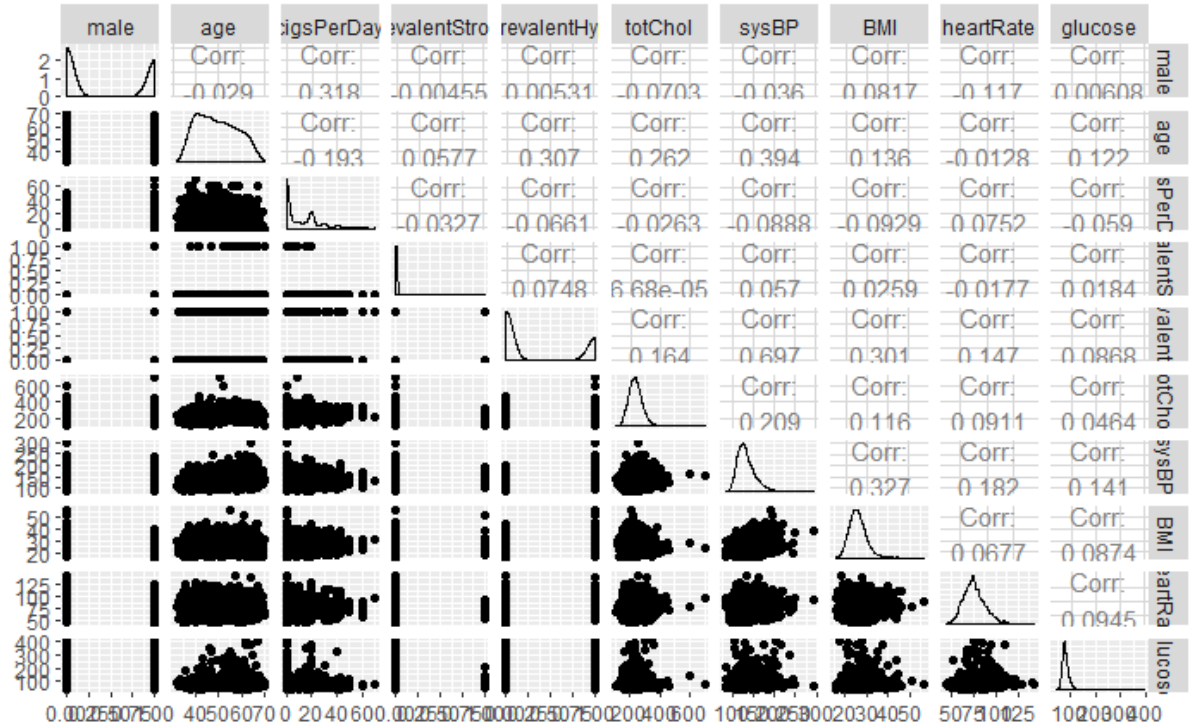
	Variable: Diabetes	
10 Year CHD	0	1
Heart Disease	522	35
No Heart Disease	3035	64
P-value	1.63e-08	

Appendix C: GGpairs plots

All Variables:



Key Variables:



Appendix D: Bestglm outputs

AIC model for *upsampled data*

Model	**Best models**
M1	TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+totChol+sysBP+BMI+heartRate+glucose
M2	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+BMI+glucose
M3	TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+totChol+sysBP+diaBP+BMI+heartRate+glucose
M4	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+glucose
M5	TenYearCHD~male+age+education+cigsPerDay+prevalentStroke+prevalentHyp+totChol+sysBP+BMI+heartRate+glucose

BIC model for *upsampled data*

Model	**Best models**
M1	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+glucose
M2	TenYearCHD~male+age+cigsPerDay+totChol+sysBP+glucose
M3	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+heartRate+glucose
M4	TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+totChol+sysBP+heartRate+glucose
M5	TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+totChol+sysBP+BMI+heartRate+glucose

AIC model for *original train data*

Model	**Best models**
M1	TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+totChol+sysBP+BMI+glucose
M2	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+BMI+glucose
M3	TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp + totChol +sysBP+BMI+heartRate+glucose

M4	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol + sysBP+glucose
M5	TenYearCHD~male+age+education+cigsPerDay+prevalentStroke+prevalent Hyp + totChol +sysBP+BMI+heartRate+glucose

BIC model for *original train data*

Model	**Best models**
M1	TenYearCHD~male+age+cigsPerDay+sysBP+glucose
M2	TenYearCHD~male+age+cigsPerDay+prevalentHyp+sysBP+glucose
M3	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+glucose
M4	TenYearCHD~male+age+sysBP+glucose
M5	TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+BMI+glucose

Appendix F: 10-fold cross validation results

Upsampled data

	AIC Models	**BIC Models**
Model	**Cv Error**	**Cv Error**
M1	0.2078618	0.2086926
M2	0.2084207	0.2086127
M3	0.2082740	0.2083002
M4	0.2087310	0.2081291
M5	0.2079568	0.2079808

Original train data

	AIC Models	**BIC Models**
Model	**Cv Error**	**Cv Error**
M1	0.1147461	0.1151176

M2	0.1147847	0.1147366
M3	0.1147792	0.1143396
M4	0.1144003	0.1156334
M5	0.1149298	0.1148439

Appendix: R Code and Output

An exploration into the framingham dataset - Can we predict heart disease?

Splitting the data

```
set.seed(1)
smp_size = floor(0.80*nrow(framingham))
train_ind = sample(seq_len(nrow(framingham)), size = smp_size)
train = framingham[train_ind,]
test = framingham[-train_ind,]
x_train = train[,-16]
y_train = train$TenYearCHD
```

Resampling Methods

```
set.seed(1)
upsampled_train = upSample(x_train, y_train, list = FALSE, yname = "TenYearCHD")
```

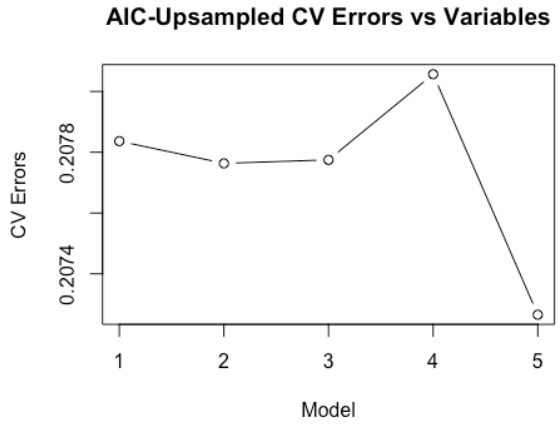
Subset Selection (Using Deviance to choose K best subsets, then ranking according to AIC)

```
## ON UPSAMPLED DATA
aic.upsampled <- bestglm(Xy=upsampled_train,
                         family=binomial,
                         IC="AIC",
                         method = "forward")
## 15 BEST AIC MODELS
aic.upsampled.5.best <- aic.upsampled$Subsets
plot(0:15,aic.upsampled.5.best$AIC,main="AIC-Upsampled vs No. of Variables", xlab="No. of V
ariables", ylab="AIC", type="b")


## ON ORIGINAL DATA
aic.original <- bestglm(Xy=train,
                         family=binomial,
                         IC="AIC",
                         method = "forward")
## 15 BEST AIC MODELS
aic.original.5.best <- aic.original$Subsets
plot(seq(0,15),aic.original.5.best$Subsets[['AIC']], main = 'AIC-Original vs No. of Variabl
es', type = 'b', xlab = 'Number of dependent variables', ylab = 'AIC')
```

Subset Selection (Using Deviance to choose K best subsets, then ranking according to BIC)

```
set.seed(1)
## ON UPSAMPLED DATA
bic.upsampled <- bestglm(Xy=upsampled_train,
                         family=binomial,
                         IC="BIC",
                         method = "forward")
## 15 BEST BIC MODELS
bic.upsampled.5.best <- bic.upsampled$Subsets
plot(0:15,bic.upsampled.5.best$BIC,main="BIC-Upsampled vs No. of Variables", xlab="No. of V
ariables", ylab="BIC", type="b")

## ON ORIGINAL DATA
bic.original <- bestglm(Xy=train,
```

```
                      family=binomial,
                      IC="BIC",
                      method = "forward")
## 15 BEST BIC MODELS
```



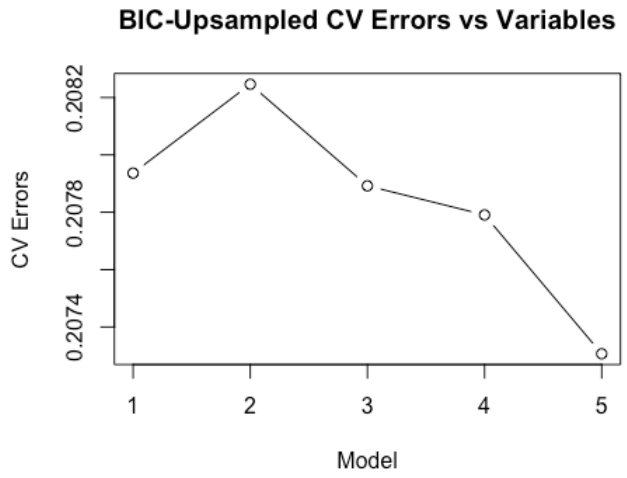

```
bic.original.5.best <- bic.original$Subsets
bic.original.5.best
plot(bic.original$Subsets[['BIC']][0:15], main = 'BIC-Original vs No. of Variables', type =
'b', xlab = 'Number of dependent variables', ylab = 'BIC')
```

