Heart Disease Prediction Model

Bishwas Ghimire

12/02/2019

Introduction and Objective

Early diagnosis can be a game changer when it comes to treating life-threatening diseases like heart disease. Therefore, doctors and researchers are always looking for symptoms, markers, or other tell-tale methods that can alert them to their patient's increased health risk as early in the treatment process as possible.

A multitude of variables are involved in a patient's health outcome, especially in something as complicated as the presence or absense of heart disease. A significant subset of these variables can be quantified in different ways and encoded as numeric values or categorical data. This situation naturally qualifies as one where the sheer magnitude of information available on the subject can easily be overwhelming to a single or a small group of doctors or human scientists. In this scenario, any help that we can get from today's superfast computers and advanced machine learning algorithms is, we think, highly desirable. This is where data science tools and algorithms can come in handy by screening through large piles of data to help doctors narrow down their diagnosis and make informed treatment decisions early on. This is what motivates us to work in this project.

Our main goal is to develop a machine learning model to predict the presence (or risk) of heart disease based on variables like age, sex, blood pressure, chest pain type, maximum heart rate, resting electrocardiographic results, etc. that are likely to have an impact on heart condition.

Data Exploration

We begin by importing the dataset.

```
#import hear data
heart.data = read.csv(file="heart.csv", header=TRUE)
summary(heart.data)
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0
##  1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0
##  Median :55.00   Median :1.0000   Median :1.000   Median :130.0
##  Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0
##       chol            fbs            restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :240.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.3   Mean   :0.1485   Mean   :0.5281   Mean   :149.6
##  3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##      exang           oldpeak         slope             ca
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.80   Median :1.000   Median :0.0000
```

```
##  Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean   :0.7294
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.   :4.0000
##       thal           target
##  Min.   :0.000   Min.   :0.0000
##  1st Qu.:2.000   1st Qu.:0.0000
##  Median :2.000   Median :1.0000
##  Mean   :2.314   Mean   :0.5446
##  3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :3.000   Max.   :1.0000
```

Here is some more information about the dataset.

Variable name	Description
age	age in years
sex	1 = male; 0 = female
cp	chest pain type (4 categories)
trestbps	resting blood pressure (in mm Hg on admission to the hospital)
chol	serum cholestoral in mg/dl
fbs	fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
restecg	resting electrocardiographic results (3 categories)
thalach	maximum heart rate achieved
exang	exercise induced angina (1 = yes; 0 = no)
oldpeak	ST depression induced by exercise relative to rest
slope	the slope of the peak exercise ST segment (3 categories)
ca	number of major vessels (5 categories) colored by flourosopy
thal	3 = normal; 6 = fixed defect; 7 = reversable defect
target	1 = heart disease; 0 = no heart disease

Since we can see that some of the explanatory variables are categorical, we must convert them into factor variables.

```
#covert factor variables
cols = c('sex','fbs','cp','restecg','exang','ca','slope','thal','target')
heart.data[cols] = lapply(heart.data[cols], factor)
```

Let's quickly look at the summary statistics again to make sure that categorical and continuous data types are now interpreted correctly.

```
summary(heart.data)
```

```
##       age        sex     cp         trestbps          chol       fbs
##  Min.   :29.00   0: 96   0:143   Min.   : 94.0   Min.   :126.0   0:258
##  1st Qu.:47.50   1:207   1: 50   1st Qu.:120.0   1st Qu.:211.0   1: 45
##  Median :55.00           2: 87   Median :130.0   Median :240.0
##  Mean   :54.37           3: 23   Mean   :131.6   Mean   :246.3
##  3rd Qu.:61.00                   3rd Qu.:140.0   3rd Qu.:274.5
##  Max.   :77.00                   Max.   :200.0   Max.   :564.0
##  restecg    thalach      exang      oldpeak     slope   ca      thal
##  0:147   Min.   : 71.0   0:204   Min.   :0.00   0: 21   0:175   0:  2
##  1:152   1st Qu.:133.5   1: 99   1st Qu.:0.00   1:140   1: 65   1: 18
##  2:  4   Median :153.0           Median :0.80   2:142   2: 38   2:166
##          Mean   :149.6           Mean   :1.04           3: 20   3:117
##          3rd Qu.:166.0           3rd Qu.:1.60           4:  5
##          Max.   :202.0           Max.   :6.20
```

```
##  target
##  0:138
##  1:165
##
##
##
##
```

We can display the head to look at the actual firt few entires in different columns.

