Data analytics II: individual assignment

MSIN0025 Data Analytics
Student ID: 18079757
Word count: 1991

I- The Problem

In order to solve the healthcare financial crisis in the US, reforms have been applied towards a P4P (pay for performance) system[1]. This means that people will not pay for services but proportionally to "how good" the outcome of their care was relatively to their starting situation. However, this puts pressure on hospitals that have to take on more financial risk, especially non-profit hospitals as half of their funding come from fee-for-service revenue[2].

As this is a very complex problem, I decided to start by providing a tool to optimizing the most important reason for hospital admissions. According to the American heart association, the most common reason for hospitalization and for death in the US was heart diseases, with a number of deaths due to heart disease knowing a 41% increase since 1990[3]. Therefore, my analysis will try to determine the key determinants of heart disease, and how effectively can people prone to heart disease be identified.
Answering these questions will allow hospitals to intervene much earlier and head off hospitalization, saving billions of dollars. In fact, the AHRQ estimated admissions that could have been prevented, linked to heart disease, accounted for an evitable cost of $9 billion. Moreover, having less unnecessary admissions could improve the care outcomes of necessary hospitalizations and therefore, hospitals' revenue.
The decision makers of my problem are the hospitals, especially the cardiologists for whom it is important to detect and predict heart disease complications in order to act quickly, which will prevent aggravations along with the number of hospitalizations.

The data used in my analysis comes from a Kaggle competition but comes from the Cleveland clinic in January 1988. The dataset is therefore quite old, but I strongly believe that this only affects the accuracy of the measures inside the dataset due to medicine progresses, but this does not change the impact of the problem. Moreover, one should highlight that only 14 out of the 76 original attributes were selected in Kaggle's version of the dataset.
The target attribute is the "target" variable that we renamed "heart_disease". If is dependant from the 13 other variables and is equal to 0 if the patient does not have a heart disease and 1 if he does.

[1] Mckinsey, 2013
[2] Berkeley hospital review, 2020
[3] Institute for Health Metrics and Evaluation, 2015

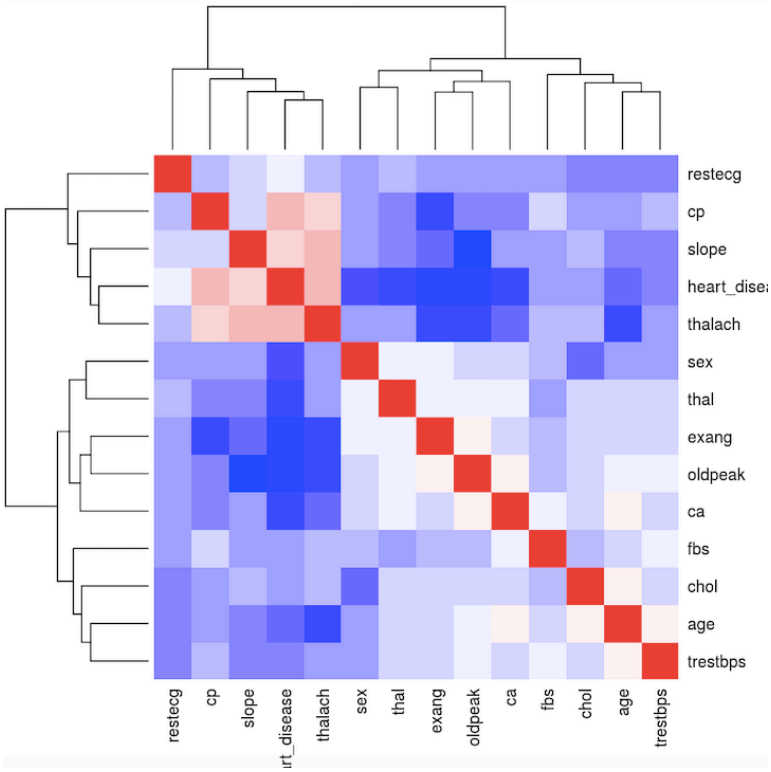
II- Understand the Data

After using the function dimension, we identified that our multivariate dataset had 303 rows and 14 columns (variables). Each row corresponds to a patient's characteristics. The dataset contains demographic attributes (sex and age) as well as the patient's health state. Half of the variables are numeric, and the other half are categorical and take binary values.
The table shown under gives a classification of variables with their meanings, values, and type[4].

Variable Name	Meaning	Value meaning	Type
age	Patient's age	29-77	Numeric
sex	Patient's gender	0 = female, 1 = male	Categorical
cp	Chest pain type	0 (no pain) 1, 2, 3	Numeric
trestbps	The person's resting blood pressure (mm Hg on admission to the hospital)	94-200	Numeric
chol	The person's cholesterol measurement in mg/dl	126-564	Numeric
fbs	The person's fasting blood sugar	if > 120 mg/dl, 1 = true; 0 = false	Categorical
restecg	Resting electrocardiographic measurement	0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria	Categorical
thalach	The person's maximum heart rate achieved	71-202	Numeric
exang	Exercise induced angina	1 = yes; 0 = no	Categorical
oldpeak	ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot)	0-6.20	Numeric
slope	the slope of the peak exercise ST segment	Value 1: upsloping, Value 2: flat, Value 3: down-sloping	Categorical
ca	The number of major vessels	0-3	Numeric
thal	A blood disorder called thalassemia	1 = normal; 2 = fixed defect; 3 = reversable defect	Categorical
target	The patient has Heart disease	0 = no, 1 = yes	Categorical

[4] UCI, 2020

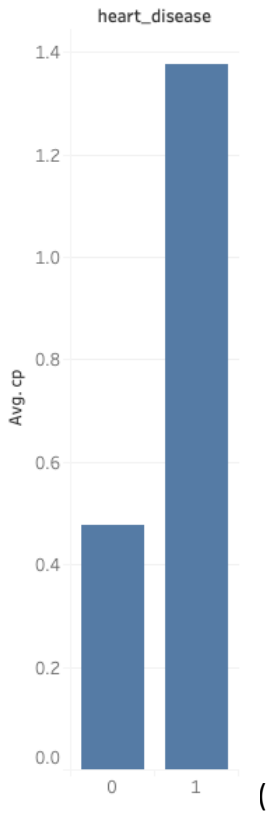
To have a first look at the key variables I computed a correlation matrix. The darker red the colour, the most positively correlated the values are and the same applies for negatively correlated values in blue. This gave us that the most positively correlated values to the target attribute were “cp”, “thalach”, “slope” and “restecg”.