CV subset selection

```
## ON UPSAMPLED DATA
set.seed(1)
## 10-FOLD CV FOR 5 BEST AIC MODELS
aic.upsampled.5.best.m1 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+
totChol+sysBP+BMI+heartRate+glucose,data=upsampled_train,family=binomial)
aic.upsampled.5.best.m2 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+BM
I+glucose,data=upsampled_train,family=binomial)
aic.upsampled.5.best.m3 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+
totChol+sysBP+diaBP+BMI+heartRate+glucose,data=upsampled_train,family=binomial)
aic.upsampled.5.best.m4 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+gl
ucose,data=upsampled_train,family=binomial)
aic.upsampled.5.best.m5 <- glm(TenYearCHD~male+age+education+cigsPerDay+prevalentStroke+pre
valentHyp+totChol+sysBP+BMI+heartRate+glucose,data=upsampled_train,family=binomial)

cv.err.aic.upsampled.5.best.m1 <- cv.glm(upsampled_train,aic.upsampled.5.best.m1,K=10)$delt
a[1]
cv.err.aic.upsampled.5.best.m2 <- cv.glm(upsampled_train,aic.upsampled.5.best.m2,K=10)$delt
a[1]
cv.err.aic.upsampled.5.best.m3 <- cv.glm(upsampled_train,aic.upsampled.5.best.m3,K=10)$delt
a[1]
cv.err.aic.upsampled.5.best.m4 <- cv.glm(upsampled_train,aic.upsampled.5.best.m4,K=10)$delt
a[1]
cv.err.aic.upsampled.5.best.m5 <- cv.glm(upsampled_train,aic.upsampled.5.best.m5,K=10)$delt
a[1]
cv.errors.aic.upsampled <- c(cv.err.aic.upsampled.5.best.m1,cv.err.aic.upsampled.5.best.m2,
cv.err.aic.upsampled.5.best.m3,cv.err.aic.upsampled.5.best.m4,cv.err.aic.upsampled.5.best.m
5)
plot(cv.errors.aic.upsampled,main="AIC-Upsampled CV Errors vs Variables", xlab="Model", yla
b="CV Errors", type="b")
```

```
cv.errors.aic.upsampled

## [1] 0.2078365 0.2077634 0.2077749 0.2080566 0.2072655
```

```
## 10-FOLD CV FOR 5 BEST BIC MODELS
bic.upsampled.5.best.m1 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+gl
ucose,data=upsampled_train,family=binomial)
bic.upsampled.5.best.m2 <- glm(TenYearCHD~male+age+cigsPerDay+totChol+sysBP+glucose,data=up
sampled_train,family=binomial)
bic.upsampled.5.best.m3 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+he
artRate+glucose,data=upsampled_train,family=binomial)
bic.upsampled.5.best.m4 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+
totChol+sysBP+heartRate+glucose,data=upsampled_train,family=binomial)
bic.upsampled.5.best.m5 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+
totChol+sysBP+BMI+heartRate+glucose,data=upsampled_train,family=binomial)

cv.err.bic.upsampled.5.best.m1 <- cv.glm(upsampled_train,bic.upsampled.5.best.m1,K=10)$delt
a[1]
cv.err.bic.upsampled.5.best.m2 <- cv.glm(upsampled_train,bic.upsampled.5.best.m2,K=10)$delt
a[1]
cv.err.bic.upsampled.5.best.m3 <- cv.glm(upsampled_train,bic.upsampled.5.best.m3,K=10)$delt
a[1]
cv.err.bic.upsampled.5.best.m4 <- cv.glm(upsampled_train,bic.upsampled.5.best.m4,K=10)$delt
a[1]
cv.err.bic.upsampled.5.best.m5 <- cv.glm(upsampled_train,bic.upsampled.5.best.m5,K=10)$delt
a[1]
cv.errors.bic.upsampled <- c(cv.err.bic.upsampled.5.best.m1,cv.err.bic.upsampled.5.best.m2,
cv.err.bic.upsampled.5.best.m3,cv.err.bic.upsampled.5.best.m4,cv.err.bic.upsampled.5.best.m
5)
plot(cv.errors.bic.upsampled,main="BIC-Upsampled CV Errors vs Variables", xlab="Model", yla
```

