**ISyE 6740 – Spring 2021**

**Final Project Report**



**Team Member Names:**

Chris Dedow GTID# 903566165 Siham Elmali GTID# 903643694 Alex Haan GTID# 903548955

**Project Title**: Predicting 10-Year Risk of Coronary Heart Disease

[**Problem Statement**](#_heading=h.wv5ml2ie6arm)2

[**Methodology**](#_heading=h.id4iuifoxkjt)2

[**Data Collection and Preprocessing**](#_heading=h.h6x8iqji4fnm)3

[Data Collection2,3,4](#_heading=h.yqyxeq3iu8a0) 3

[Data Preprocessing](#_heading=h.dkmoe9ni4cny) 4

[Exploratory Data Analysis](#_heading=h.16eccrc9vh74) 4

[Categorical Data Handling](#_heading=h.eaebi1t5kf5d) 6

[Train/Validation/Test Data Split](#_heading=h.qxzj06jq1pxl) 7

[Imputing Missing Values](#_heading=h.4bz8t5wivcxz) 7

[Outlier Detection and Removal](#_heading=h.fwpvgtyc29g3) 7

[Data Standardization](#_heading=h.hrcdgpfz9meo) 9

[Data Balancing](#_heading=h.bql97merekwv) 9

[Feature Selection](#_heading=h.i57hbip6dgf3) 10

[**Modeling**](#_heading=h.hh5vp74splcs)11

[Modeling Plan](#_heading=h.yhqe8jyycir1) 11

[Logistic Regression](#_heading=h.tlygmtmf9r26) 12

[Support Vector Machines](#_heading=h.m45w0hrpyo8s) 13

[Random Forest](#_heading=h.m45w0hrpyo8s) 13

[Extreme Gradient Boosting (XGBoost)](#_heading=h.m45w0hrpyo8s) 14

[**Evaluation**](#_heading=h.l97lgl6giv2d)14

[**Results**](#_heading=h.wco2vbgjm88h)16

[**Collaboration**](#_heading=h.s93c0oijufeo)17

[**References**](#_heading=h.1573zoq2tw85)18

[**Appendix**](#_heading=h.2loqr533oxh8)19

[Feature selection](#_heading=h.4wceaactag9o) 19

[Sequential Forward Selection](#_heading=h.ns1qqg876748) 19

[Boruta Feature selection](#_heading=h.wqm4rg2ipu1m) 20

# Problem Statement

Can we predict coronary heart disease and prescribe actions to reduce an individual’s 10-year risk factor?

Over 17.5 million people die each year due to cardiovascular diseases. This accounts for roughly 30% of all deaths globally and is expected to continue increasing. Approximately 8 million people out of the 17.5 million deaths are due to coronary heart disease1.

A reduction in coronary heart disease related deaths can come from a better understanding of the underlying factors that cause elevated risk levels. This project analyzes data taken from one of the largest running medical studies ever conducted, The Framingham Heart Study. The aim of this project is to highlight that you can predict individuals who have elevated 10-year risk levels of coronary heart disease so that physicians can prescribe solutions to help reduce this risk.

# Methodology

To complete this project, our team has chosen to take a structured approach by leveraging machine learning methodology that includes the following steps: Data Collection and Preprocessing, Modeling, and Results and Evaluation.

The data preprocessing phase took the longest as our dataset had several features with missing values, outliers, and had an unbalanced response variable. We handled missing values through imputation techniques, identified outliers and test model variations including and excluding them to see how they affected our end results. We also balanced the dataset using an oversampling technique to allow our models to learn from an equal distribution of binary outcomes. This will minimize the misclassification rate between our binary outcomes.

After handling missing values, outliers, and balancing the data, feature selection was used to minimize noise in our dataset that did not materially impact the predictability of our model. Several feature selection methods were used to pick the right predictors for our analysis and to simplify our models.

In the modeling phase, we used the selected features to build four models using various sci-kit learn packages. The models we decided to work with were chosen because they are both performant and versatile when it comes to supervised learning problems. All of the models had gone through cross validation using non-exhaustive cross validation methods (k-fold cross validation, repeated stratified cross validation, etc.). The parameters in each model had undergone hyperparameter tuning using cross validated parameter grid search to ensure that we are fitting the best models into our data.

# Data Collection and Preprocessing

## Data Collection2,3,4

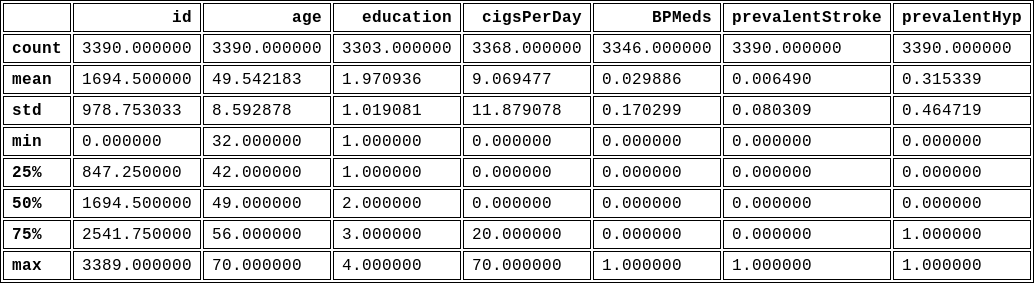
The data source used in this project is based on the famous Framingham Heart Study which began in 1948 with 5209 adult subjects from Framingham, Massachusetts and has continued to this day over 72 years spanning over three generations. It was created to help identify the common factors or characteristics that contribute to cardiovascular disease. A description of the features in the dataset can be found below:

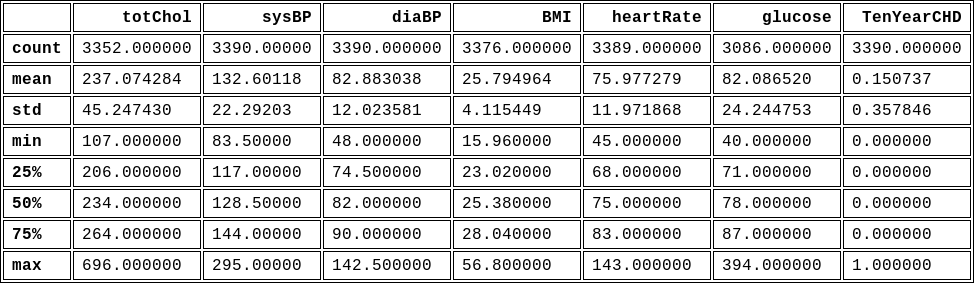
|  |  |
| --- | --- |
| **Feature** | **Description** |
| age | Age of the patient (Continuous; Although the recorded ages have been truncated to whole numbers, the concept of age is continuous) |
| education | Rating ranging from 1 to 4 |
| sex | Male or female ("M" or "F") |
| is\_smoking | Whether or not the patient is a current smoker ("YES" or "NO") |
| cigsPerDay | The number of cigarettes that the person smoked on average in one day. (can be considered continuous as one can have any number of cigarettes, even half a cigarette.) |
| BPMeds | Whether or not the patient was on blood pressure medication (Nominal) |
| prevalentStroke | Whether or not the patient had previously had a stroke (Nominal) |
| prevalentHyp | Whether or not the patient was hypertensive (Nominal) |
| diabetes | Whether or not the patient had diabetes (Nominal) |
| totChol | Total cholesterol level (Continuous) |
| sysBP | Systolic blood pressure (Continuous) |
| diaBP | Diastolic blood pressure (Continuous) |
| BMI | Body Mass Index (Continuous) |
| heartRate | Heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of a large number of possible values.) |
| glucose | Glucose level (Continuous) |
| TenYearCHD | 10 year risk of coronary heart disease CHD (binary: “1”, means “Yes”, “0” means “No”) |

*Table 1: Description of the variables in the dataset*

## Data Preprocessing

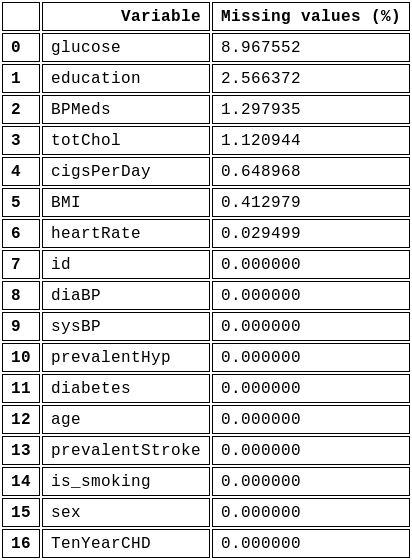
### Exploratory Data Analysis





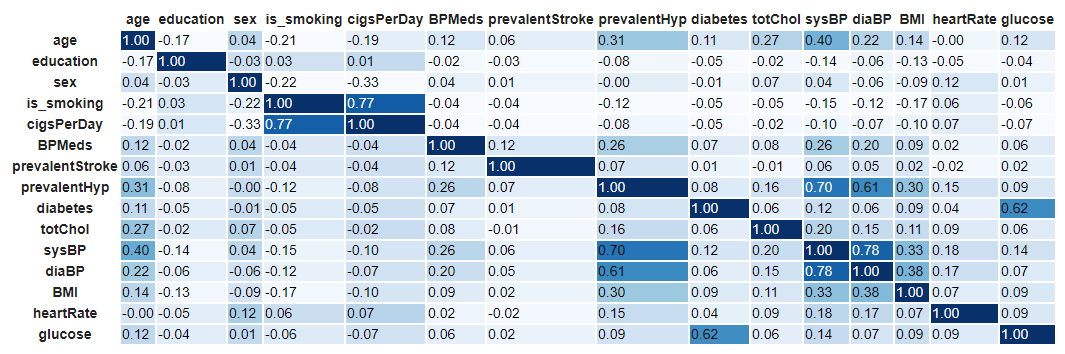
*Table 2: Summary statistics for the dataset*

7 out of 16 or 43.75% of features have missing values in the table below:



*Table 3: missing values count*

Correlation matrix of the features:



*Figure 1: Correlation matrix*

We can restrict the correlation coefficient to greater or equal to +/- 0.5 to isolate the highly correlated variables in the dataset.