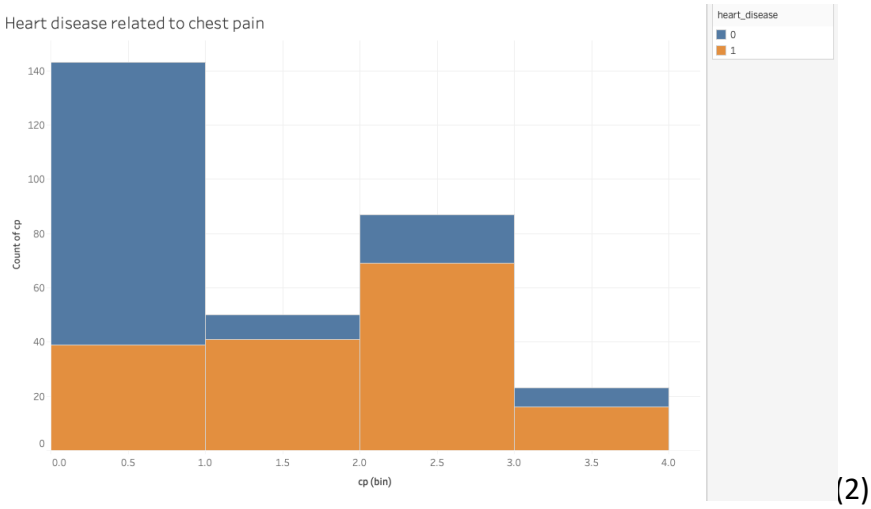
This is confirmed by the correlation table below which gives us how correlated these 4 values are to the target.

heart_disease	1
cp	0.433798261506894
thalach	0.421740933810675
slope	0.345877078241725
restecg	0.137229502873773

In order to understand these correlations, we made and analysed visualisations with tableau and R. **For tableau,** had to change the type of the categorical variables from measures (automatic setting of tableau) to dimensions beforehand.



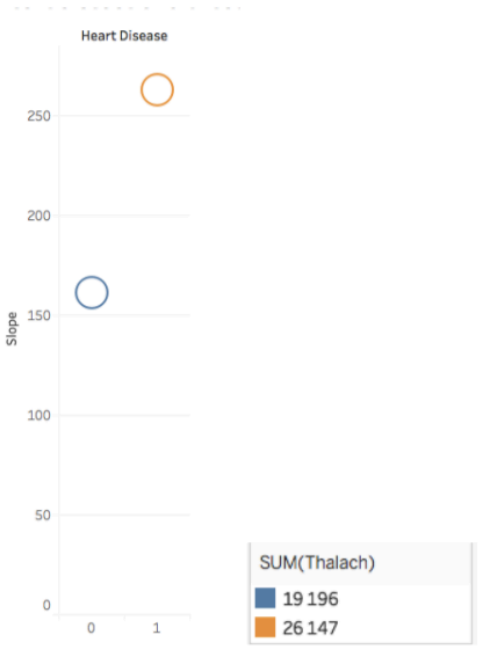
(1)



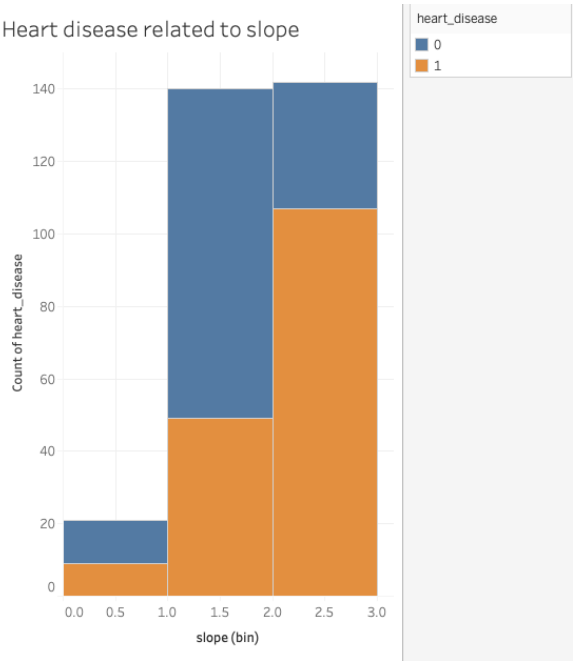
(2)

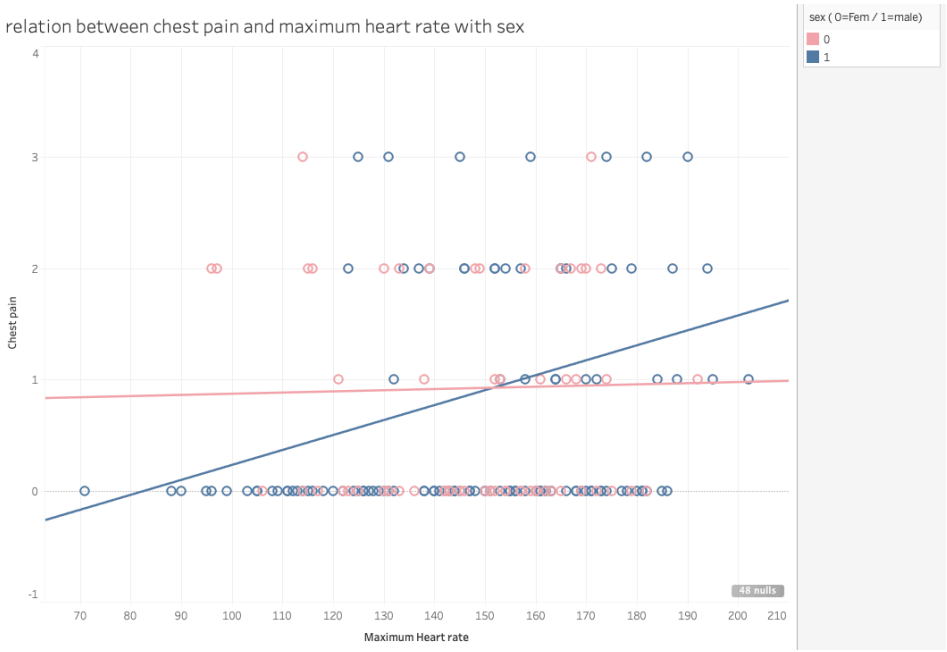
Then, we looked a little deeper into cp, which has the highest correlation to out target variable. The first barplot (1) effectively showed that people with heart disease have higher chest pain types. Then, we made a stacked barplot (2) showing the distribution of heart disease in chest pain type, by creating bins of size 1 in cp. This showed that a higher proportion of people with heart disease lied in the higher categories of chest pain. In fact, less than half of the population with no chest pain have a heart disease.