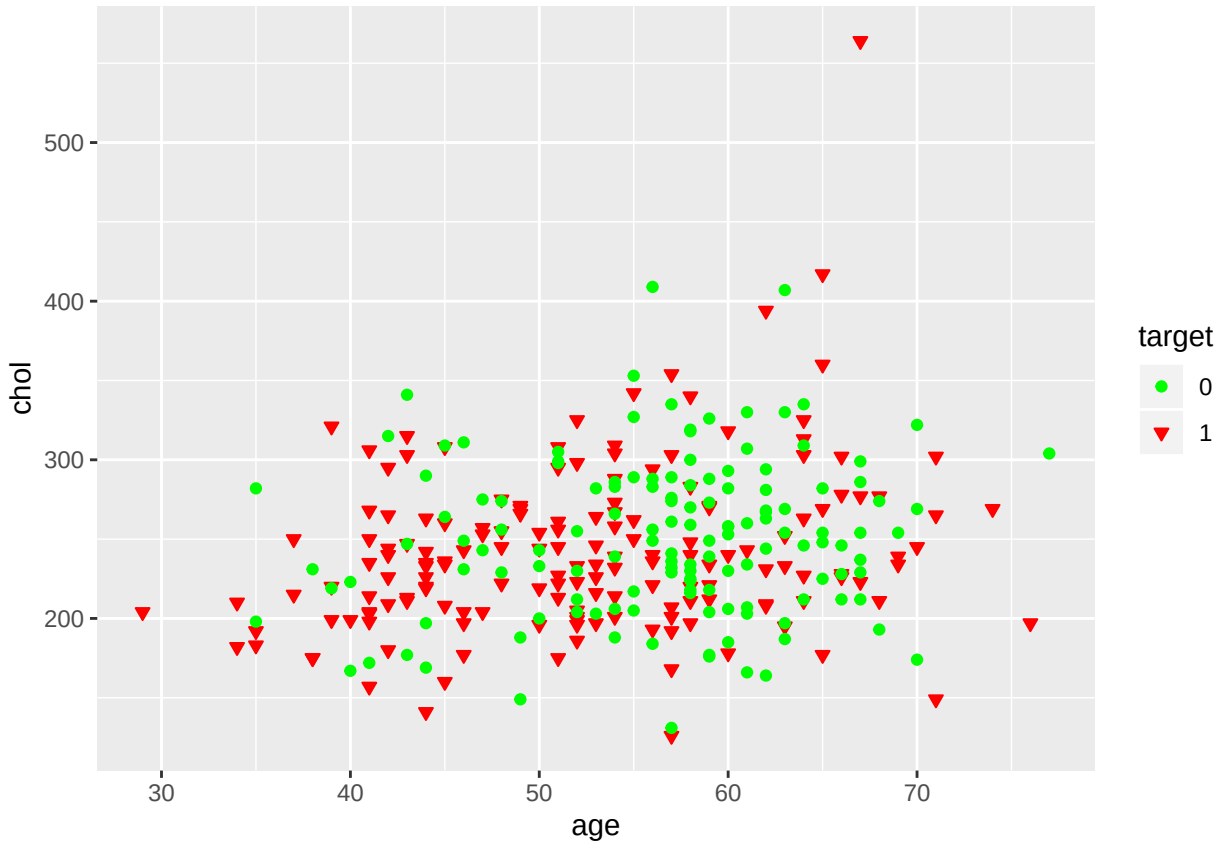
```
head(heart.data)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 1  63   1  3      145  233   1       0     150     0     2.3     0  0    1
## 2  37   1  2      130  250   0       1     187     0     3.5     0  0    2
## 3  41   0  1      130  204   0       0     172     0     1.4     2  0    2
## 4  56   1  1      120  236   0       1     178     0     0.8     2  0    2
## 5  57   0  0      120  354   0       1     163     1     0.6     2  0    2
## 6  57   1  0      140  192   0       1     148     0     0.4     1  0    1
##   target
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
```