### 

*Figure 2: The highly correlated features in the dataset*

Correlation coefficients that are higher than an absolute value of 0.5 are shown above. Later in this paper we will discuss in more detail how Elastic Net helps handle these correlated features.

### Categorical Data Handling

Most of the features in our dataset are continuous or nominal (0s and 1s). The only two variables that we had to convert to binary values were `*sex*` and *`is\_smoking`*.

We dropped the following from our dataset:

1. id - not relevant as a predictor variable
2. is\_smoking - high correlation with cigsPerDay
3. TenYearCHD - response variable

### Train/Validation/Test Data Split

We split our dataset 60/20/20 where 60% is designed for training, 20% for validation and the remaining 20% is dedicated to testing.

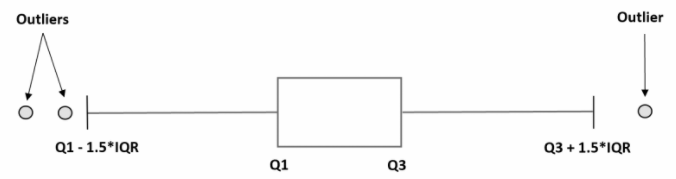
### Imputing Missing Values

The data source had 7 out of 16 or 43.75% of features with missing values. The feature with the highest percentage of missing values of 9% was *`glucose`*. We handled missing values with an imputation method that uses a K-nearest neighbors algorithm. Each missing value is imputed using the mean value from the five nearest neighbors found in the training set.

### Outlier Detection and Removal

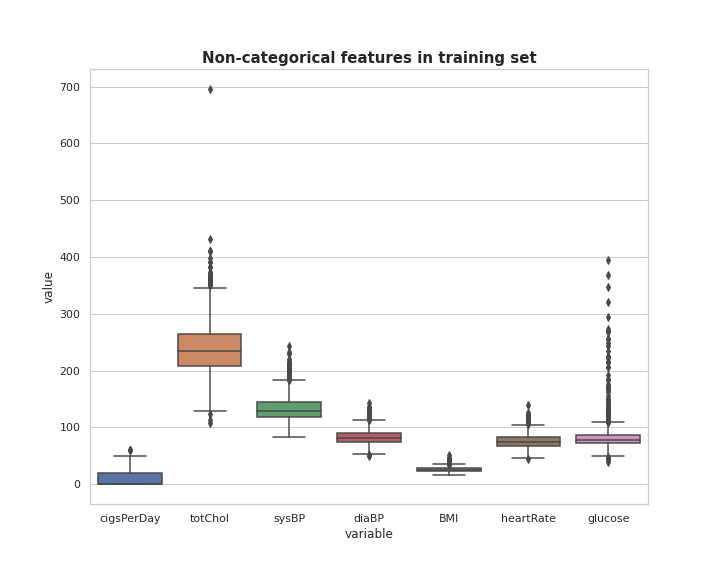
An outlier is a data point that lies far away from the rest of the data points in a dataset. Outlier detection and removal (optionally) is an integral part of the preprocessing stage of any machine learning pipeline. In most cases, outliers can be removed but sometimes we ought to investigate why such data points exist in our dataset. As the saying goes *"I am not an outlier; I just haven't found my distribution yet”5.* It is vital to investigate outliers and understand them before blindly removing them. For our modeling purposes, we decided to first investigate some of the outliers to make sure it makes sense to remove them then build out models with and without outliers. Comparing models built on the dataset with and without outliers will give us a much better understanding on how outliers affect the performance of the models.

To detect these outliers, we followed a statistical approach. One commonly used and the most trusted approach in the research field is the interquartile range or **IQR.** The IQR statistical approach is the difference between the 25th percentile (**Q1**) and the 75th percentile (**Q3**) in a dataset. It measures the spread of the middle 50% of values. An observation is declared to be an outlier if it has a value 1.5 times greater than the IQR or 1.5 times less than the IQR.

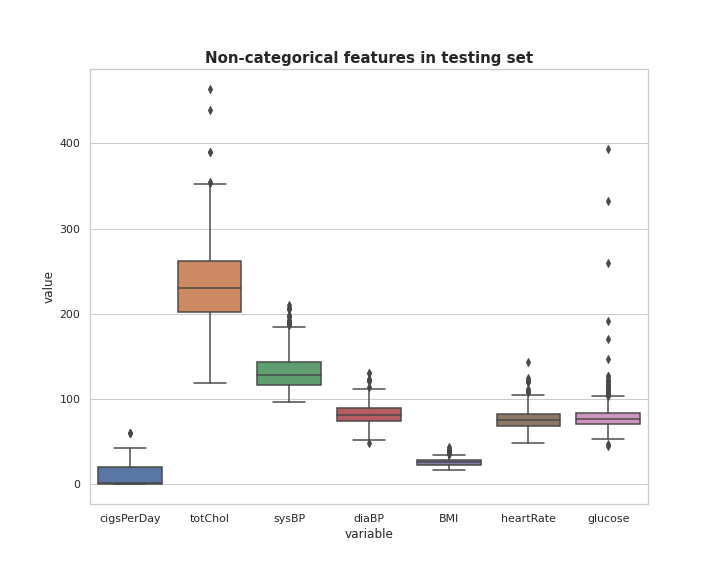
****

*Figure 3: IQR approach*

The outliers were detected in both training and test sets separately and visually identified using boxplots.