(3)



(4)





(5)

Then, we explored the second and third highest correlated variables thalach (maximum heart rate) and slope (the slope of the peak exercise ST segment). We first found that (3) the higher the slope value -so the more downward sloping the ST segment was- and the higher the thalach of patients, the more likely they were to also have a heart disease. Moreover, the fourth graph showed us that a downward slope is the most related slope to heart disease, which explains the third graph that has been impacted by the value 3, which is the highest, of the downward slope.
Finally, by linking the thalach and cp variables in a scatterplot coloured by sex, we found, thanks to the regression line, that thalach and cp were positively correlated, especially for men.

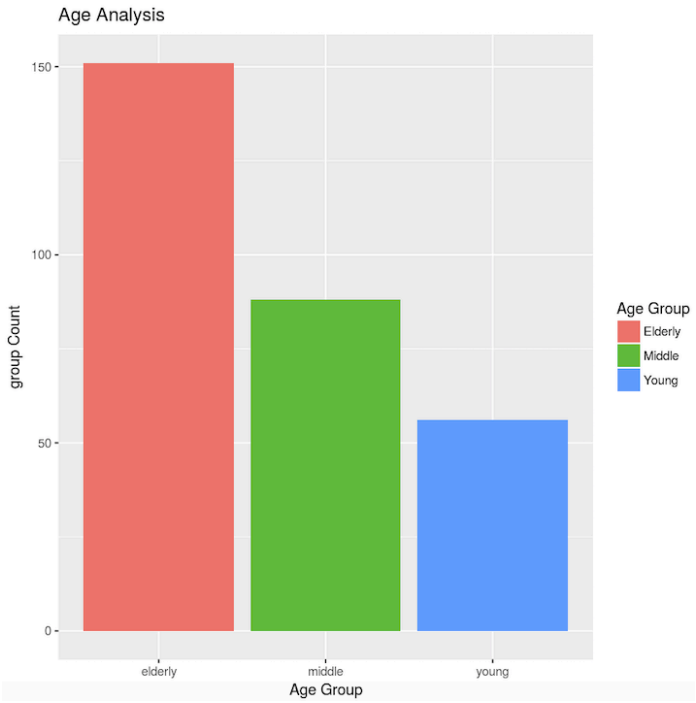
Then, we used R starting by building an aggregation table, which gave us an idea of what the mean sick and healthy patients would look like for the most correlated values. A mean sick patient therefore has higher chest pain, thalach and restecg, along with a downward sloping ST segment.

heart_disease	cp	thalach	slope	restecg
0	0.4782609	139.1014	1.166667	0.4492754
1	1.3757576	158.4667	1.593939	0.5939394

(6)

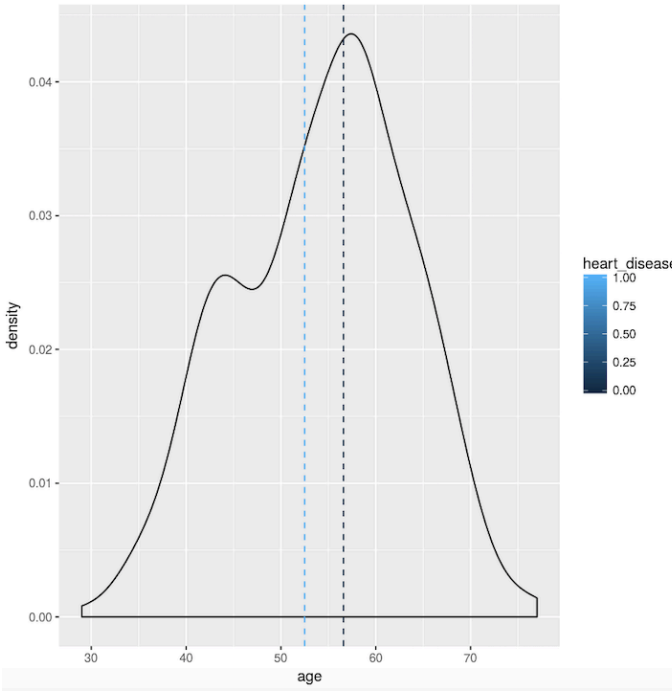
Then, with this profile, we conducted an analysis around age, and started by checking the distribution of age groups (groups of 16 years range). This revealed that the dataset contained a larger proportion of “Elderly” which might have biased visualizations (7). Therefore, in order

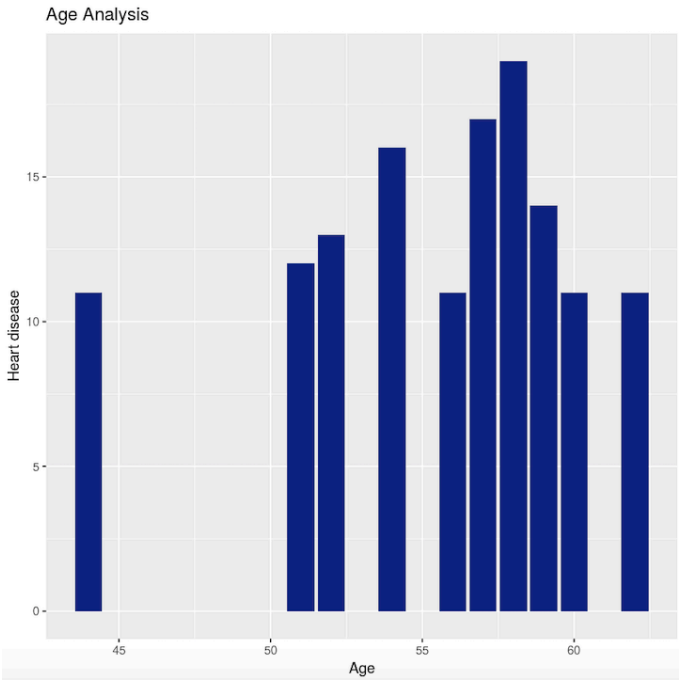
to see the impact of age on heart disease, we made a density chart with the mean age for heart disease and health (8). We found that even though there were more older people with heart disease (9), the mean age of sick people was younger than the one of healthy people (8).