Data Visualization

R's ggplot2 package allows us to make plots that are colored by target classes, which can be helpful. For instance, we can see if the effect of cholesterol on heart disease is different for different ages.

```
library(ggplot2)
gg = ggplot(heart.data, aes(x=age, y=chol, color = target)) +
  geom_point(aes(color = target, shape = target, fill= target)) +
  scale_colour_manual(values = c("Green","Red")) +
  scale_shape_manual(values = c(21, 25))+
  scale_fill_manual(values = c("Green","Red"))
gg
```



We can see in the plot that red traingles for target = 1 indicating the presence of heart disease appear to be clustered on one side and green circles for target = 0 indicating the absence of heart disease are clustered to the other side. Shortly speaking, whether a certain level of cholesterol is indicative of heart problem in a patient or not seems to depend on how old the patient is.

Let's plot another one of these plots for maximum heart rate achieved by the patient.

```
gg = ggplot(heart.data, aes(x=age, y=thalach, color = target)) +
  geom_point(aes(color = target, shape = target, fill=target)) +
  scale_colour_manual(values = c("Green","Red")) +
  scale_shape_manual(values = c(21, 25))+
  scale_fill_manual(values = c("Green","Red"))
gg
```

Train-test Split

After our preliminary exploration of the dataset, we can begin model development, but before we begin building any model, let's sequester a portion of the dataset for cross-validation. Below, we do a 80-20 train-test split on our original dataset.

```
################################################
#### Train-Test Split for Cross Validation #####

#define the size of the training set
train.idx <- floor((nrow(heart.data)/5)*4)

#sample rows
set.seed(13) #set seed for reproducible results
idx = sample(nrow(heart.data))
heart.data <- heart.data[idx, ]

#get training set
heart.train <- heart.data[1:train.idx, ]
#set aside test set
heart.test <- heart.data[(train.idx+1):nrow(heart.data), ]
head(heart.test)
```

```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca
## 300  45   1  3      110  264   0       1     132     0     1.2     1  0
## 214  61   0  0      145  307   0       0     146     1     1.0     1  0
```

```
## 245  56   1  0      132  184   0       0     105     1     2.1     1  1
## 42   48   1  1      130  245   0       0     180     0     0.2     1  0
## 26   71   0  1      160  302   0       1     162     0     0.4     2  2
## 45   39   1  2      140  321   0       0     182     0     0.0     2  0
##     thal target
## 300    3      0
## 214    3      0
## 245    1      0
## 42     2      1
## 26     2      1
## 45     2      1
```

Logistic Regression Model

Using the training data, we will develop a model that can help us predict our target variable, and we will cross-validate the model's performance on the sequestered test data. Since our target is a binary categorical variable, we need a binary classification algorithm. We will use a logistic regression model, which is the classification counterpart of a linear regression model. To build the logistic regression model, we can use R's glm function with parameter 'family' set as 'binomial'.

```
my.model = glm(target~age+sex+cp+thalach+exang+oldpeak+ca + I(age*thalach), data = heart.train, family =
summary(my.model)
```

```
##
## Call:
## glm(formula = target ~ age + sex + cp + thalach + exang + oldpeak +
##     ca + I(age * thalach), family = "binomial", data = heart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2741  -0.4409   0.1625   0.4947   2.3740
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -21.604009  10.421647  -2.073 0.038173 *
## age                0.371809   0.176155   2.111 0.034799 *
## sex1              -1.826618   0.483044  -3.781 0.000156 ***
## cp1                0.705477   0.564324   1.250 0.211253
## cp2                1.731339   0.527284   3.284 0.001025 **
## cp3                2.201339   0.776274   2.836 0.004571 **
## thalach            0.162540   0.067030   2.425 0.015313 *
## exang1            -0.767906   0.462378  -1.661 0.096758 .
## oldpeak           -0.841163   0.233436  -3.603 0.000314 ***
## ca1               -1.793489   0.501795  -3.574 0.000351 ***
## ca2               -2.249856   0.726302  -3.098 0.001950 **
## ca3               -1.872745   0.803625  -2.330 0.019786 *
## ca4               -0.027640   1.494777  -0.018 0.985247
## I(age * thalach)  -0.002526   0.001146  -2.205 0.027466 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 333.48  on 241  degrees of freedom
## Residual deviance: 168.18  on 228  degrees of freedom
## AIC: 196.18
##
## Number of Fisher Scoring iterations: 6
```

Interpretation

Let's briefly talk about interpreting the model results. The model shown here is a logistic regression model that can predict the binary target variable, which is the presence or absence of heart disease. The coefficients in the model can be interpreted as the marginal effect on log odds of the target variable (log(p/(1-p)) where p is the probability ranging from 0 to 1). For instance, unit increase in age increases the log odds of having heart disease by 0.5, or being male reduces the log odds by 1.51 points.