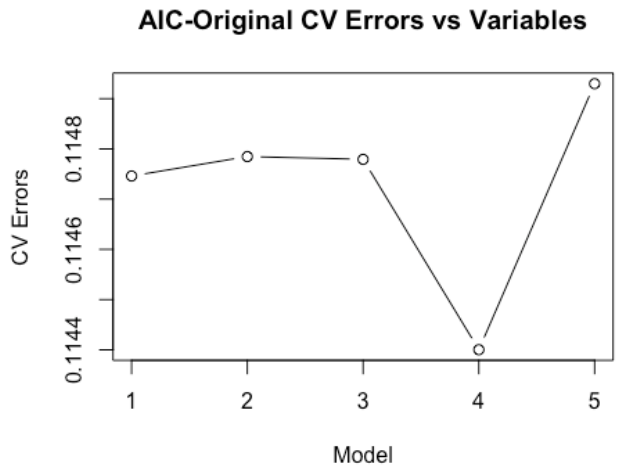



```
b="CV Errors", type="b")
```

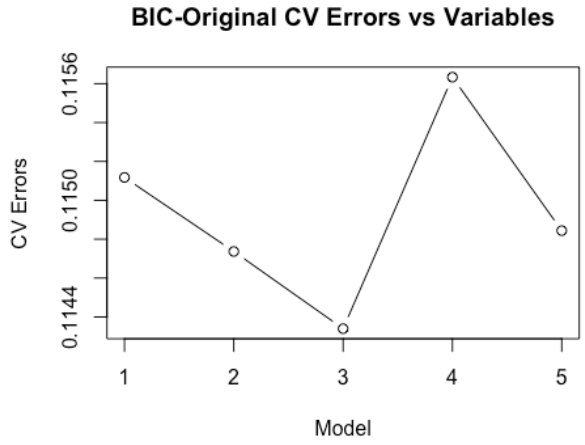
```
cv.errors.bic.upsampled

## [1] 0.2079365 0.2082463 0.2078922 0.2077907 0.2073066

## ON ORIGINAL DATA
## 10-FOLD CV FOR 5 BEST AIC MODELS
aic.original.5.best.m1 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp+t
otChol+sysBP+BMI+glucose,data=train,family=binomial)
aic.original.5.best.m2 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+BMI
+glucose,data=train,family=binomial)
aic.original.5.best.m3 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentStroke+prevalentHyp +
totChol +sysBP+BMI+heartRate+glucose,data=train,family=binomial)
aic.original.5.best.m4 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol + sysBP+g
lucose,data=train,family=binomial)
aic.original.5.best.m5 <- glm(TenYearCHD~male+age+education+cigsPerDay+prevalentStroke+prev
alentHyp + totChol +sysBP+BMI+heartRate+glucose,data=train,family=binomial)
```

```
cv.err.aic.original.5.best.m1 <- cv.glm(train,aic.original.5.best.m1,K=10)$delta[1]
cv.err.aic.original.5.best.m2 <- cv.glm(train,aic.original.5.best.m2,K=10)$delta[1]
cv.err.aic.original.5.best.m3 <- cv.glm(train,aic.original.5.best.m3,K=10)$delta[1]
cv.err.aic.original.5.best.m4 <- cv.glm(train,aic.original.5.best.m4,K=10)$delta[1]
cv.err.aic.original.5.best.m5 <- cv.glm(train,aic.original.5.best.m5,K=10)$delta[1]
cv.errors.aic.original <- c(cv.err.aic.original.5.best.m1,cv.err.aic.original.5.best.m2,cv.
err.aic.original.5.best.m3,cv.err.aic.original.5.best.m4,cv.err.aic.original.5.best.m5)
plot(cv.errors.aic.original,main="AIC-Original CV Errors vs Variables", xlab="Model", ylab=
"CV Errors", type="b")
```

```
cv.errors.aic.original
```




```
## [1] 0.1147461 0.1147847 0.1147792 0.1144003 0.1149298

## 10-FOLD CV FOR 5 BEST BIC MODELS
bic.original.5.best.m1 <- glm(TenYearCHD~male+age+cigsPerDay+sysBP+glucose,data=train,famil
y=binomial)
bic.original.5.best.m2 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+sysBP+glucose,dat
a=train,family=binomial)
bic.original.5.best.m3 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+glu
cose,data=train,family=binomial)
bic.original.5.best.m4 <- glm(TenYearCHD~male+age+sysBP+glucose,data=train,family=binomial)
bic.original.5.best.m5 <- glm(TenYearCHD~male+age+cigsPerDay+prevalentHyp+totChol+sysBP+BMI
+glucose,data=train,family=binomial)

cv.err.bic.original.5.best.m1 <- cv.glm(train,bic.original.5.best.m1,K=10)$delta[1]
cv.err.bic.original.5.best.m2 <- cv.glm(train,bic.original.5.best.m2,K=10)$delta[1]
cv.err.bic.original.5.best.m3 <- cv.glm(train,bic.original.5.best.m3,K=10)$delta[1]
cv.err.bic.original.5.best.m4 <- cv.glm(train,bic.original.5.best.m4,K=10)$delta[1]
cv.err.bic.original.5.best.m5 <- cv.glm(train,bic.original.5.best.m5,K=10)$delta[1]
cv.errors.bic.original <- c(cv.err.bic.original.5.best.m1,cv.err.bic.original.5.best.m2,cv.
err.bic.original.5.best.m3,cv.err.bic.original.5.best.m4,cv.err.bic.original.5.best.m5)
plot(cv.errors.bic.original,main="BIC-Original CV Errors vs Variables", xlab="Model", ylab=
"CV Errors", type="b")
```




```
cv.errors.bic.original

## [1] 0.1151176 0.1147366 0.1143396 0.1156334 0.1148439

which.min(cv.errors.bic.original)

## [1] 3

bic_models = c('bic.original.5.best.m1','bic.original.5.best.m2','bic.original.5.best.m3',
'bic.original.5.best.m4','bic.original.5.best.m5')
aic_models = c('aic.original.5.best.m1','aic.original.5.best.m2','aic.original.5.best.m3','
aic.original.5.best.m4','aic.original.5.best.m5')
bic_up = c('bic.upsampled.5.best.m1','bic.upsampled.5.best.m2','bic.upsampled.5.best.m3','b
ic.upsampled.5.best.m4','bic.upsampled.5.best.m5')
aic_up = c('aic.upsampled.5.best.m1','aic.upsampled.5.best.m2','aic.upsampled.5.best.m3','a
ic.upsampled.5.best.m4','aic.upsampled.5.best.m5')


## BEST 4 MODELS
## AIC UPSAMPLED
aic.upsampled.final <- get(aic_up[which.min(cv.errors.aic.upsampled)])
summary(aic.upsampled.final)

##
## Call:
## glm(formula = TenYearCHD ~ male + age + education + cigsPerDay +
##     prevalentStroke + prevalentHyp + totChol + sysBP + BMI +
##     heartRate + glucose, family = binomial, data = upsampled_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.20990  -1.00293   0.09434   1.00317   2.40601
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       6.8369165  0.4263133  16.037  < 2e-16 ***
## male1            -0.4866679  0.0676158  -7.198 6.13e-13 ***
## age              -0.0678148  0.0041954 -16.164  < 2e-16 ***
## education         0.0776916  0.0311995   2.490 0.012769 *
## cigsPerDay       -0.0235445  0.0028480  -8.267  < 2e-16 ***
## prevalentStroke1 -0.6994552  0.3940311  -1.775 0.075877 .
## prevalentHyp1    -0.3276252  0.0881357  -3.717 0.000201 ***
## totChol          -0.0031711  0.0007015  -4.521 6.16e-06 ***
## sysBP            -0.0103405  0.0019924  -5.190 2.10e-07 ***
## BMI              -0.0199136  0.0076503  -2.603 0.009242 **
## heartRate         0.0041057  0.0026812   1.531 0.125693
## glucose          -0.0059702  0.0013009  -4.589 4.45e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6887.1  on 4967  degrees of freedom
## Residual deviance: 5953.1  on 4956  degrees of freedom
## AIC: 5977.1
##
## Number of Fisher Scoring iterations: 4

## AIC ORIGINAL
aic.original.final <- get(aic_models[which.min(cv.errors.aic.original)])
summary(aic.original.final)

##
## Call:
## glm(formula = TenYearCHD ~ male + age + cigsPerDay + prevalentHyp +
##     totChol + sysBP + glucose, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8604   0.2836   0.4270   0.5946   2.1168
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    8.662719   0.588510  14.720  < 2e-16 ***
## male1         -0.555830   0.120815  -4.601 4.21e-06 ***
## age           -0.066173   0.007170  -9.229  < 2e-16 ***
## cigsPerDay    -0.020691   0.004695  -4.407 1.05e-05 ***
## prevalentHyp1 -0.280812   0.151466  -1.854 0.063746 .
## totChol       -0.002243   0.001263  -1.776 0.075757 .
## sysBP         -0.012401   0.003268  -3.795 0.000148 ***
## glucose       -0.008375   0.001925  -4.350 1.36e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2476.9  on 2923  degrees of freedom
## Residual deviance: 2187.9  on 2916  degrees of freedom
## AIC: 2203.9
##
## Number of Fisher Scoring iterations: 5

## BIC UPSAMPLED
bic.upsampled.final <- get(bic_up[which.min(cv.errors.bic.upsampled)])
summary(bic.upsampled.final)

##
## Call:
## glm(formula = TenYearCHD ~ male + age + cigsPerDay + prevalentStroke +
##     prevalentHyp + totChol + sysBP + BMI + heartRate + glucose,
##     family = binomial, data = upsampled_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2134  -1.0068   0.1013   1.0022   2.4541
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       7.1299467  0.4107047  17.360  < 2e-16 ***
## male1            -0.4770628  0.0674236  -7.076 1.49e-12 ***
## age              -0.0691716  0.0041622 -16.619  < 2e-16 ***
## cigsPerDay       -0.0238571  0.0028473  -8.379  < 2e-16 ***
## prevalentStroke1 -0.7286898  0.3928961  -1.855 0.063644 .
```

```
## prevalentHyp1    -0.3132237  0.0878796  -3.564 0.000365 ***
## totChol          -0.0029874  0.0006964  -4.290 1.79e-05 ***
## sysBP            -0.0106576  0.0019871  -5.364 8.16e-08 ***
## BMI              -0.0221842  0.0075939  -2.921 0.003485 **
## heartRate         0.0038451  0.0026786   1.436 0.151137
## glucose          -0.0060223  0.0013031  -4.621 3.81e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6887.1  on 4967  degrees of freedom
## Residual deviance: 5959.3  on 4957  degrees of freedom
## AIC: 5981.3
##
## Number of Fisher Scoring iterations: 4

## BIC ORIGINAL

bic.original.final <- get(bic_models[which.min(cv.errors.bic.original)])
summary(bic.original.final)

##
## Call:
## glm(formula = TenYearCHD ~ male + age + cigsPerDay + prevalentHyp +
##     totChol + sysBP + glucose, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8604   0.2836   0.4270   0.5946   2.1168
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    8.662719   0.588510  14.720  < 2e-16 ***
## male1         -0.555830   0.120815  -4.601 4.21e-06 ***
## age           -0.066173   0.007170  -9.229  < 2e-16 ***
## cigsPerDay    -0.020691   0.004695  -4.407 1.05e-05 ***
## prevalentHyp1 -0.280812   0.151466  -1.854 0.063746 .
## totChol       -0.002243   0.001263  -1.776 0.075757 .
## sysBP         -0.012401   0.003268  -3.795 0.000148 ***
## glucose       -0.008375   0.001925  -4.350 1.36e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2476.9  on 2923  degrees of freedom
## Residual deviance: 2187.9  on 2916  degrees of freedom
## AIC: 2203.9
##
## Number of Fisher Scoring iterations: 5
```

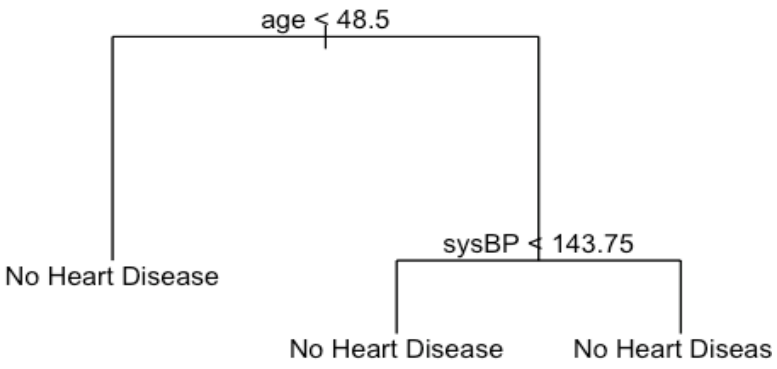
Creating Functions for Ensemble Learning

```
cv_pruned = function(model_name, size){
  cv_model = cv.tree(get(model_name), K =10, FUN = prune.misclass)
  print(cv_model)
  plot(cv_model$size, cv_model$dev, type = "b", main = "Cross Validation: Deviance versus s
ize",  ylab = 'Number of obs misclassified', xlab = 'Number of Nodes')
  nn = cv_model$size[which.min(cv_model$dev)]
  if (size == 0){
    prune_model = prune.misclass(get(model_name), best = nn)
    plot(prune_model)
    title("Pruned Classification Tree for Framingham Data Set")
    text(prune_model, pretty = 0)
```

```
  }
  else {
    prune_model = prune.misclass(get(model_name), best = size)
    plot(prune_model)
    title("Pruned Classification Tree for Framingham Data Set")
    text(prune_model, pretty = 0)
  }
  return(prune_model)
}

plotting.oob = function(model_name){
  oob.error.data = data.frame(Trees = rep(1:nrow(get(model_name)$err.rate), times = 3),
                              Type = rep(c("OOB", "No Heart Disease","Heart Disease"), each =
nrow(get(model_name)$err.rate)),
                              Error = c(get(model_name)$err.rate[,'OOB'],
                                        get(model_name)$err.rate[,'No Heart Disease'],
                                        get(model_name)$err.rate[,'Heart Disease']))
  ggplot(data = oob.error.data, aes(x = Trees, y = Error)) +
        geom_line(aes(color = Type))

}

analyse_train_confusion = function(cmatrix){
  m2 = cmatrix[1,1]/(cmatrix[1,1]+ cmatrix[2,1])
  m3 = cmatrix[2,2]/(cmatrix[1,2]+cmatrix[2,2])
  return(list('sens' = m2, 'spec' = m3))
}
```