(7)

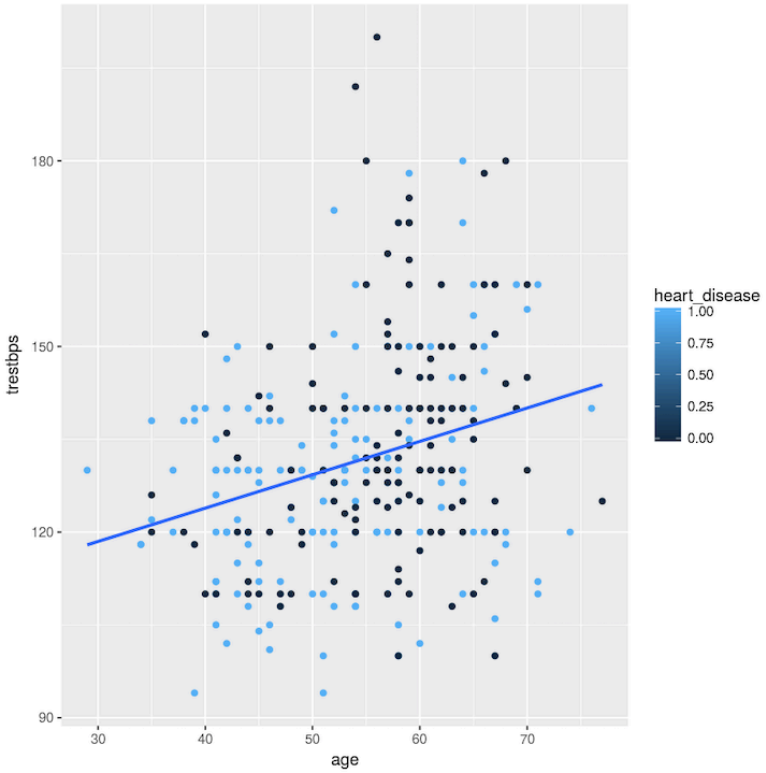
(8)



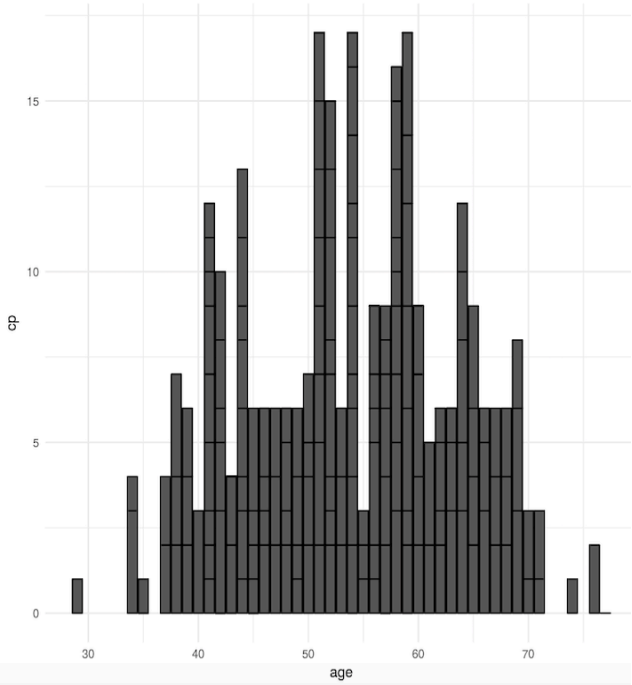


(9)

Then, we could realise other demographic analysis, bearing in mind this bias. We identified that older patients had higher resting blood pressure (trestbps), and that higher restecg were positively correlated to the proportion of sick patients (10). We also found that patients between 50 and 60 had higher cp than the others (11).



(10)



(11)

III- Preparing the Data

First, we checked if there were any missing values using the function “summary” and found that there as none. Then, we chose to derive the target variable (renamed heart_disease) as our target attribute.

Moreover, From the correlogram above, no variables appeared to be highly correlated, which made it reasonable to take the 14 variables into account in the modelling section.

Then, because the variables in the dataset could take very different ranges, some being categorical, other continuous, we decided to normalize the data so that variables with a wider range like chol would not unnecessarily have more weight in the prediction.

Once these checks done, we were ready to split the data into the training (75% of the data) and testing (25%) set using the caTools library. Later on, we made 2 other sub-datasets according to the most predictive variables, for model optimization and validation set for the ensemble models.

IV- Generate and Test Prediction Models

As we have to predict a discontinuous target variable (0: healthy, 1: Sick), we faced a classification problem, so we fitted three models: Logistic regressions, regression trees and XgBoost.
The scoring rule used was the BSS (Brier skill score), where a score of 1 corresponds to a perfect forecast. The closer to -∞ it gets, the worst the prediction. It is most appropriate to binary predictions, therefore to our problem. This scoring rule is based on relative accuracy of models compared to a mean squared error function. Therefore, it has to be used as a comparative tool for relative model accuracy.
Then, we computed a weighted average and two stacker ensemble models.

1) Logistic regressions (LR)

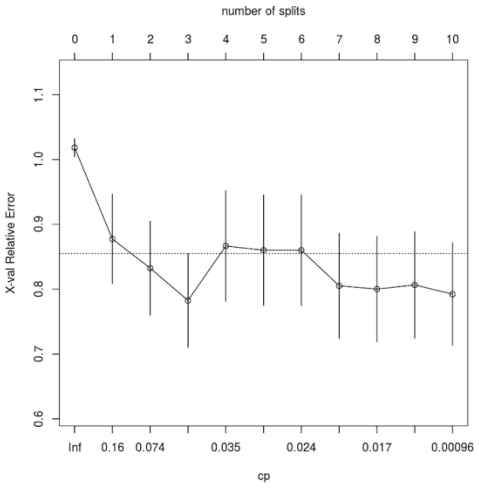
We started by the LR to check the significance of the variables. That way, we were able to filter the variables for our two first models, using the summary function of the initial LR. Thus, we created 2 sub training and testing datasets: the first (train.select1 and test.select1) only with ***, ** and * variables. The second one (train.select2 and test.select2) was made only with the ** and *** variables. For all three datasets, we did two LR, one classic, and one with interactions (^2). The best score was achieved by the select1 dataset compared to the initial one.