*Figure 4: Boxplot of the non-categorical features in the training set*

**

*Figure 5: Boxplot of the non-categorical features in the testing set*

Our approach was to use the IQR statistical method to detect and remove outliers on each of the seven non-categorical features separately. We then joined the features with the remaining categorical features to rebuild our training and testing dataset. This approach introduced missing values in our datasets which were solved for by using a k-nearest neighbors imputation method from scikit-learn. We chose this novel approach to handle outliers because we didn't want to throw away important information from the features that did not have outliers in a particular row. We leveraged KNN imputation with five nearest neighbors which essentially imputed missing values with the mean of the five nearest neighbors. This is a clever imputation method to do because you avoid somewhat skewing the mean of the dataset.

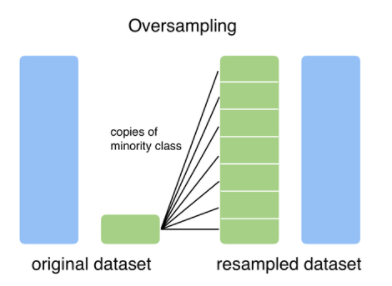
### Data Standardization

In this step we scale our training datasets with and without outliers. Standardization is used to remove bias in our model fitting and model learned function that would otherwise be present since each feature consists of a different scale. This is achieved by using feature-wise standardization with the **StandardScaler()** function. It normalizes each feature individually setting its mean = 0 and its standard deviation = 1. We scale the independent features in the training set and use this fitted model to transform both our validation and test sets so we are able to later test for performance of our model in a standardized approach.

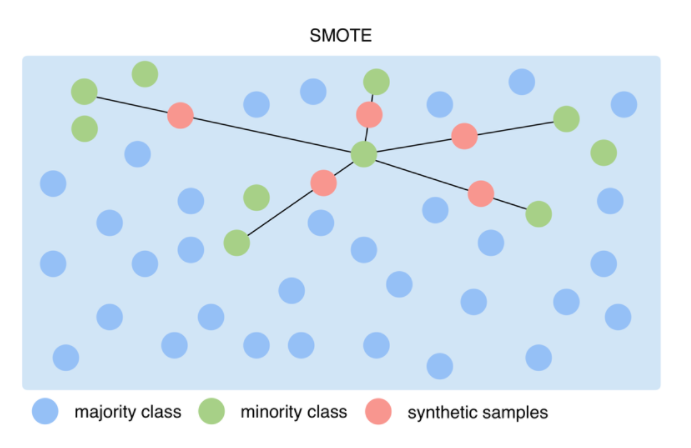
### Data Balancing

During the exploratory data analysis phase, we discovered that our target class was highly imbalanced. Balancing the dataset was pivotal because we didn't want our models to be more biased to the dominant target class.

To balance the dataset, we used a statistical method called **Synthetic Minority Oversampling Technique** **(SMOTE)**.

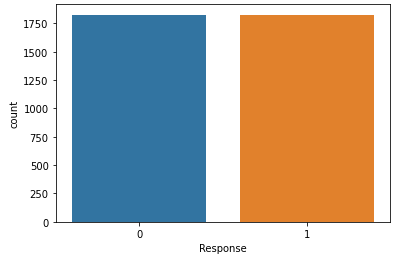
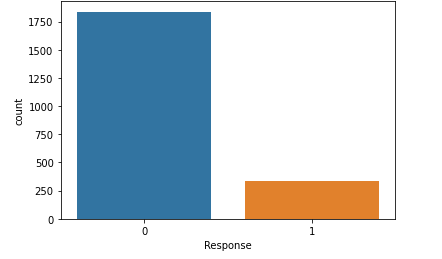


*Figure 6: Oversampling illustration6*



*Figure 7: SMOTE Illustration6*

The SMOTE algorithm works by generating new data from existing minority cases. The method does not change the number of majority cases. The newly generated synthetic samples are not just copies of the existing minority samples. Instead, The SMOTE algorithm takes sample features for each target class and its nearest neighbors, then combines them together to create new samples. This method increases the features for each class, thus creating a more balanced dataset7.



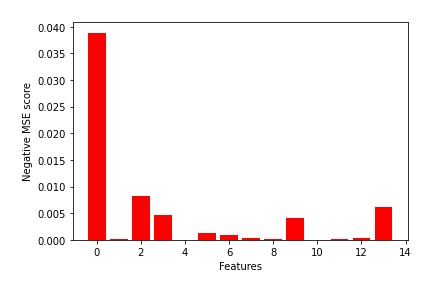
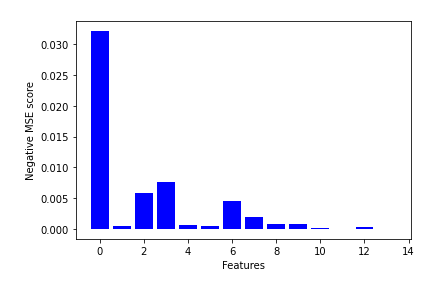
*Figure 8: Response variable before oversampling Figure 9: Response variable after oversampling*

### Feature Selection

Feature selection is an automatic method to select only the attributes in the dataset that are most relevant to the predictive modeling of the response variable (TenYearCHD). To select the most important features from our data, we relied on a commonly used approach, Elastic Net. Elastic Net was chosen because it allowed us to have the desired variable selection benefits of a Lasso Regression model while retaining the predictive benefits of a Ridge Regression model.