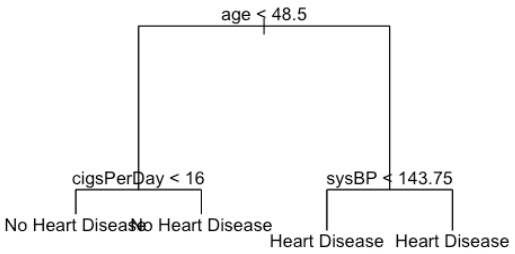
Decision Tree on Original Training Set

```
tree.framingham = tree(TenYearCHD~., data =train, y = TRUE, model = TRUE)
summary(tree.framingham)

##
## Classification tree:
## tree(formula = TenYearCHD ~ ., data = train, model = TRUE, y = TRUE)
## Variables actually used in tree construction:
## [1] "age"   "sysBP"
## Number of terminal nodes:  3
## Residual mean deviance:  0.789 = 2305 / 2921
## Misclassification error rate: 0.1505 = 440 / 2924

plot(tree.framingham)
text(tree.framingham, pretty = 0)
```



Decision Tree on Upsampled Training Set

```
# INITIAL TREES AFTER RESAMPLING
set.seed(1)
tree.framingham.upsampled = tree(TenYearCHD~., data =upsampled_train, y = TRUE, model = TRU
E)
summary(tree.framingham.upsampled)

##
## Classification tree:
## tree(formula = TenYearCHD ~ ., data = upsampled_train, model = TRUE,
##     y = TRUE)
## Variables actually used in tree construction:
## [1] "age"        "cigsPerDay" "sysBP"
## Number of terminal nodes:  4
## Residual mean deviance:  1.253 = 6222 / 4964
## Misclassification error rate: 0.3508 = 1743 / 4968

plot(tree.framingham.upsampled)
text(tree.framingham.upsampled, pretty = 0)
```

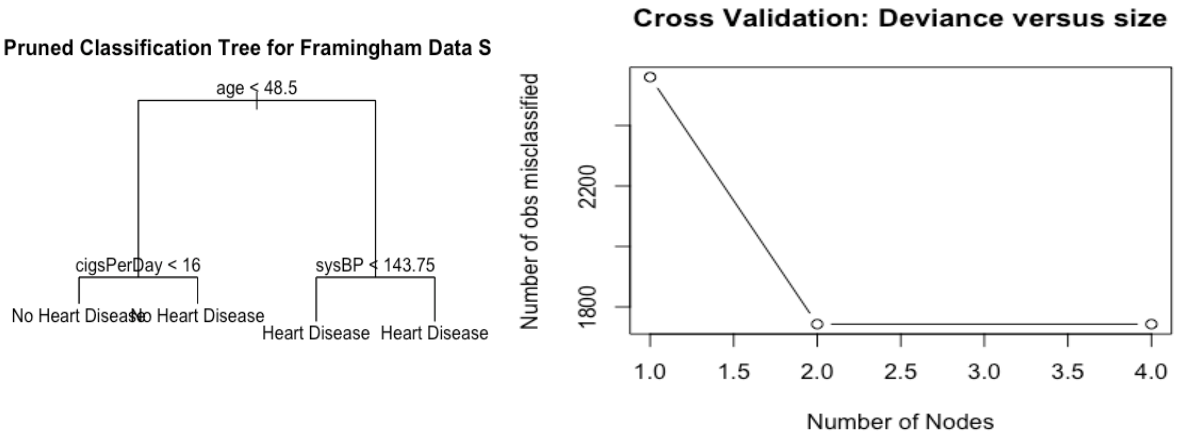



```
# APPLYING CV ANALYSIS
result.upsampled = cv_pruned('tree.framingham.upsampled',0)

## $size
## [1] 4 2 1
```

```
##
## $dev
## [1] 1743 1743 2560
##
## $k
## [1] -Inf    0  741
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
summary(result.upsampled)
```

```
##
## Classification tree:
## tree(formula = TenYearCHD ~ ., data = upsampled_train, model = TRUE,
##     y = TRUE)
## Variables actually used in tree construction:
## [1] "age"        "cigsPerDay" "sysBP"
## Number of terminal nodes:  4
## Residual mean deviance:  1.253 = 6222 / 4964
## Misclassification error rate: 0.3508 = 1743 / 4968
```

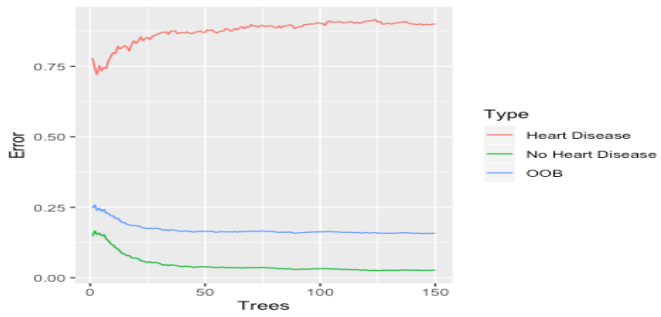


Bagging

```
set.seed(1)
bag.framingham = randomForest(TenYearCHD~., data = train, mtry = 15,
                              importance = TRUE, ntree = 150)
bag.framingham.upsampled = randomForest(TenYearCHD~., data = upsampled_train, mtry = 15,
                              importance = TRUE, ntree = 200)


bag.framingham
```

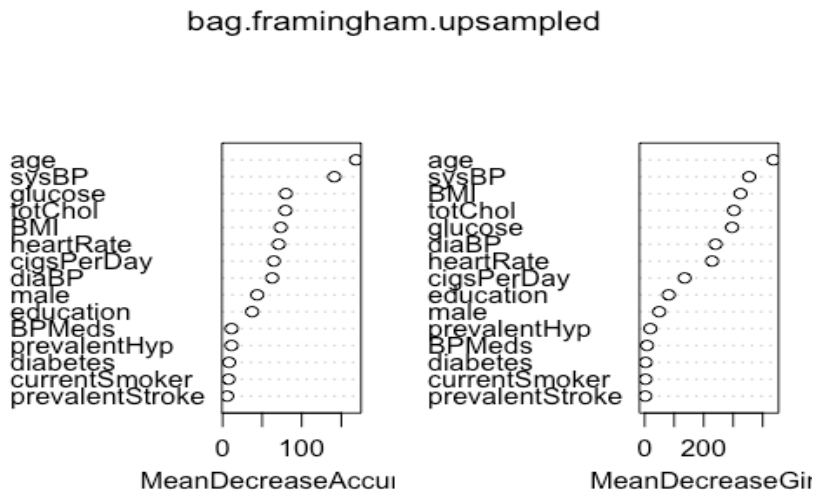
```
##
## Call:
##  randomForest(formula = TenYearCHD ~ ., data = train, mtry = 15,      importance = TRUE,
ntree = 150)
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 15
##
##         OOB estimate of  error rate: 15.83%
## Confusion matrix:
```

```
##                  Heart Disease No Heart Disease class.error
## Heart Disease               44              396  0.90000000
## No Heart Disease            67             2417  0.02697262

bag.framingham.upsampled

##
## Call:
##  randomForest(formula = TenYearCHD ~ ., data = upsampled_train,      mtry = 15, importan
ce = TRUE, ntree = 200)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 15
##
##         OOB estimate of  error rate: 2.94%
## Confusion matrix:
##                  Heart Disease No Heart Disease class.error
## Heart Disease             2480                4 0.001610306
## No Heart Disease           142             2342 0.057165862

analyse_train_confusion(bag.framingham$confusion)

## $sens
## [1] 0.3963964
##
## $spec
## [1] 0.859225

analyse_train_confusion(bag.framingham.upsampled$confusion)

## $sens
## [1] 0.9458429
##
## $spec
## [1] 0.998295

# PLOTTING OOB ERROR DATA
plotting.oob('bag.framingham')
```




```
plotting.oob('bag.framingham.upsampled')
```


```
# IMPORTANCE
importance(bag.framingham.upsampled)

##                 Heart Disease No Heart Disease MeanDecreaseAccuracy
## male                43.740300        8.1956631             43.746634
## age                189.199879       13.6746354            168.537036
## education           37.742396        3.4560269             37.357143
## currentSmoker        7.745625       -0.9622198              7.559888
## cigsPerDay          64.861172        4.8757584             64.781062
## BPMeds              10.250450        4.5759277             11.343285
## prevalentStroke      6.343859       -0.5638774              5.808750
## prevalentHyp        11.235023       -1.1180555             11.302770
## diabetes             8.239474       -2.5186434              8.281103
## totChol             82.341061        4.7382053             79.459109
## sysBP              143.012852       16.8478800            141.191476
## diaBP               62.452347        9.4682886             62.635986
## BMI                 73.809475        4.4608554             73.413500
## heartRate           72.784854        3.8289527             70.935477
## glucose             80.435588        9.7650140             79.913169
##                 MeanDecreaseGini
## male                   49.354610
## age                   436.235114
## education              81.385014
## currentSmoker           2.751542
## cigsPerDay            135.405870
## BPMeds                  8.001491
## prevalentStroke         2.073678
## prevalentHyp           18.936027
## diabetes                3.161459
## totChol               302.786847
## sysBP                 354.079053
## diaBP                 240.114879
## BMI                   324.801360
## heartRate             228.288349
## glucose               296.043953

varImpPlot(bag.framingham.upsampled)
```