Model Development Strategy

The final model shown here is not the first model we pick. We start by taking into account all the variables that are avaiable to us. We then simplify the model by removing the terms that are statistically insignificant by using likelihood-ratio test shown below as an example.

```
###If model0 is a logistic regression submodel of model, then
###LRtest compares them with a likelihood ratio test and returns
###the p-value.
LRtest=function(model0,model){
  1-pchisq(model0$deviance-model$deviance,
           df=model0$df.residual-model$df.residual)}

big.model = glm(target~age+sex+cp+thalach+exang+oldpeak+ca + I(age*thalach)+chol+restecg+fbs, data = hea

small.model = my.model
LRtest(small.model,big.model)
```

```
## [1] 0.2446697
```

The p-value shows that we don't have evidence to reject the null hypothesis that variables 'chol', 'restecg', 'fbs' are statistically insignificant. Hence, we can drop these from our model. We visually explore the dataset to look for any suggestions of interaction among different variables. When we see something visually, we check to see if it is statistically significant. Notice that we have decided to include an interaction term (age * thalach) in our final model, because it appears to be statistically significant when checked using the likelihood ratio test.

Analysis of Deviance

To quickly see how each varaible is reducing the residual deviance, we can use 'anova'.

```
anova(my.model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: target
```

```
##
## Terms added sequentially (first to last)
##
##
##                  Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                241     333.48
## age               1   15.118       240     318.36  0.000101 ***
## sex               1   31.301       239     287.06 2.209e-08 ***
## cp                3   50.546       236     236.51 6.111e-11 ***
## thalach           1   12.271       235     224.24  0.000460 ***
## exang             1    5.222       234     219.02  0.022306 *
## oldpeak           1   21.589       233     197.43 3.378e-06 ***
## ca                4   23.907       229     173.53 8.337e-05 ***
## I(age * thalach)  1    5.344       228     168.18  0.020798 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model Diagnostics (using group plots)

Since we are working with logistic regression, the residuals don't have the same meaning and role in model diagnostics as they do for linear regression because our target variable