We selected a 10-fold cross validated Elastic Net model which we then fed into a Permutation Importance model as an estimator. This permutation method was used to evaluate the importance of each feature using negative mean squared error or MSE as a baseline scoring metric. Each feature column from the dataset is permuted and the metric is evaluated again. At the end of this process, our final permutation importance for each feature is defined as the difference between the baseline metric and the metric evaluated from permuting the said feature8.

We applied this feature selection method to the data with and without outliers.



*Figure 10: Feature importance on no-outlier dataset*  *Figure 11: Feature importance on outlier dataset*

# Modeling

## Modeling Plan

In order to determine which data preparation steps would lead to the greatest model performance, we chose to try four variations of our dataset for training our Logistic Regression, Support Vector Machine (SVM), Random Forest, and Extreme Gradient Boosting (XGBoost) models using the following dataset variants:

* Outliers excluded, all features included
* Outliers excluded, subset of features included
* Outliers included, all feature included
* Outliers included, subset of features included

Feature Index:

subset\_outlier\_feature\_indx = [0,2,3,5,6,9,13]

subset\_no\_outlier\_feature\_indx = [0,2,3,4,6,7,8,9]

1. age
2. education
3. sex
4. cigsPerDay
5. BPMeds
6. prevalentStroke
7. prevalentHyp
8. diabetes
9. totChol
10. sysBP
11. diaBP
12. BMI
13. heartRate
14. glucose

**Hypothesis:** The removal of outliers and less impactful features would lead to greater model performance due to lower risk of overfitting.

The four models below were all trained on the four different dataset variants and compared against the validation set. We decided to optimize our models for Recall as our performance metric of choice and to use it for model comparisons instead of overall accuracy, precision and F1 score which are other common model metrics used for evaluating performance. We chose Recall because it measures how often our model correctly identifies someone has a 10-year coronary heart risk.

Recall = TP / (TP + FN)

The following four models all use RepeatedStratifiedKFold as our cross validation function of choice with five splits repeated 3 times with a specified random state.

## Logistic Regression

The first model we chose for binary classification is Logistic Regression. Unlike many other classification algorithms, Logistic Regression leverages probability to determine a likelihood ranging from 0 to 1 that the response variable meets a certain class. This feature that separates Logistic Regression from Linear Regression is the complex Sigmoid cost function by mapping any real value from 0 to 1.

To ensure that Logistic Regression gave us the greatest possible performance, we spent time tuning the model’s hyperparameters by using Scikit-learn’s GridSearchCV function. The parameters included in hyperparameter tuning were the following:

* *Inverse Regularization Parameter (C)*: A penalty term that retains strength modification of Regularization by being inversely positioned to the Lambda regulator. C = 1 / lambda where lambda controls the trade off between allowing the model to increase its complexity. If lambda is low or closer to 0 it will be more likely to overfit the data while if it is too high it will underfit the data. We chose to try values with range = np.linspace(0.001, 2, 100)
* *Penalty:* An input parameter that specifies the regularizer that is applied to the coefficients in the model. We tried the “l1” and “l2” penalties which equal the absolute value of the coefficients or the square of the coefficients, respectively.

## Support Vector Machines

The second model we chose was the Support Vector Machine (SVM) which is a linear model that predicts the binary response variable based on a decision boundary created by the algorithm. The algorithm functions to separate the two binary response variables by generating a hyperplane that maximizes the margin between the two classes.

For hyperparameter tuning, we chose to optimize based on the following variables:

* *Regularization Parameter (C)*: Otherwise known as “soft classifier”, the Regularization Parameter is the penalty for misclassification. We tried the following values: 0.1, 1, 10, and 100
* *Gamma:*
  + ‘scale’ ((default) is passed then it uses 1 / (n\_features \* X.var()) as value of gamma)
  + ‘auto’ (uses 1 / n\_features)
* *Kernel:*
  + ‘rbf’ (computes the similarity of how close two points are to each other)
  + ‘sigmoid’ (equivalent to a two-layer perceptron neural network)

## Random Forest

The third model we chose to use was Random Forest which is an ensemble model that leverages multiple decision trees and bagging to come to a conclusive prediction for classification. By combining subsets of the data to train various decision trees, Random Forest can overcome bias and variation in the data to reduce overfitting and return high predictive performance.

To optimize the model, we tested variations of the following parameter:

* *n\_estimators:* The number of decision trees that the model creates before taking the maximum prediction votes or taking the averages of predictions. We tried the number of trees ranging from 1 to 200.