	Initial	Select2	Select1
simple	0,522	0.44	**0.525**
With interactions	**-0.53**	0.49	0.34

2) Regression trees

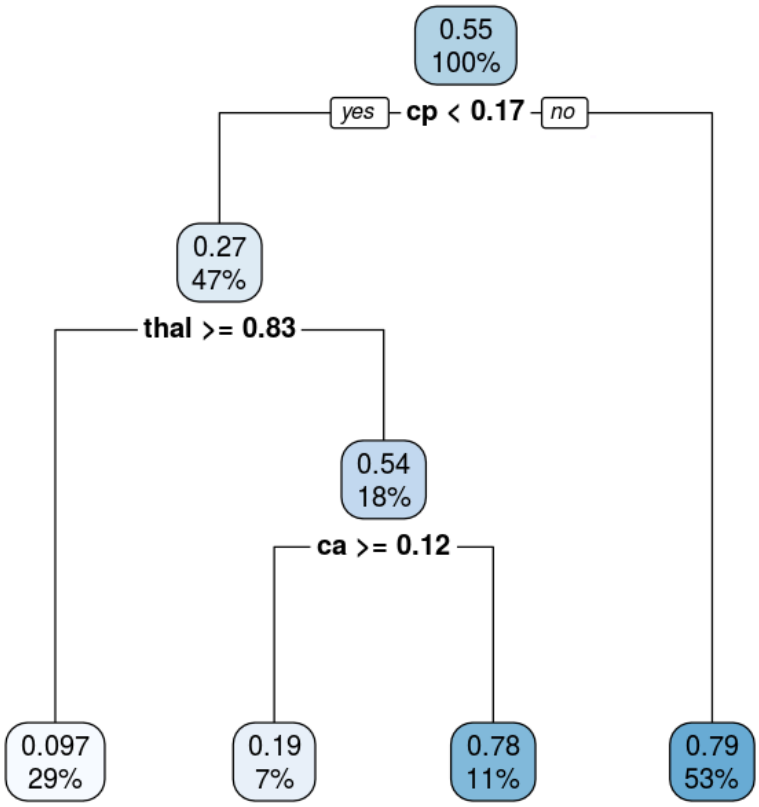
We proceeded the same way for Regression trees and computed one initial, one with select1, and one with the 4 less important variables of the variable importance table. These models will be practical for interpretation as they give visual output. The highest performance, achieved with the initial dataset, was better than the best LR.

	Initial	Varimp	Select1
BSS	**0.56**	**0.37**	0.46



We plotted pruned trees in order to avoid overfitting, by selecting a tree size minimizing the cross validation. Here is our best performing tree and the variable importance.

cp	14.9802040169558
thal	10.6268947507949
thalach	9.97382137663388
exang	6.79265303072854
oldpeak	6.33650381361339
age	4.01976487557555
slope	3.65524622374901
ca	3.4663669911695
trestbps	3.19560010861084
sex	2.00059215694427
chol	1.54049628659692



This tree illustrates that cp is the most important variable in predicting heart_disease. This can serve as a yes/no checklist for hospital staff. Starting with the CP, they can continue examining their patients against the decision tree to see if they should do further test or let them go. The tree also gives the chances of heart disease, so hospitals can decide what risk they are willing to take.

3) XgBoosts

We started by running a simple XgBoost model, which gave us a more in-depth relation analysis between variables. It gave us the relative importance of each variable when taken as predictors or our target variable.

Feature	Gain	Cover	Frequency
cp	0.301856020	0.073300203	0.046573876
age	0.148458398	0.129983623	0.248929336
oldpeak	0.085908475	0.151315789	0.103854390
ca	0.084561748	0.028461712	0.036937901
chol	0.083766976	0.215594646	0.167558887
thalach	0.079026726	0.151485204	0.137044968
thal	0.062276569	0.039332505	0.028907923
trestbps	0.058060004	0.167099616	0.146145610
sex	0.052134618	0.011181387	0.028372591
slope	0.021932399	0.014612040	0.018736617
exang	0.014284409	0.009924893	0.014989293
fbs	0.004751224	0.001270612	0.004817987
restecg	0.002982434	0.006437768	0.017130621

Therefore, we ran a second XgBoost without the fbs and restecg variables, which gave us a slightly higher score of **0,378** compared to 0.376 for the first one

4) Ensemble model

We build a weighted average (WA) and two stacker models. The WA achieved the best score ever gotten **(BSS = 0.57).** We found that this model was best performing when putting 50% of the weight in our two best performing models (LR and regression tree).

0.45035146	4.326390e-01	3.781702e-01	2.869450e-01	1.589634e-01	-5.774591e-03	-2.072690e-01	-4.455198e-01	-7.20527e-01
0.48795246	5.056552e-01	4.866015e-01	4.307914e-01	3.382249e-01	2.089020e-01	4.282267e-02	-1.600130e-01	-1.00000e+12
0.47534308	5.284609e-01	5.448223e-01	5.244273e-01	4.672759e-01	3.733681e-01	2.427039e-01	-1.000000e+12	-1.00000e+12
0.41252332	5.010562e-01	5.528328e-01	5.678529e-01	5.461166e-01	4.876239e-01	-1.000000e+12	-1.000000e+12	-1.00000e+12
0.29949318	4.234412e-01	5.106328e-01	5.610681e-01	5.747469e-01	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.00000e+12
0.13625266	2.956158e-01	4.182225e-01	5.040729e-01	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.00000e+12
-0.07719824	1.175800e-01	2.756019e-01	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.00000e+12
-0.34085952	-1.106662e-01	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.00000e+12
-0.65473118	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.000000e+12	-1.00000e+12

row	col
5	5

The results of the stacker models confirmed the finding that XgBoost model did not give an accurate prediction. Even if overall, stacking worked less efficiently, the first model using the regression tree performed way better **(0.49)** than the second using XgBoost **(0.19).**

V- <u>Problem conclusions and recommendations</u>

	Highest BSS Score
Logistic regression	0.52
Regression trees	0.56
XgBoost	0.378
Ensemble	**0.57**

To conclude, the most accurate model to predict whether a patient as a heart disease or not is our WA ensemble model. With this model, hospital staff will be able to tell if it is necessary to admit a patient in the hospital or not, given their health data. Moreover, as we have identified the most predictive variables, hospitals will be able to cut down their costs by limiting the tests to the most revealing ones (heart rate, slope of ST segment, and resting ECG) combined with the patient's age, sex and level of chest pain.

Following this analysis, we can advise hospitals to be more reluctant toward admitting young patients (30-50 years old), with an upward sloping ST segment, low level of chest pain, low maximum heart rate (140pbm) and low ECG (45mm/sec). If a patient has different characteristics, we would recommend doing further tests.

However, our findings have some limits because of (i) the nature of the dataset and (ii) the nature of our analysis.
In fact, one should take into account that the prediction models are based on 1/5 of the variables of the original dataset, so we might miss other importantly predictive variables. The

dataset is also quite small with only 303 rows which did not allow us to train more complex models such as Neural Networks, neither to filter the most predictive variables too much (E.g. select2 always has lower BSS).
Moreover, doctors already know what the most common patient characteristics of an ill patient are, so this would only help less qualified hospital secretaries. There is also a psychological question in this functioning because patients that worry about their heart might not want to leave the hospital without doing any tests or seeing a doctor.