is now binary with two distint categories. When we make the prediction using our model, we would expect to output either a 0 or a 1, signifying either a 'yes' or a 'no' for the presence or absense of heart disease. Thus, we cannot do possibly do things like checking for the normality of errors as it would make no sense to do that. The error would itself always be a 0 (if we get the target right) or a 1 (if we miss the target).

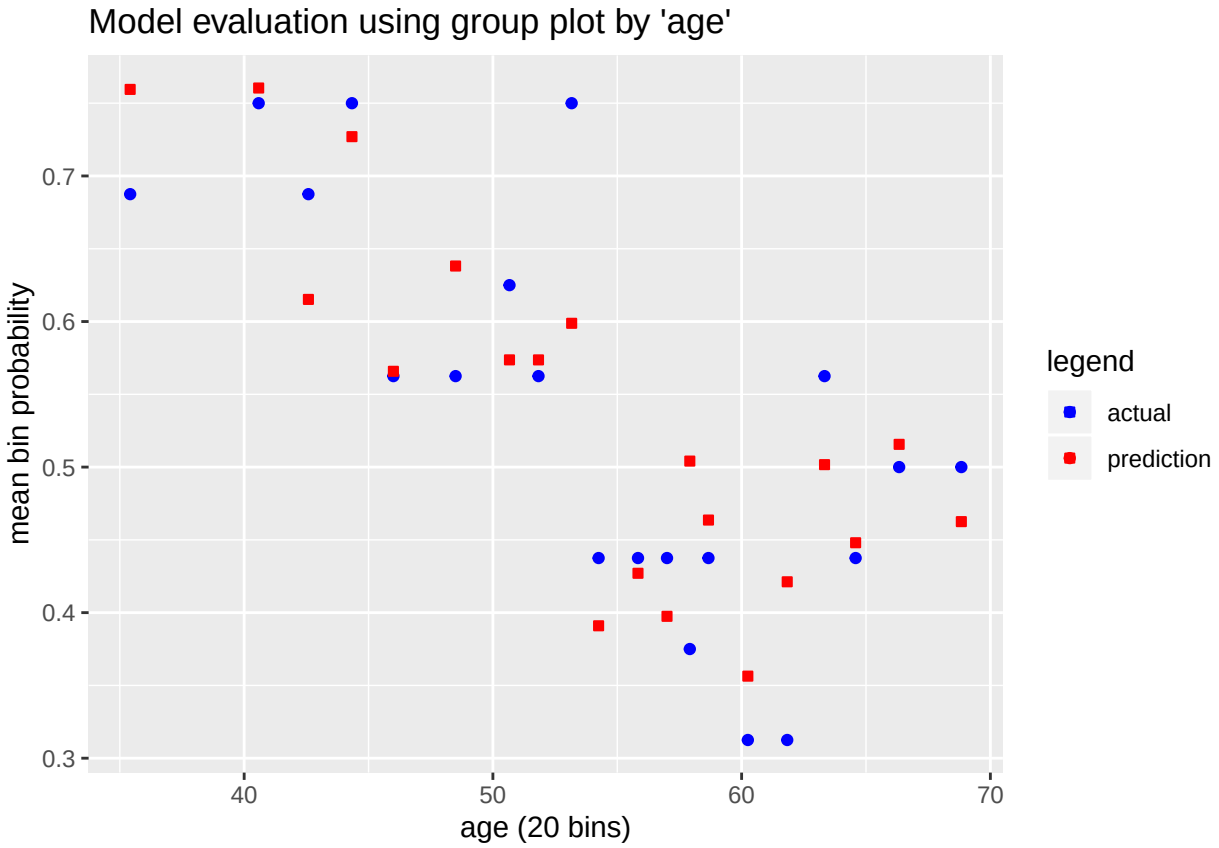
Skipping the details, in logistic regression, the output of the model is always a number between 0 and 1 and can be interpreted as the probability that the target variable is 1.

We can make group plots where we group an independent variable, *age* in our case, into many bins and plot the fraction of the targets in that bin that are true (target = 1). This number would roughly be an estimate for the probablity that the target is 1 if an individual (represented by one full row of data) belongs to that particular *age* bin.

Since we cannot analyze the residuals using conventional approaches, in order to ensure that our model is making reasonable and sensible predictions, we will look at the group plot of the fraction of true target values for each *age* bin and compare that to the group plot of the mean predicted target probabilities for that bin.

```
#sigmoid maps a real number to a number between 0 and 1
pi = function(g){
  return(exp(g)/(1+exp(g)))}
#logit calculates the logit of a number p between 0 and 1.
logit=function(p){
  log(p/(1-p))}

###Given a vector of numerical values x, a response vector Y, and a
###number of groups k, groupplot returns a list.
###x=mean of x groups
###Pi=mean of Y values in each group (with Wilson's adjustment)
###g=logit of Pi
groupplot=function(x,Y,k){
  sortframe=sort(x,index=TRUE)
  x=sortframe$x
  Y=Y[sortframe$ix]
```

```
  xmeans=1:k
  Pi=1:k
  s=floor(length(x)/k)  #groupsize

  for(i in 1:k){
    index=(((i-1)*s+1):(i*s))
    xmeans[i]=mean(x[index])
    Pi[i]=(sum(Y[index])+2)/(s+4)}

  g=logit(Pi)
  return(list(x=xmeans,g=g,Pi=Pi))}

as.numeric.factor = function(x) {as.numeric(levels(x))[x]}

#number of bins
bins = 20

group.true = groupplot(heart.train$age,as.numeric.factor(heart.train$target),bins)

y.hat = predict(my.model,newdata = heart.train, type = 'response' )
group.pred = groupplot(heart.train$age,y.hat,bins)

x1 = group.true$x
y1 = group.true$Pi
x2 = group.pred$x
y2 = group.pred$Pi
bin.e = y2-y1
df <- data.frame(x1, y1, x2, y2,bin.e)