## Extreme Gradient Boosting (XGBoost)

The fourth and final model we chose to leverage was Extreme Gradient Boosting (XGBoost). XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework with the focus of high speed and performance. Similar to bagging-based ensemble algorithms, XGBoost builds improved decision trees sequentially with information obtained from the previous iteration.

For hyperparameter tuning, we chose to optimize based on the following variables:

* *min\_child\_weight*: The minimum sum of instance weight needed in a leaf node. We tried the following values: 1,5, and 10
* *gamma*: It represents the minimum loss reduction required to make a further partition on a leaf node of the tree. We tried the following values: 0.5, 1, 1.5, 2, and 5
* *subsample*: represents the ratio of the training instances. We tried the following values: 0.6, 0.8, and 1.0
* *colsample\_bytree*: represents the subsample ratio of columns when constructing each tree. We tried the following values: 0.6, 0.8, and 1.0
* *max\_depth*: represents the maximum depth of a tree. We tried the following values: 3, 4, and 5

# Evaluation

To compare the performance of the four various models, we trained them on the four dataset variations then tested each model on the validation set we set aside earlier when splitting the data into train, validation, and test. As a reminder, each model was hyperparameter tuned in hopes of optimizing Recall for the “Yes” classification. However, additional metrics such as Accuracy and F1 Score were still considered to ensure the model was still performing well overall.

Below is the performance of each model on the specified training set for predicting the “Yes” that the patient is at risk for coronary heart disease.

* Models Trained on: Outliers Included, All Features Included

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Recall | Accuracy | F1 Score |
| Random Forest | 0.2680 | 0.8171 | 0.2955 |
| Support vector machines | 0.1753 | 0.7198 | 0.1518 |
| XGBoost | 0.2474 | 0.7979 | 0.2595 |
| Logistic regression | 0.6495 | 0.6888 | 0.3739 |

* Models trained on: Outliers Included, Subset of Features Included

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Recall | Accuracy | F1 score |
| Random Forest | 0.3402 | 0.7714 | 0.2086 |
| Support vector machines | 0.4536 | 0.6755 | 0.2857 |
| XGBoost | 0.3608 | 0.8083 | 0.3500 |
| Logistic regression | 0.6598 | 0.7006 | 0.3867 |

* Models trained on: Outliers Excluded, All Features Included

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Recall | Accuracy | F1 score |
| Random Forest | 0.2371 | 0.8201 | 0.2738 |
| Support vector machines | 0.1546 | 0.7242 | 0.1382 |
| XGBoost | 0.2680 | 0.8127 | 0.2905 |
| Logistic regression | 0.6598 | 0.6740 | 0.3668 |

* Models trained on: Outliers Excluded, Subset of Features Included

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Recall | Accuracy | F1 score |
| Random Forest | 0.3299 | 0.7714 | 0.2922 |
| Support vector machines | 0.3918 | 0.6652 | 0.2508 |
| XGBoost | 0.2990 | 0.7802 | 0.2802 |
| Logistic regression | 0.6804 | 0.6873 | 0.3837 |

Based on the above results, the model that performed the best on the validation set was the Logistic Regression model that was trained on the subset of features with outliers excluded. The results for the model were the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Recall | Accuracy | F1 Score |
| Logistic regression | 0.6804 | 0.6873 | 0.3837 |

# Results

When evaluating the results of all models and data variants, it was clear that the Logistic Regression model outperformed others with the highest Recall regardless of which data variant was used for training. It is important to note that Logistic Regression had the lowest or second lowest Accuracy for all data variants, but as it is our priority to ensure all patients with coronary heart disease risk are not missed, the superior Recall still outweighs the lower Accuracy.

With Logistic Regression now determined as the optimal model, we then confirmed our results by testing the four data variations of Logistic Regression against the test set that we held aside originally. Below are the results of Logistic Regression based on different training variants:

|  |  |  |  |
| --- | --- | --- | --- |
| Training Variant | Recall | Accuracy | F1 score |
| Outliers: Full Features | 0.6585 | 0.6796 | 0.3830 |
| Outliers: Subset Features | 0.6585 | 0.6722 | 0.3776 |
| No Outliers: Full Features | 0.6585 | 0.6703 | 0.3763 |
| No Outliers: Subset Features | 0.6585 | 0.6685 | 0.3763 |

As shown in the above table, the Recall score on the test set performs quite well with the same score of 0.6585. Accuracy and F1 Score fluctuate with a variation of +/- 0.01. An interesting finding is that the Recall and F1 Score were identical for variants that excluded outliers. Additionally, it appears that the models that included outliers had higher Accuracy and F1 Score than those without outliers, but the minimal gain is trivial based on the increased chance of overfitting. These results, however, go against our results on the validation set, where the Recall was highest when only a subset of features were utilized.

Based on the results above if we were to choose a model for production it would be the one that excludes outliers and uses a subset of features. This would allow for greater computational efficiencies as well as a lower likelihood of overfitting.