References

Kaggle.com. 2020. *Heart Disease UCI*. [online] Available at: <https://www.kaggle.com/ronitf/heart-disease-uci#heart.csv> [Accessed 23 April 2020].

Consultant, M., 2013. [online] Healthcare.mckinsey.com. Available at: <https://healthcare.mckinsey.com/sites/default/files/the-trillion-dollar-prize.pdf> [Accessed 23 April 2020].

Rappleye, E., 2020. *Value-Based Care Putting Pressure On Nonprofit Hospital Finances, Survey Finds.* [online] Beckershospitalreview.com. Available at: <https://www.beckershospitalreview.com/finance/value-based-care-is-squeezing-nonprofit-hospital-finances-survey-finds.html> [Accessed 23 April 2020].

Institute for Health Metrics and Evaluation. 2015. *Deaths From Cardiovascular Disease Increase Globally While Mortality Rates Decrease*. [online] Available at: <http://www.healthdata.org/news-release/deaths-cardiovascular-disease-increase-globally-while-mortality-rates-decrease> [Accessed 23 April 2020].

Archive.ics.uci.edu. 2020. *UCI Machine Learning Repository: Online Shoppers Purchasing Intention Dataset Data Set*. [online] Available at: <https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset> [Accessed 23 April 2020].

Code for Heart disease

April 23, 2020

1 Importing libraries

In [ ]:
```
.libPaths("/usr/local/lib/R/site-library")
library(xgboost)
library(randomForest)
library(rpart)
library(lubridate)
library(doSNOW)
library(foreach)
library(parallel)
library(ggplot2)
library(scales)
library(dplyr)
library(caTools)
library(foreach)
library(Matrix)
library(Metrics)
```

2 Cores

In [ ]:
```
#Set our multiple cores as separate workers and cluster them.
workers <- detectCores()
#Ask the number of cores we have on our laptop
workers
cluster <- makeCluster(workers, type = "SOCK")
registerDoSNOW(cluster)
```

3 Import and familiarise with the dataset

In [ ]:
```
data <- read.csv('heart.csv')
```

In [ ]:
```
head(data)
```

In [ ]:
```
dim(data)
```

In [ ]:
```
summary(data)
```

3.1 Dealing with missing values

After analysing the summary of data we observe that no null/NA is present in our data but all the columns are considered as numeric even the categorical data as well. So, we will perform some data preprocessing steps to convert the categorical column to factor.

```
In [ ]: #Rename the target column for better understanding.
        colnames(data)[colnames(data)=="target"] <- "heart_disease"

In [ ]: head(data)
```

3.2 Visualisations

Correlation heatmap and table

```
In [ ]: #Correlation heatmap
        Rate <- data
        res<-cor(data[,c(1:14)],use="complete.obs")
        col<- colorRampPalette(c("blue", "white", "red"))(20)
        heatmap(x = res, col = col, symm = TRUE)

In [ ]: #Correlation table
        targetCol <- which(names(data)=="heart_disease")
        startCol <- which(names(data)=="age")
        endCol <- which(names(data)=="thal")
        sort(cor(data[,c(targetCol, startCol:endCol)],
                 use="complete.obs")[1,],decreasing = TRUE) [1:5]
```

Agregation table

```
In [ ]: #Aggregation table
        aggregate(cbind(cp,thalach, slope,restecg) ~ heart_disease,
                                data= data,
                                FUN=mean)
```

3.2.1 Age

age group distribution plot

```
In [ ]: # Group the different ages in three groups (young, middle, old)
        young <- data[which((data$age<45)), ]
        middle <- data[which((data$age>=45)&(data$age<55)), ]
        elderly <- data[which(data$age>55), ]
        groups <- data.frame(age_group = c("young","middle","elderly"),
                             group_count = c(NROW(young$age),
                                             NROW(middle$age),
                                             NROW(elderly$age)))

        #ploting different age groups
        ggplot(groups, aes(x=groups$age_group,
```

```
                        y=groups$group_count, fill=groups$age_group)) +
  ggtitle("Age Analysis") +
  xlab("Age Group")  +
  ylab("group Count") +
  geom_bar(stat="identity") +
  scale_fill_discrete(name = "Age Group",
                      labels = c("Elderly", "Middle", "Young"))
```

Heart disease by age

```
In [ ]: data %>%
          group_by(age) %>%
          count() %>%
          filter(n > 10) %>%
          ggplot()+
          geom_col(aes(age, n), fill = "navyblue")+
          ggtitle("Age Analysis") +
          xlab("Age")  +
          ylab("AgeCount")
```

Density plot

```
In [ ]: library(plyr)
        mu <- ddply(data, "heart_disease", summarise, grp.mean=mean(age))
        head(mu)

        ggplot(data, aes(x= age, color= heart_disease)) +
          geom_density()

        p<-ggplot(data, aes(x= age, color= heart_disease)) +
          geom_density()+
          geom_vline(data=mu, aes(xintercept=grp.mean,
                                  color= heart_disease),
                     linetype="dashed")
        p
```

Blood pressure, age and heart disease

```
In [ ]: ggplot(data, aes(x=age,y=trestbps)) +
        geom_point(aes(color= heart_disease)) +
        geom_smooth(method='lm',se=FALSE)
```

Chest pain level by age

```
In [ ]: ggplot(data, aes(x=age, y=cp))+
        geom_bar(stat="identity", color="black")+
        scale_fill_manual(values=c("#999999", "#E69F00", "#56B4E9"))+
        theme_minimal()
```

4 Data preparation

4.1 Normalizing the data

```
In [ ]: normalize <- function(x) {
          return ((x - min(x)) / (max(x) - min(x)))
        }

In [ ]: head(data)

In [ ]: ## Creating a copy of the original data.
        data_norm <- data

        ## Normalizing our 10 variables.
        data_norm$age <- normalize(data$age)
        data_norm$sex <- normalize(data$sex)
        data_norm$cp <- normalize(data$cp)
        data_norm$trestbps <- normalize(data$trestbps)
        data_norm$chol <- normalize(data$chol)
        data_norm$fbs <- normalize(data$fbs)
        data_norm$restecg <- normalize(data$restecg)
        data_norm$thalach <- normalize(data$thalach)
        data_norm$exang <- normalize(data$exang)
        data_norm$oldpeak <- normalize(data$oldpeak)
        data_norm$slope <- normalize(data$slope)
        data_norm$ca <- normalize(data$ca)
        data_norm$thal <- normalize(data$thal)
        data_norm$heart_disease <- normalize(data$heart_disease)
```