ggplot(df) + geom_point(aes(x = x1, y = y1, color = "actual", fill='actual'), shape = 21) + geom_point(a
scale_colour_manual(name = 'legend', values = c("actual"="Blue","prediction" = "Red")) +
scale_fill_manual(name = 'legend',values = c("actual" = "Blue","prediction" = "Red")) +
labs(x = 'age (20 bins)', y = 'mean bin probability', title ='Model evaluation using group plot by \'ag
```

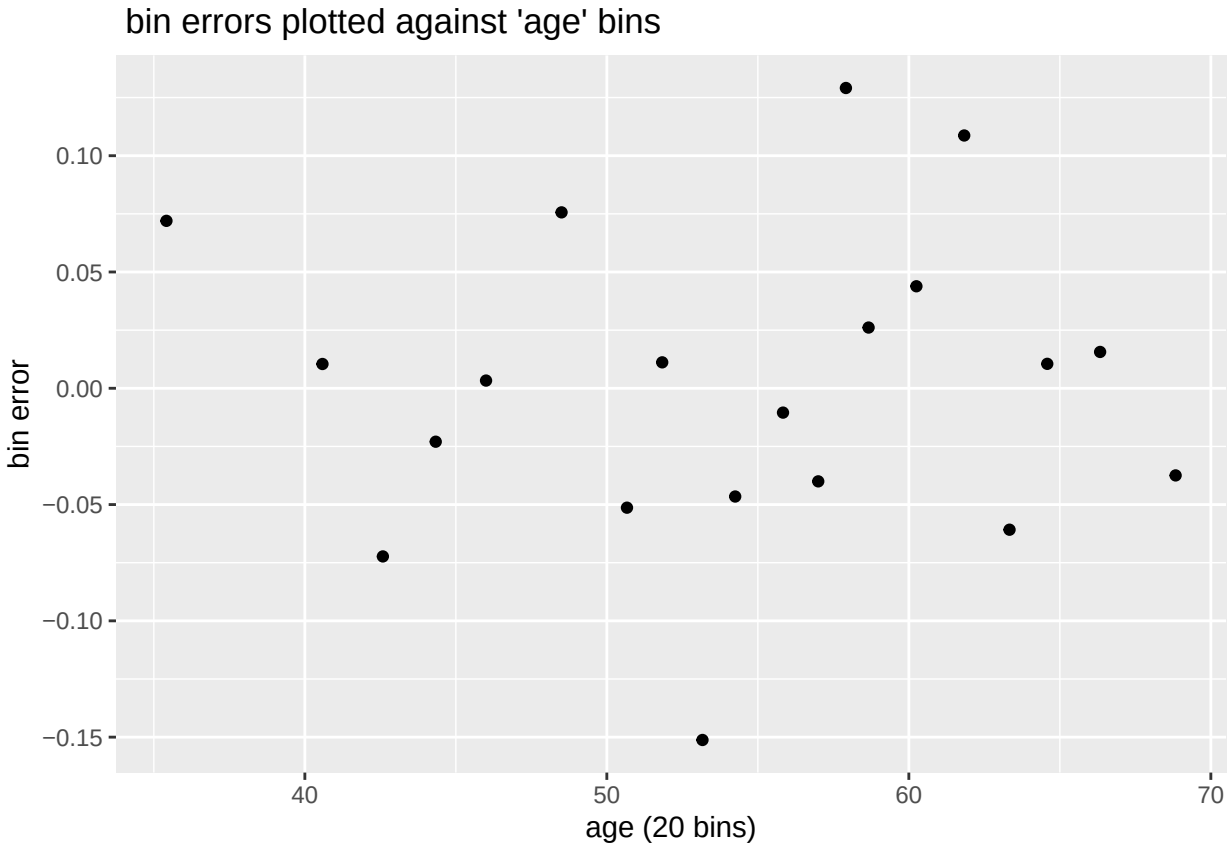


We can observe in the plot that actual and predicted values more or less follow a similar trend, which tells us the model is making reasonable predictions. Note that we have used the actual predicted probabilities ranging anywhere between 0 and 1 while averaging the prediction for each bin, not the converted binary values.

One thing we need to emphasize is the fact that even though we are only looking at an arbitrarily picked single variable, *age* in a multivariate model, while making predictions on probabilities from the logistic regression model, all indepdendent variables considered in the model are taken into account. Therefore, mean predictions for the bins incorporate the model's overall performance in predicting the target variable after considering all other relevant independent variables in addition to *age*.

We believe we can go a step further with this idea. We can determine how much we are off by in each bin (bin error) and use this as a proxy for 'residual' in linear regression models. Predicted probabilities that are used to determine the bin errors take into account all independent variables involved in training the model, so bin errors, in a real sense, measure the amount by which the model's final prediction differs from the truth; the difference is instead of doing this for a single data point, which in case of a logistic regression model doesn't really make sense, the binning concept looks at small bins of data within the larger test set in order to be able to calculate meaningful probabilities. When we plot the bin errors, we should expect to see white noise. Let's look at the plot.