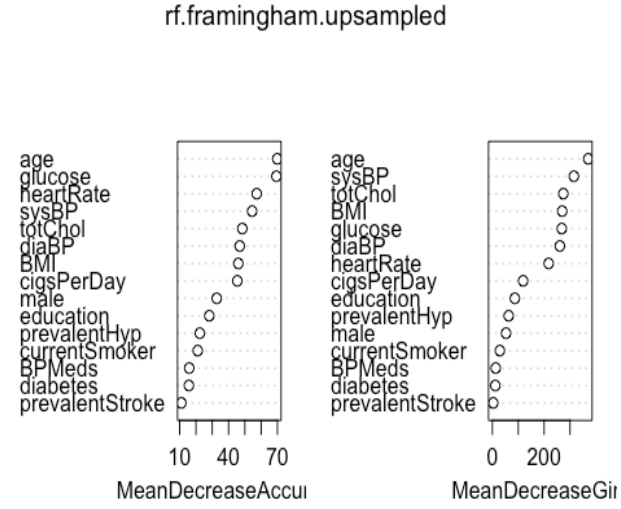
Random Forest

```
set.seed(1)
# ORIGINAL DATA
rf.framingham = randomForest(TenYearCHD~., data = train, ntree = 200)
rf.framingham.upsampled = randomForest(TenYearCHD~., data = upsampled_train, ntree= 200, im
portance = TRUE)

rf.framingham

##
## Call:
##  randomForest(formula = TenYearCHD ~ ., data = train, ntree = 200)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14.91%
## Confusion matrix:
##                  Heart Disease No Heart Disease class.error
## Heart Disease               30              410  0.93181818
## No Heart Disease            26             2458  0.01046699

rf.framingham.upsampled

##
## Call:
##  randomForest(formula = TenYearCHD ~ ., data = upsampled_train,      ntree = 200, import
ance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 2.23%
## Confusion matrix:
##                  Heart Disease No Heart Disease class.error
## Heart Disease             2478                6 0.002415459
## No Heart Disease           105             2379 0.042270531
```

```
importance(rf.framingham.upsampled)
```

```
##                 Heart Disease No Heart Disease MeanDecreaseAccuracy
## male                     32.90307      4.109757381              32.73275
## age                      69.53346     13.880822862              69.89361
## education                28.26064      2.112482315              28.16272
## currentSmoker            20.80603     -1.460360342              21.07643
## cigsPerDay               45.15983      3.169491296              45.37925
## BPMeds                   16.08587      3.637449802              15.85655
## prevalentStroke          11.79749     -0.000953457              11.03107
## prevalentHyp             22.08582      7.623439577              22.37277
## diabetes                 15.29193      0.990168525              15.65892
## totChol                  49.71379      4.004223774              48.42977
## sysBP                    53.82635     12.415456113              54.43045
## diaBP                    45.71666     11.155059278              46.85953
## BMI                      46.24443      4.473171992              45.99622
## heartRate                57.60150      1.849259008              57.28556
## glucose                  71.28729      7.017250024              69.34195
##                 MeanDecreaseGini
## male                   52.835351
## age                   370.558555
## education              86.913090
## currentSmoker          28.989641
## cigsPerDay            118.743303
## BPMeds                 12.934407
## prevalentStroke         3.426312
## prevalentHyp           62.976130
## diabetes               11.473894
## totChol               274.120815
## sysBP                 314.937205
## diaBP                 260.724387
## BMI                   269.868099
## heartRate             217.634365
## glucose               268.191460
```

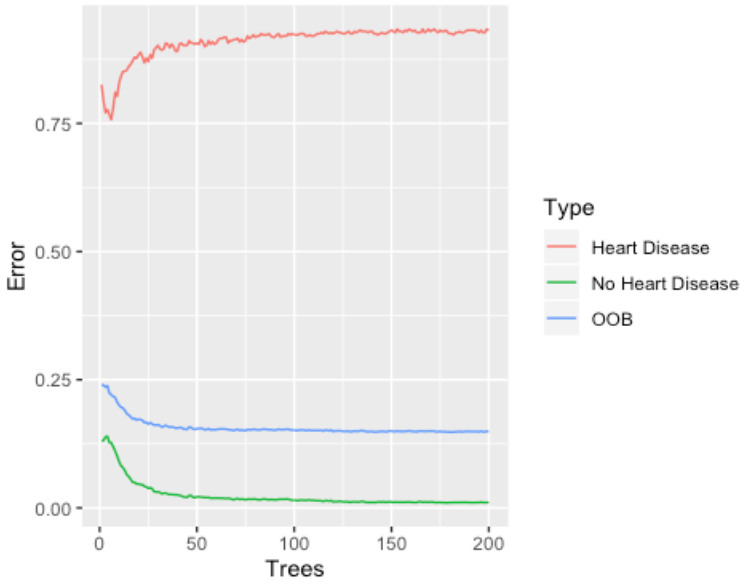
```
varImpPlot(rf.framingham.upsampled)
```



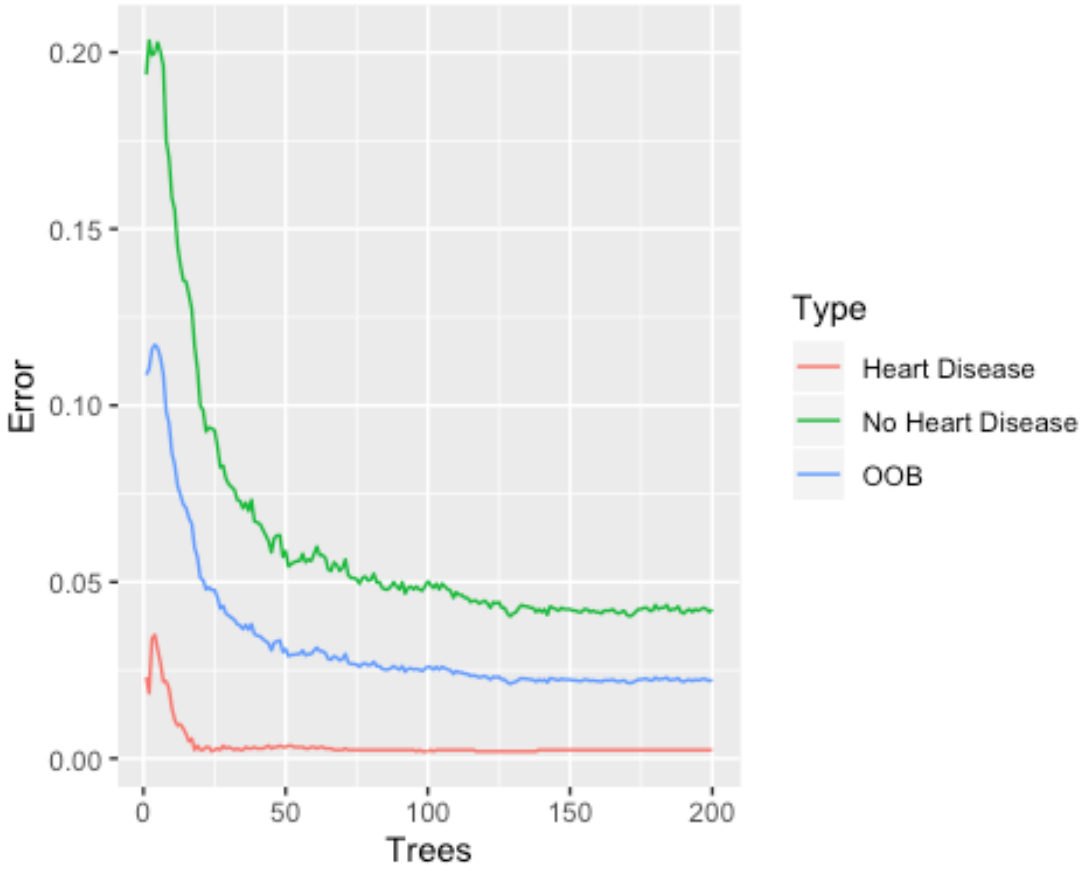

```
analyse_train_confusion(rf.framingham$confusion)
```

```
## $sens
## [1] 0.5357143
##
## $spec
## [1] 0.8570432
```

```
analyse_train_confusion(rf.framingham.upsampled$confusion)

## $sens
## [1] 0.9593496
##
## $spec
## [1] 0.9974843

plotting.oob('rf.framingham')
```




```
plotting.oob('rf.framingham.upsampled')
```