4.2 Split into training and testing dataset

```
In [ ]: #Create a training set and a test set library(caTools)
        set.seed(201)
        sample = sample.split(data_norm, SplitRatio = 0.75)
        train = subset(data_norm, sample == TRUE)
        test = subset(data_norm, sample == FALSE)
```

4.3 Creating sub training and testing sets for Logistic regression

```
In [ ]: print(colnames(train))

In [ ]: #We create new set without the variables that we
        #identify useless in the data preparation
        train.select1 <- train[,c(2,3,4,8,9,10,12,13,14)]
        test.select1 <- test[,c(2,3,4,8,9,10,12,13,14)]
        train.select2 <- train[,c(2,3,8,12,14)]
        test.select2 <- test[,c(2,3,8,12,14)]
        train.varimp <- train[,-c(2,4,5,12)]
        test.varimp <- test[,-c(2,4,5,12)]
```

5 Scoring Rule

```
In [ ]: # Write the mean return function to calculate the Brier Score.
        brier.score <- function(pred, real) {
        return(mean((pred - real)^2)) }
```

```
In [ ]: #Use the propotion of people having the disease
        #in the smaller traning set as the prediction for all the people
        #in the validation set
        #We call it the naive prediction.
        train.disease.rate <- mean(train$heart_disease)
        test.realizations <- test$heart_disease
        naive.pred <- rep(train.disease.rate, length(test.realizations))
        brier.ref <- brier.score(naive.pred, test.realizations)
```

```
In [ ]: #Write function according to the predicted value,
        #the actual value and the brier ref calculated above
        #We then get the Brier Skill Score
        skill.score <- function(pred, real, brier.ref) {
        brier.score <- brier.score(pred, real)
        return(1 - brier.score/ brier.ref) }
```

6 Models

6.1 Logistic regression

6.1.1 Logistic regression with all variables

6.1.2 without interactions

```
In [ ]: logit.reginit <- glm(heart_disease~., data = train,
                             family = "binomial")
        pred.logit.reginit <- predict(logit.reginit, test,
                                      type = "response")
```

```
In [ ]: #Calculate the score
        skill.score(pred.logit.reginit, test$heart_disease, brier.ref)
```

```
In [ ]: summary(logit.reginit)
```

6.1.3 With interractions

```
In [ ]: logit.reginit1 <- glm(heart_disease ~ . + .^2 ,
                              data = train, family = "binomial")
        pred.logit.reginit1 <- predict(logit.reginit1, test,
                                       type = 'response')
```

```
In [ ]: skill.score(pred.logit.reginit1, test$heart_disease, brier.ref)
```

6.1.4 Logistic regression with train.select1

6.1.5 Without interactions

```
In [ ]: logit.reg1 <- glm(heart_disease~., data = train.select1,
                         family = "binomial")
        pred.logit.reg1 <- predict(logit.reg1,
                                   test.select1, type = "response")

In [ ]: #Calculate the score
        skill.score(pred.logit.reg1, test$heart_disease, brier.ref)
```

6.1.6 With interactions

```
In [ ]: logit.regint1 <- glm(heart_disease ~ . + .^2 ,
                            data = train.select1, family = "binomial")

        pred.logit.regint1 <- predict(logit.regint1,
                                      test.select1, type = 'response')

In [ ]: skill.score(pred.logit.regint1,test$heart_disease, brier.ref)
```

6.1.7 Logistic regression with train.select2

6.1.8 without interactions

```
In [ ]: logit.reg2 <- glm(heart_disease~., data = train.select2,
                         family = "binomial")
        pred.logit.reg2 <- predict(logit.reg2, test.select2,
                                   type = "response")

In [ ]: #Calculate the score
        skill.score(pred.logit.reg2, test$heart_disease, brier.ref)
```

6.1.9 Logistic regression with interactions

```
In [ ]: logit.regint2 <- glm(heart_disease ~ . + .^2 ,
                            data = train.select2, family = "binomial")
        pred.logit.regint2 <- predict(logit.regint2,
                                      test.select2, type = 'response')

In [ ]: skill.score(pred.logit.regint2, test$heart_disease, brier.ref)
```

6.2 Regression tree

6.2.1 Regression tree without chosing variables

```
In [ ]: reg.tree <- rpart(heart_disease ~ ., data = train,
                         control = rpart.control(cp = 0.0001 ) )
        reg.tree$variable.importance
```

```
In [ ]: reg.tree.pred <- predict(reg.tree,newdata= test,type="vector")

In [ ]: #Calculate score
        skill.score(reg.tree.pred, test$heart_disease, brier.ref)

In [ ]: plotcp(reg.tree, upper = "splits")

In [ ]: reg.tree.opt <- prune(reg.tree, cp = 0.05)

In [ ]: library("rpart.plot")
        rpart.plot(reg.tree.opt)
```

6.2.2 Regression tree with train.select1

```
In [ ]: print(colnames(train))

In [ ]: reg.tree1 <- rpart(heart_disease ~ ., data = train.select1,
                          control = rpart.control(cp = 0.0001 ) )
        reg.tree1$variable.importance

In [ ]: reg.tree1.pred <- predict(reg.tree1,newdata= test.select1,
                                  type="vector")

In [ ]: #Calculate score
        skill.score(reg.tree1.pred, test$heart_disease, brier.ref)

In [ ]: plotcp(reg.tree1, upper = "splits")

In [ ]: reg.tree1.opt <- prune(reg.tree, cp = 0.0020)

In [ ]: library("rpart.plot")
        rpart.plot(reg.tree1.opt)
```

6.2.3 Regression tree with variable importance

```
In [ ]: reg.tree2 <- rpart(heart_disease ~ ., data = train.varimp,
                          control = rpart.control(cp = 0.0001 ) )
        reg.tree2$variable.importance

In [ ]: reg.tree2.pred <- predict(reg.tree2,newdata= test.varimp,
                                  type="vector")

In [ ]: #Calculate score
        skill.score(reg.tree2.pred, test$heart_disease, brier.ref)
```

7 XgBoost

7.1 XgBoost with all variables

```
In [ ]: # Create sub datasets necessary to run a random forest
        xtrain <- train[,-c(14)]
        #The variable "heart_disease" is the one we are predicting
        ytrain <- train[, 14]
        xtest <- test[,-c(14)]
        ytest <- test[, 14]
```

```
In [ ]: #Create sub datasets because XG Boost does not work with
        #non-numeric values.
        xtrain.xgb <- model.matrix(~ 0 + ., data = xtrain)
        ytrain.xgb <- as.vector(ytrain)
        xtest.xgb <- model.matrix(~ 0 + ., data = xtest)
        ytest.xgb <- as.vector(ytest)
```

```
In [ ]: xgb_m1 <- xgboost(xtrain.xgb,
                         ytrain, max.depth = 18,
                         nthread = workers,
                         nround = 200,
                         objective = "reg:linear",
                         verbose = 0)
        xgb.pred_m1 <- predict(xgb_m1, xtest.xgb)
```

```
In [ ]: skill.score(xgb.pred_m1, ytest.xgb, brier.ref)
```

```
In [ ]: #We want to display the contribution of each variable
        #in this model.
        xgb.importance(colnames(xtrain.xgb), model = xgb_m1)
        print(cbind(1: ncol(xtrain),(colnames(xtrain))))
```