# 

# Collaboration

|  |  |  |
| --- | --- | --- |
| **Section** | **Owner** | **Description** |
| Proposal | All | Find dataset, prepare proposal  with methodology and  problem description |
| EDA | Siham | Data inspection, checking for missing values, correlation matrix, handling categorical features |
| Data split and standardization | Alex | Splitting and standardizing the data |
| Imputing missing values | Siham | Imputing missing values |
| Outlier Detection | Siham | Implement IQR method to find outliers in non-categorical features and provide the outlier indices that need to be removed |
| Outlier Removal | Chris | Remove outliers from the non-categorical features and impute using knn |
| Data Balancing | Siham | Balance the dataset using SMOTE |
| Feature Selection | All | Alex, Chris: Elastic net  Siham: Permutation Importance using Elastic net, Sequential forward selection and Boruta feature selection (mentioned in appendix) |
| Modeling | Siham and Alex | Model Implementation |
| Evaluation and Results | All | Evaluating model performance and discussing the results |
| Write-up | All | All team members contributed to this piece equally |

# 

# References

The Framingham Heart Study began in 1948 with 5209 adult subjects from Framingham, Massachusetts and has continued to this day over 72 years spanning over three generations. It was created to help identify the common factors or characteristics that contribute to cardiovascular disease.

1. US National Library of Medicine National Institutes of Health: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4966216/#ref3>
2. Dataset source: <https://www.kaggle.com/christofel04/cardiovascular-study-dataset-predict-heart-disea?select=train.csv>
3. About the Framingham Heart Study: <https://framinghamheartstudy.org/fhs-about/>
4. Framingham Coding Manual: <https://www.ncbi.nlm.nih.gov/projects/gap/cgi-bin/GetPdf.cgi?id=phd000456.5>
5. URL: <https://www.maths.usyd.edu.au/u/jchan/Statistic_joke.pdf>
6. URL: <https://blog.strands.com/unbalanced-datasets>
7. URL: <https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/smote>
8. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html>

**Technical Details:**

We used Python and Google collab, a cloud based Jupyter Notebook environment, to complete this project.

# 

# 

# 

# 

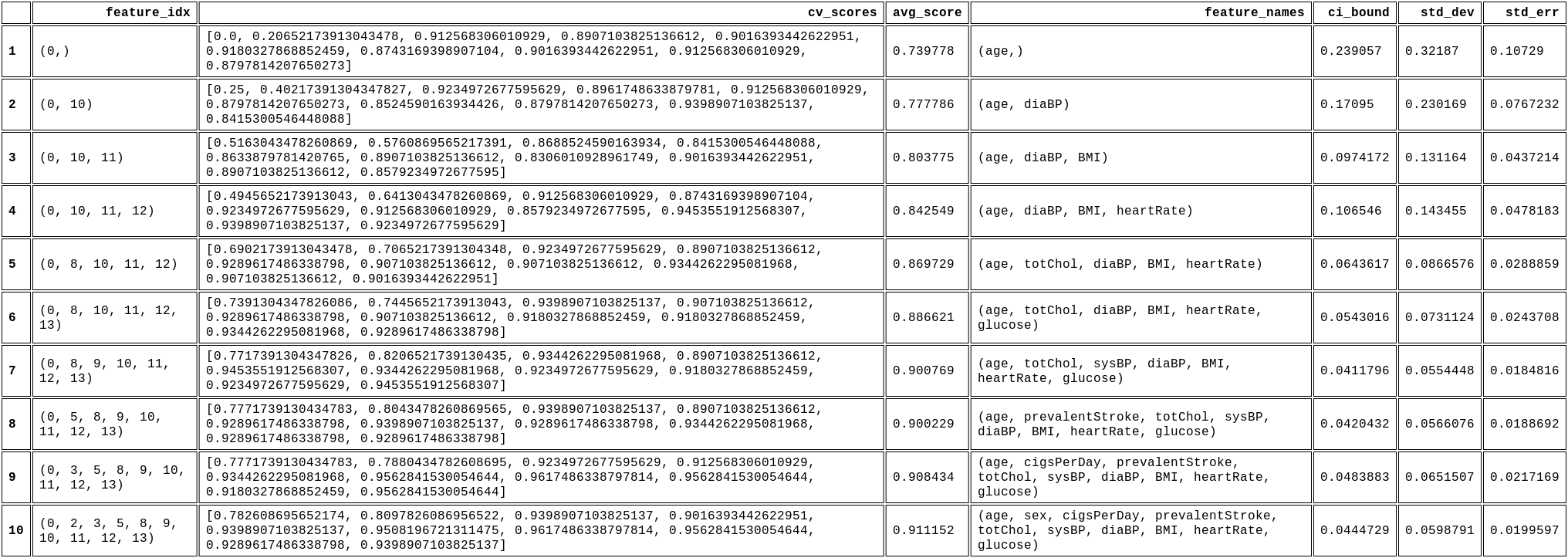
# Appendix

## Feature selection

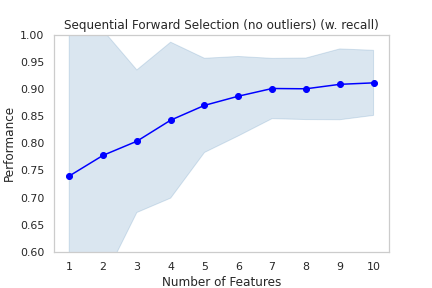
To select the most relevant features for our modeling, our team explored several variable selection methods. Sequential forward selection and Boruta feature selection are widely used and perform best when used with a tree based estimator.