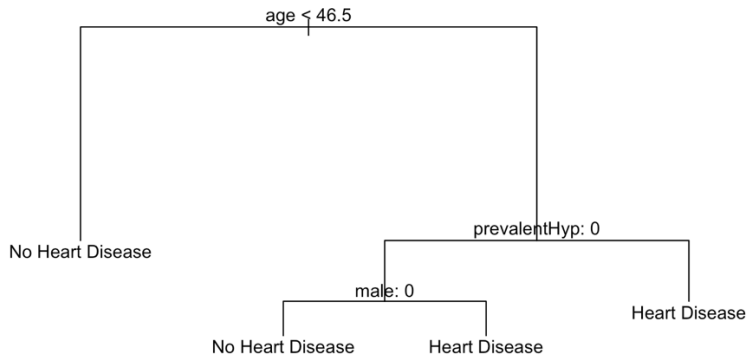
Performance On Test Data

```
set.seed(1)
results_df = data.frame()
models = c('tree.framingham.upsampled','bag.framingham.upsampled','rf.framingham.upsampled'
,'aic.upsampled.final', 'aic.original.final', 'bic.upsampled.final', 'bic.original.final',
'rf.framingham')

get_metrics = function(model_name, num){
  if ((startsWith(model_name,'aic') | (startsWith(model_name,'bic')))){
    model_pred = predict(get(model_name), test, type = "response")
    model_pred <- ifelse(model_pred > 0.5, "Heart Disease", "No Heart Disease")
    cmatrix = table(model_pred, test$TenYearCHD)
  }
  else {
    model_pred = predict(get(model_name), test, type = "class")
    cmatrix = table(model_pred, test$TenYearCHD)
  }
  # print(model_name)
  # print(cmatrix)
  m1 = cmatrix[2,2]/(cmatrix[1,2]+cmatrix[2,2])
  m2 = cmatrix[1,1]/(cmatrix[1,1]+ cmatrix[2,1])
  m3 = (cmatrix[1,2]+cmatrix[2,1])/nrow(test)
  return(data.frame('model' = model_name, 'err' = m3, 'sens' = m2, 'spec' = m1))
}

for (i in 1:length(models)){
  results_df = rbind(get_metrics(models[i]), results_df)
}

print(results_df)
```

```
##                        model       err       sens        spec
## 1            rf.framingham 0.1598361 0.06837607 0.986991870
## 2        bic.original.final 0.8456284 0.94871795 0.003252033
## 3       bic.upsampled.final 0.6489071 0.31623932 0.357723577
## 4        aic.original.final 0.8456284 0.94871795 0.003252033
## 5       aic.upsampled.final 0.6571038 0.30769231 0.349593496
## 6   rf.framingham.upsampled 0.1612022 0.14529915 0.970731707
## 7  bag.framingham.upsampled 0.1762295 0.17094017 0.947967480
## 8 tree.framingham.upsampled 0.4303279 0.74358974 0.536585366
```

Final Model

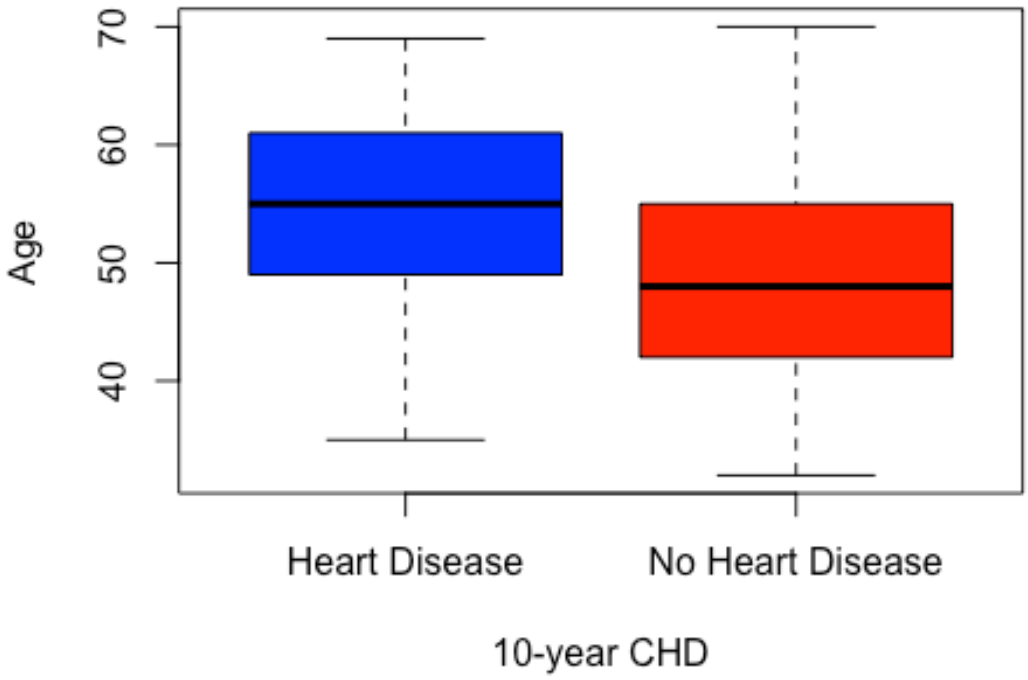
```
## Upsampling framingham dataset

framingham_x = framingham[,-16]

framingham_y = framingham$TenYearCHD

framingham_upsampled <- upSample(framingham_x, framingham_y, list = FALSE, yname = "TenYearCHD")



## Refitting Trees model

tree.framingham.final = tree(TenYearCHD~., data =framingham_upsampled, y = TRUE, model = TRUE)

summary(tree.framingham.upsampled)

plot(tree.framingham.upsampled)

text(tree.framingham.upsampled, pretty = 0)
```



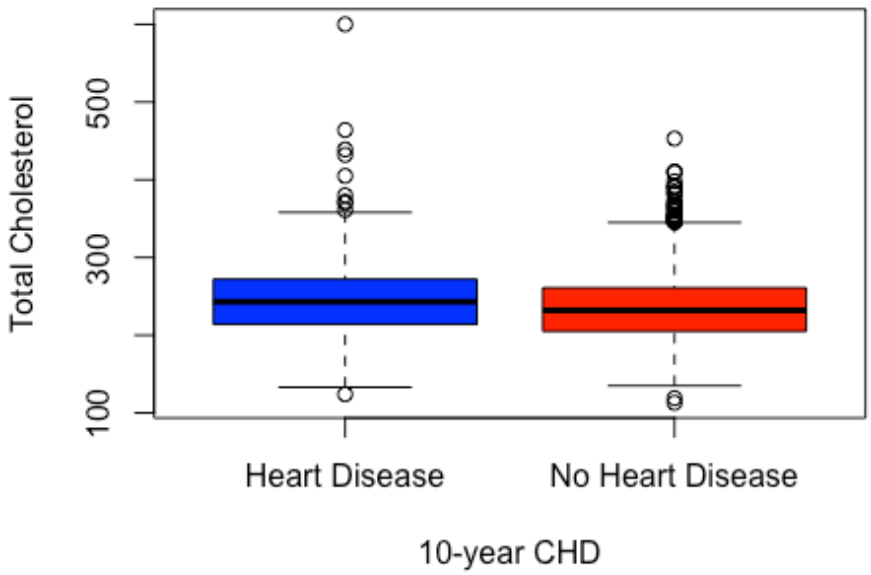
Exploration

```
attach(framingham)
male.table <- table(TenYearCHD, male)
male.table
```

```
##                   male
## TenYearCHD           0    1
##   Heart Disease     250  307
##   No Heart Disease 1784 1315

age.boxplot <- boxplot(framingham$age~TenYearCHD,col=c("blue","red"),
                       xlab="10-year CHD",ylab="Age")
```