7.2 XgBoost without the 2 less important variables

```
In [ ]: xgb_m2 <- xgboost(xtrain.xgb[,-c(6,7)],
                         ytrain, max.depth = 18,
                         nthread = workers,
                         nround = 200,
                         objective = "reg:linear",
                         verbose = 0)
        xgb.pred_m2 <- predict(xgb_m2, xtest.xgb[,-c(6,7)])
```

```
In [ ]: skill.score(xgb.pred_m2, ytest.xgb, brier.ref)
```

8 Ensemble model

8.0.1 Wheighted average

```
In [ ]: m1_weight <- seq(0.1,0.9,0.1)
        m2_weight <- seq(0.1,0.9,0.1)
```

```
# set up a matrix to store the Brier skill scores
bss_matrix <- matrix(0,9,9)
for (i in 1:9) {
    for (j in 1:9){
        if (m2_weight[j]+ m1_weight[i] > 1)
        {bss_matrix[i,j] = -1000000000000
         #lower limit is minus infinity
        }else{
            ensemble_pred <- m1_weight[i]*pred.logit.reg1 +
            reg.tree.pred*m2_weight[j] +
            xgb.pred_m2*(1-m1_weight[i] - m2_weight[i])
                bss_matrix[i,j] <- skill.score(ensemble_pred,
                                               test$heart_disease,
                                               brier.ref)
            }
} }
bss_matrix
which(bss_matrix == max(bss_matrix), arr.ind = TRUE)
```

8.0.2 Stacking

In [ ]:
```
(nrowTrain <- nrow(train))
(nrowSmallTrain <- round(nrowTrain*0.75))
(nrowvalid <- nrowTrain - nrowSmallTrain)

set.seed(201)
# generate row numbers of the training set.
rowIndicesSmallTrain <- sample(1:nrowTrain,
                               size = nrowSmallTrain,
                               replace = FALSE)
smalltrain.df <- train[rowIndicesSmallTrain, ]
valid.df <- train[-rowIndicesSmallTrain, ]
```

In [ ]:
```
smalltrain1.df<- smalltrain.df[,c(2,3,4,8,9,10,12,13,14)]
valid1.df <- valid.df[,c(2,3,4,8,9,10,12,13,14)]
train.select2 <- train[,c(2,3,8,12,14)]
test.select2 <- test[,c(2,3,8,12,14)]
```

In [ ]:
```
M1 <- lreg <- glm(heart_disease~., data = smalltrain1.df,
                  family = "binomial")
M1.pred <- predict(lreg, valid1.df, type = "response")
```

In [ ]:
```
# Regression tree
M2 <- rpart(heart_disease ~ .,
                  data = smalltrain.df,
                  control = rpart.control(cp = 0.0001 ) )

M2.pred <- predict(reg.tree2,newdata= valid.df,type="vector")
```

In [ ]:
```
# XGBOOST
smallxtrain.xgb <- model.matrix(~ 0 + ., data = smalltrain.df[,-14])
smallytrain.xgb <- as.vector(smalltrain.df$heart_disease)

xvalid.xgb <- model.matrix(~ 0 + ., data = valid.df[,-14])
yvalid.xgb <- as.vector(valid.df$heart_disease)

M3 <- xgboost(smallxtrain.xgb[,-c(6,7)], smallytrain.xgb,
                      max.depth = 18,
                      nthread = workers,
                      nround = 200,
                      objective = "reg:linear",
                      verbose = 0)


M3_pred <- predict(M3, xvalid.xgb[,-c(6,7)])
```

In [ ]:
```
stacker.df <- data.frame(heart_disease = valid.df$heart_disease,
                            M1.pred = M1.pred,
                            M2.pred = M2.pred,
                            M3.pred = M3_pred)

head(stacker.df)
```

In [ ]:
```
#Fit a regression tree to the predictions as stacker model 1
stackerModel_1 <- rpart(heart_disease ~ .,
                    data = stacker.df,
                    control = rpart.control(cp = 0.0004))
```

In [ ]:
```
#Fit an XGBOOST model to the predictions as stacker model 2
stacker.df.xgb <- model.matrix(~ 0 + ., data = stacker.df[,-1])
stacker.y <- as.vector(stacker.df[,1])
stackerModel_2 <- xgboost(stacker.df.xgb, stacker.y,
                      max.depth = 18,
                      nthread = workers,
                      nround = 200,
                      objective = "reg:linear",
                      verbose = 0)
```

In [ ]:
```
M1.trainAll <- glm(heart_disease~., data = train.select1,
                    family = "binomial")

M2.trainAll <- rpart(heart_disease ~ .,
                    data = train,
                    control = rpart.control(cp = 0.0001 ) )

M3.trainAll <- xgboost(xtrain.xgb[,-c(6,7)], ytrain,
                      max.depth = 18,
                      nthread = workers,
```

```
                            nround = 200,
                            objective = "reg:linear",
                            verbose = 0)
```

In [ ]:
```
M1.predict.test <- predict(M1.trainAll, test, type = "response")
M2.predict.test <- predict(M2.trainAll, test)
M3.predict.test <- predict(M3.trainAll, xtest.xgb)
```

In [ ]:
```
predict.variables <- data.frame('M1.pred' = M1.predict.test,
                                'M2.pred' = M2.predict.test,
                                'M3.pred' = M3.predict.test)

# Predict from stacker model 1 --- regression tree
stacker.predict.rt <- predict(stackerModel_1, predict.variables)

# Score the stacker model's prediction
skill.score(stacker.predict.rt, test$heart_disease, brier.ref)
```

In [ ]:
```
# Predict from stacker model 2 --- XGBOOST,
#and calculate the Brier Skill Score
predict.variables.xgb <- model.matrix(~ 0 + .,
                                      data = predict.variables)
stacker.predict.xgb <- predict(stackerModel_2,
                               predict.variables.xgb)

# Score the stacker model's prediction
skill.score(stacker.predict.xgb, test$heart_disease, brier.ref)
```