### Sequential Forward Selection

**Sequential Forward Selection on Dataset Without Outliers:**

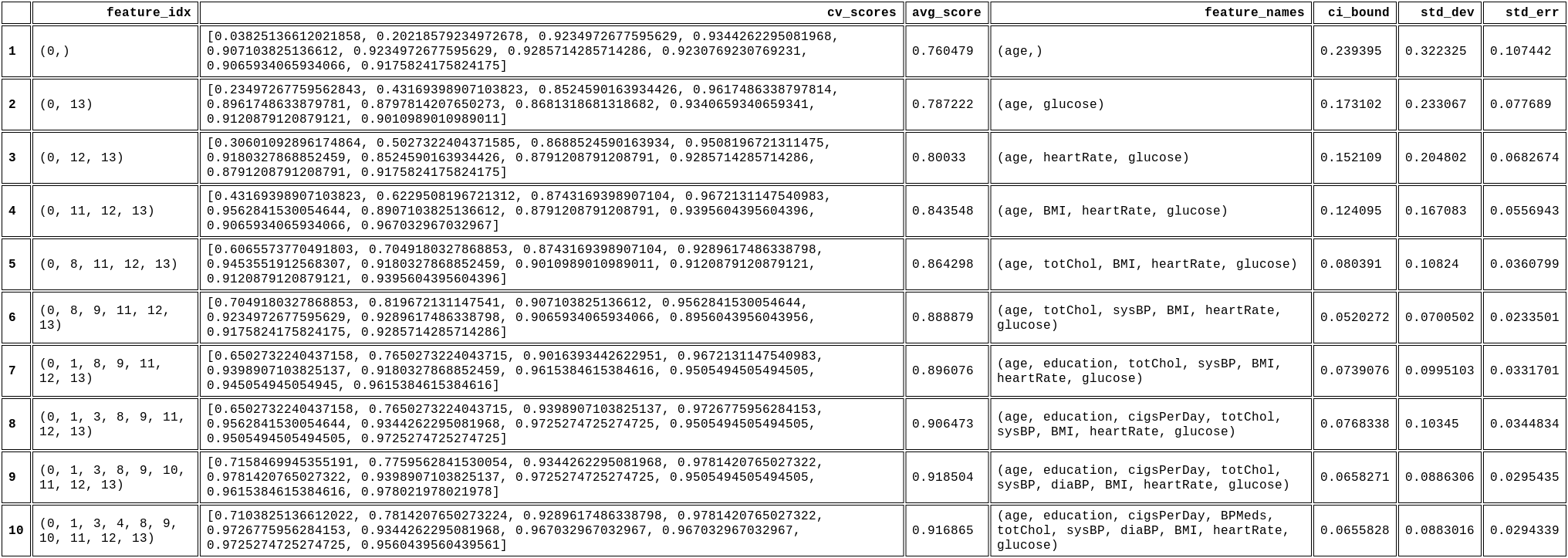


*Figure 12: Sequential Forward Feature selection on dataset without outliers*

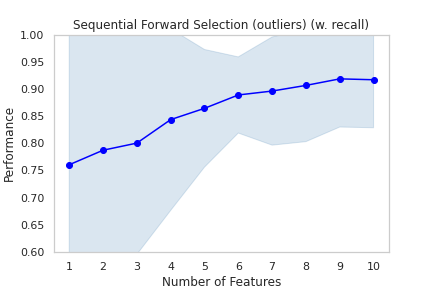


*Figure 13: Model performance (i.e Recall) vs the # of features*

**Sequential Forward Selection on Dataset With Outliers**



*Figure 14: Sequential Forward Feature selection on dataset with outliers*

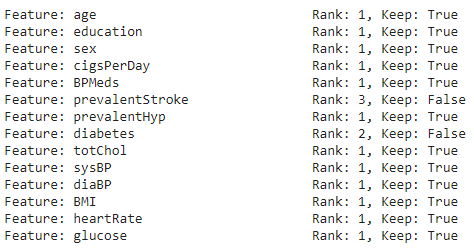


*Figure 15: Model performance (i.e Recall) vs the # of features*

### Boruta Feature selection

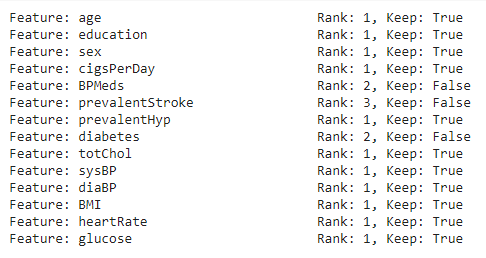
Boruta is a variable selection method that tries to find all relevant features that carry information usable for prediction. This method works best when used on a tree based estimator (randomForest in our case).

**Boruta feature selection on dataset without outliers**



*Figure 16: Features to keep or drop using Boruta feature selection on no-outlier dataset*

**Boruta feature selection on dataset with outliers**



*Figure 17: Features to keep or drop using Boruta feature selection on outlier dataset*