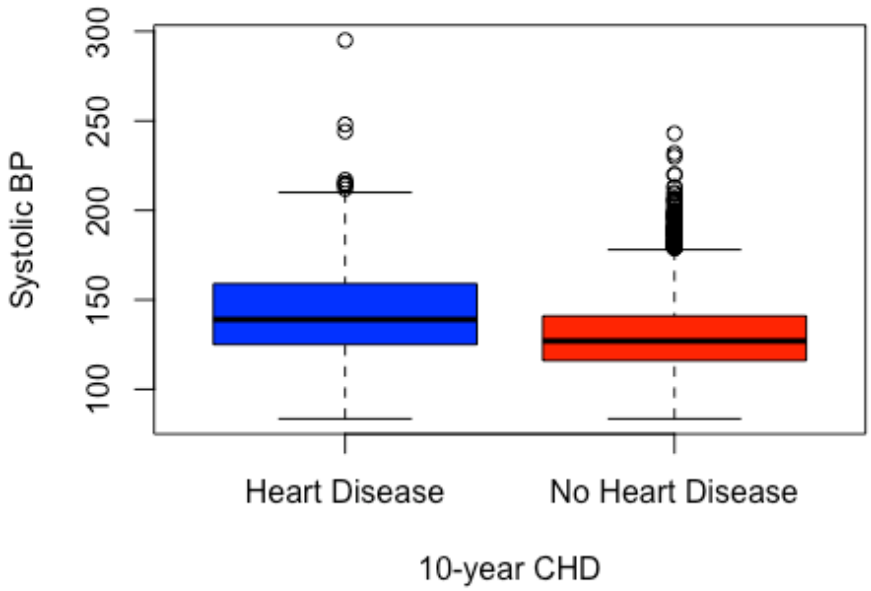



```
education.table <- table(TenYearCHD,education)
education.table

##                   education
## TenYearCHD           1    2    3    4
##   Heart Disease     291  131   75   60
##   No Heart Disease 1235  970  531  363

currentsmoker.table <- table(TenYearCHD,currentSmoker)
currentsmoker.table

##                   currentSmoker
## TenYearCHD           0    1
##   Heart Disease     272  285
##   No Heart Disease 1596 1503

cigsperday.boxplot <- boxplot(cigsPerDay~TenYearCHD,col=c("blue","red"),
                       xlab="10-year CHD",ylab="Cigs Per Day")
```

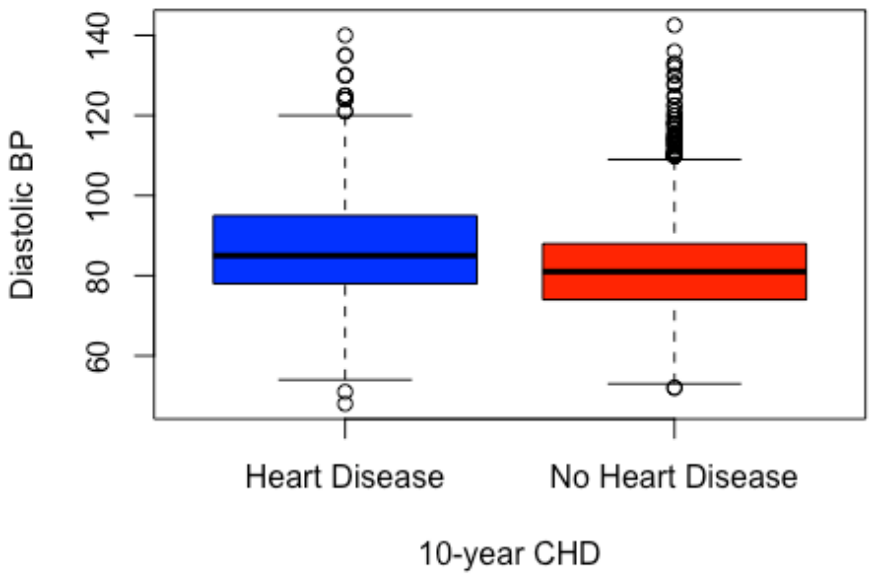



```
bpmeds.table <- table(TenYearCHD,BPMeds)
bpmeds.table

##                   BPMeds
## TenYearCHD            0    1
##   Heart Disease     520   37
##   No Heart Disease 3025   74

prevalentstroke.table <- table(TenYearCHD,prevalentStroke)
prevalentstroke.table

##                   prevalentStroke
## TenYearCHD            0    1
##   Heart Disease     549    8
##   No Heart Disease 3086   13

prevalenthyp.table <- table(TenYearCHD,prevalentHyp)
prevalenthyp.table

##                   prevalentHyp
## TenYearCHD            0    1
##   Heart Disease     273  284
##   No Heart Disease 2244  855

diabetes.table <- table(TenYearCHD,diabetes)
diabetes.table

##                   diabetes
## TenYearCHD            0    1
##   Heart Disease     522   35
##   No Heart Disease 3035   64

totchol.boxplot <- boxplot(totChol~TenYearCHD,col=c("blue","red"),
                      xlab="10-year CHD",ylab="Total Cholesterol")
```

```
sysbp.boxplot <- boxplot(sysBP~TenYearCHD,col=c("blue","red"),
                         xlab="10-year CHD",ylab="Systolic BP")
```

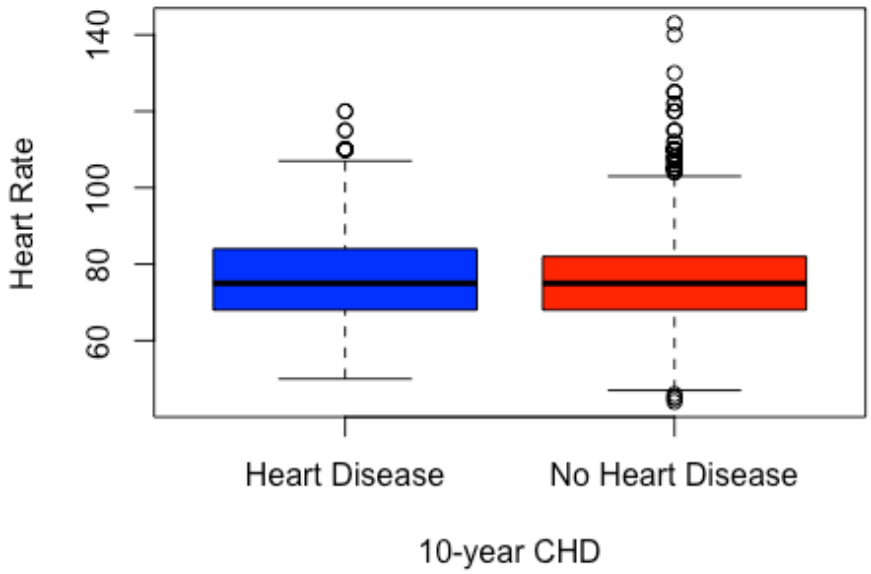



```
diabp.boxplot <- boxplot(diaBP~TenYearCHD,col=c("blue","red"),
                         xlab="10-year CHD",ylab="Diastolic BP")
```



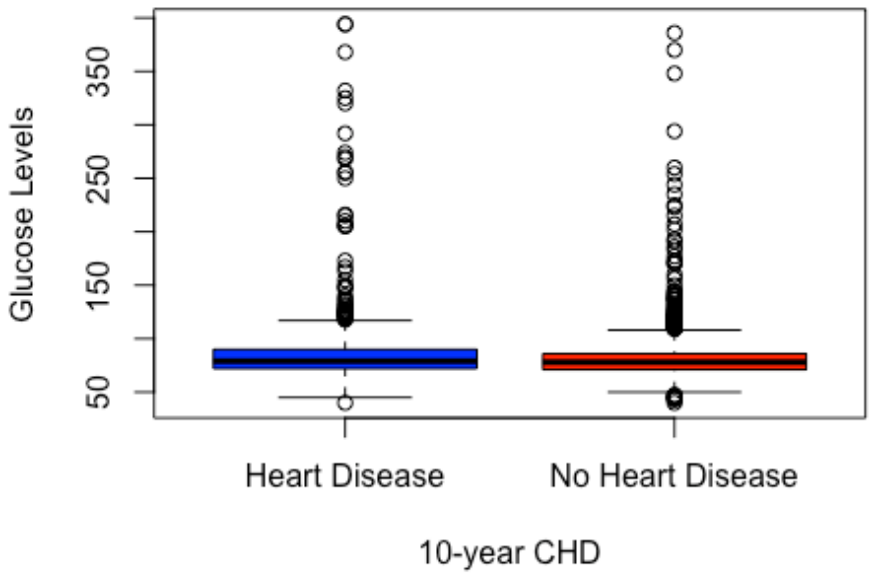

```
## [1] "Heart Disease"    "No Heart Disease"
```

```
bmi.boxplot <- boxplot(BMI~TenYearCHD,col=c("blue","red"),
                       xlab="10-year CHD",ylab="BMI")
```




```
bmi.boxplot

heartrate.boxplot <- boxplot(heartRate~TenYearCHD,col=c("blue","red"),
                       xlab="10-year CHD",ylab="Heart Rate")
```

```
glucose.boxplot <- boxplot(glucose~TenYearCHD,col=c("blue","red"),
                       xlab="10-year CHD",ylab="Glucose Levels")
```



Proportion Testing

```
male.proptest <- prop.test(c(250,307),n=c(2034,1622),alternative="two.sided",conf.level=0.99,correct=FALSE)
male.proptest

##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(250, 307) out of c(2034, 1622)
## X-squared = 30.773, df = 1, p-value = 2.9e-08
## alternative hypothesis: two.sided
## 99 percent confidence interval:
##  -0.09765647 -0.03506749
## sample estimates:
##    prop 1    prop 2
## 0.1229105 0.1892725

education.proptest <- prop.test(c(291,131,75,60),n=c(1526,1101,606,423),alternative="two.sided",conf.level=0.99,correct=FALSE)
education.proptest

##
##  4-sample test for equality of proportions without continuity
##  correction
##
## data:  c(291, 131, 75, 60) out of c(1526, 1101, 606, 423)
## X-squared = 31.063, df = 3, p-value = 8.246e-07
## alternative hypothesis: two.sided
## sample estimates:
##    prop 1    prop 2    prop 3    prop 4
## 0.1906946 0.1189827 0.1237624 0.1418440
```

```r
currentsmoker.proptest <- prop.test(c(272,285),n=c(1868,1788),alternative="two.sided",conf.level=0.99,correct=FALSE)
currentsmoker.proptest
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(272, 285) out of c(1868, 1788)
## X-squared = 1.3444, df = 1, p-value = 0.2463
## alternative hypothesis: two.sided
## 99 percent confidence interval:
##  -0.04443020  0.01685881
## sample estimates:
##    prop 1    prop 2
## 0.1456103 0.1593960
```

```r
bpmeds.proptest <- prop.test(c(520,37),n=c(3545,111),alternative="two.sided",conf.level=0.99,correct=FALSE)
bpmeds.proptest
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(520, 37) out of c(3545, 111)
## X-squared = 29.035, df = 1, p-value = 7.11e-08
## alternative hypothesis: two.sided
## 99 percent confidence interval:
##  -0.30291198 -0.07038374
## sample estimates:
##    prop 1    prop 2
## 0.1466855 0.3333333
```

```r
prevalentstroke.proptest <- prop.test(c(549,8),n=c(3635,21),alternative="two.sided",conf.level=0.99,correct=FALSE)
```

```
## Warning in prop.test(c(549, 8), n = c(3635, 21), alternative =
## "two.sided", : Chi-squared approximation may be incorrect
```

```r
prevalentstroke.proptest
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(549, 8) out of c(3635, 21)
## X-squared = 8.5469, df = 1, p-value = 0.003461
## alternative hypothesis: two.sided
## 99 percent confidence interval:
##  -0.50331275  0.04347126
## sample estimates:
##    prop 1    prop 2
## 0.1510316 0.3809524
```

```r
prevalenthyp.proptest <- prop.test(c(273,284),n=c(2517,1139),alternative="two.sided",conf.level=0.99,correct=FALSE)
prevalenthyp.proptest
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(273, 284) out of c(2517, 1139)
## X-squared = 120.51, df = 1, p-value < 2.2e-16
## alternative hypothesis: two.sided
```

```
## 99 percent confidence interval:
##  -0.1775561 -0.1042021
## sample estimates:
##    prop 1    prop 2
## 0.1084625 0.2493415

diabetes.proptest <- prop.test(c(522,35),n=c(3557,99),alternative="two.sided",conf.level=0.
99,correct=FALSE)
diabetes.proptest

##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(522, 35) out of c(3557, 99)
## X-squared = 31.892, df = 1, p-value = 1.63e-08
## alternative hypothesis: two.sided
## 99 percent confidence interval:
##  -0.33148479 -0.08208015
## sample estimates:
##    prop 1    prop 2
## 0.1467529 0.3535354
```

Data Exploration: 2 Sample t-test for non binary variables

```
data.with.TYCHD <- filter(framingham,TenYearCHD=="No Heart Disease")
data.without.TYCHD <- filter(framingham,TenYearCHD=="Heart Disease")

age.with.TYCHD <- data.with.TYCHD$age
age.without.TYCHD <- data.without.TYCHD$age
age.ttest <- t.test(age.without.TYCHD,age.with.TYCHD,alternative="two.sided",mu=0,var.equal
=FALSE,conf.level=0.99)
age.ttest

##
##  Welch Two Sample t-test
##
## data:  age.without.TYCHD and age.with.TYCHD
## t = 15.027, df = 792.08, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  4.612378 6.526298
## sample estimates:
## mean of x mean of y
##  54.27828  48.70894

sysbp.with.TYCHD <- data.with.TYCHD$sysBP
sysbp.without.TYCHD <- data.without.TYCHD$sysBP
sysbp.ttest <- t.test(sysbp.without.TYCHD,sysbp.with.TYCHD,alternative="two.sided",mu=0,var
.equal=FALSE,conf.level=0.99)
sysbp.ttest

##
##  Welch Two Sample t-test
##
## data:  sysbp.without.TYCHD and sysbp.with.TYCHD
## t = 11.417, df = 675.15, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  10.60066 16.80016
## sample estimates:
## mean of x mean of y
##  143.9811  130.2807
```

```r
cigsperday.with.TYCHD <- data.with.TYCHD$cigsPerDay
cigsperday.without.TYCHD <- data.without.TYCHD$cigsPerDay
cigsperday.ttest <- t.test(cigsperday.without.TYCHD,cigsperday.with.TYCHD,alternative="two.sided",mu=0,var.equal=FALSE,conf.level=0.99)
cigsperday.ttest

##
##  Welch Two Sample t-test
##
## data:  cigsperday.without.TYCHD and cigsperday.with.TYCHD
## t = 2.9522, df = 730.06, p-value = 0.003256
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  0.216586 3.242811
## sample estimates:
## mean of x mean of y
## 10.488330  8.758632

totchol.with.TYCHD <- data.with.TYCHD$totChol
totchol.without.TYCHD <- data.without.TYCHD$totChol
totchol.ttest <- t.test(totchol.without.TYCHD,totchol.with.TYCHD,alternative="two.sided",mu=0,var.equal=FALSE,conf.level=0.99)
totchol.ttest

##
##  Welch Two Sample t-test
##
## data:  totchol.without.TYCHD and totchol.with.TYCHD
## t = 5.1066, df = 723.43, p-value = 4.199e-07
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##   5.525944 16.834771
## sample estimates:
## mean of x mean of y
##  246.3501  235.1697

diabp.with.TYCHD <- data.with.TYCHD$diaBP
diabp.without.TYCHD <- data.without.TYCHD$diaBP
diabp.ttest <- t.test(diabp.without.TYCHD,diabp.with.TYCHD,alternative="two.sided",mu=0,var.equal=FALSE,conf.level=0.99)
diabp.ttest

##
##  Welch Two Sample t-test
##
## data:  diabp.without.TYCHD and diabp.with.TYCHD
## t = 7.7892, df = 684.89, p-value = 2.503e-14
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  3.347974 6.670166
## sample estimates:
## mean of x mean of y
##  87.15799  82.14892

bmi.with.TYCHD <- data.with.TYCHD$BMI
bmi.without.TYCHD <- data.without.TYCHD$BMI
bmi.ttest <- t.test(bmi.without.TYCHD,bmi.with.TYCHD,alternative="two.sided",mu=0,var.equal=FALSE,conf.level=0.99)
bmi.ttest

##
##  Welch Two Sample t-test
##
## data:  bmi.without.TYCHD and bmi.with.TYCHD
## t = 4.5453, df = 718.79, p-value = 6.434e-06
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 99 percent confidence interval:
##  0.4002117 1.4535148
## sample estimates:
## mean of x mean of y
##  26.56984  25.64298

hr.with.TYCHD <- data.with.TYCHD$heartRate
hr.without.TYCHD <- data.without.TYCHD$heartRate
hr.ttest <- t.test(hr.without.TYCHD,hr.with.TYCHD,alternative="two.sided",mu=0,var.equal=FA
LSE,conf.level=0.99)
hr.ttest

##
##  Welch Two Sample t-test
##
## data:  hr.without.TYCHD and hr.with.TYCHD
## t = 1.2275, df = 762.44, p-value = 0.22
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -0.7552585  2.1237813
## sample estimates:
## mean of x mean of y
##  76.31059  75.62633

gl.with.TYCHD <- data.with.TYCHD$glucose
gl.without.TYCHD <- data.without.TYCHD$glucose
gl.ttest <- t.test(gl.without.TYCHD,gl.with.TYCHD,alternative="two.sided",mu=0,var.equal=FA
LSE,conf.level=0.99)
gl.ttest

##
##  Welch Two Sample t-test
##
## data:  gl.without.TYCHD and gl.with.TYCHD
## t = 4.6041, df = 600.66, p-value = 5.059e-06
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##   3.559285 12.665306
## sample estimates:
## mean of x mean of y
##   88.7325   80.6202
```

Interaction Terms

```
interaction.data <- dplyr::select(framingham,male,age,cigsPerDay,prevalentStroke,prevalentH
yp,totChol,sysBP,BMI,heartRate,glucose)
ggpairs(interaction.data)
```