```
ggplot(df) + geom_point(aes(x = x1, y = bin.e)) +
labs(x = 'age (20 bins)', y = 'bin error', title =' bin errors plotted against \'age\' bins')
```



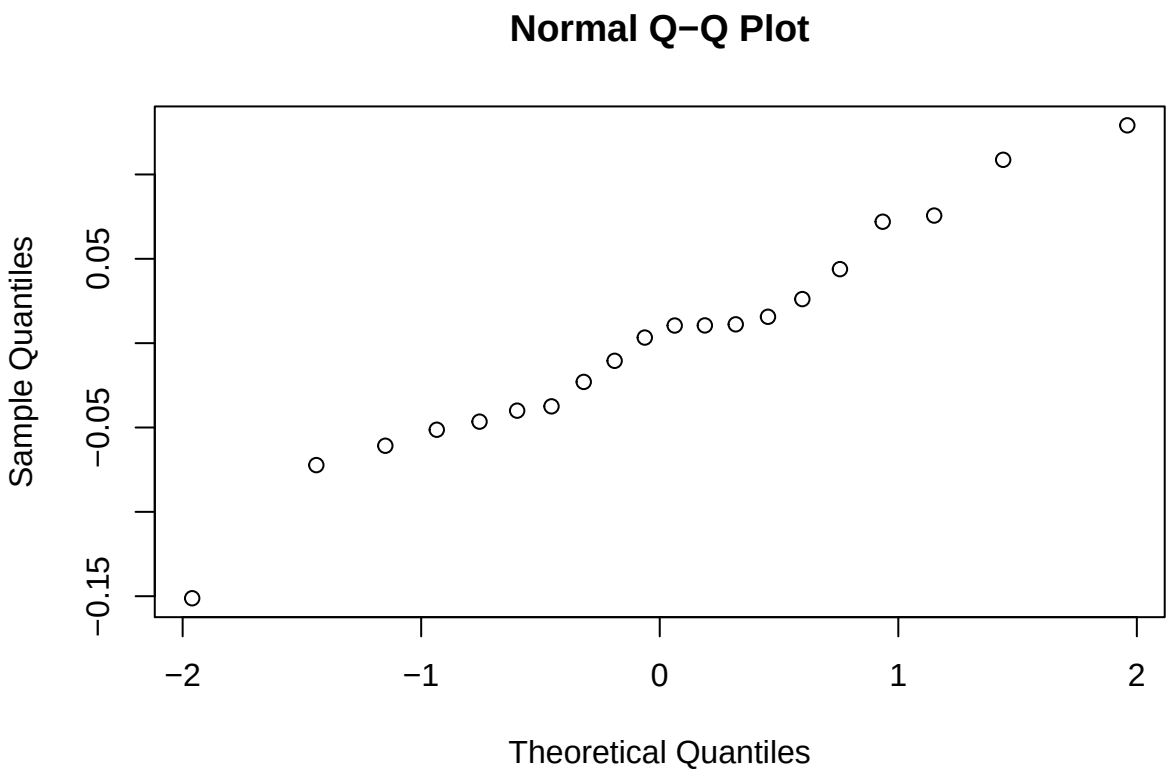
The errors appear more or less random with a mean around zero. Let's check the mean.

```
mean(bin.e)
```

```
## [1] 0.0006667994
```

Mean is very close to zero as expected. Let's visually inspect normality of the bin errors using QQ plot.

```
qqnorm(bin.e)
```



This is suggestive but not conclusive yet. We can conduct a rigorous statistical test called Shapiro-Wilks test to test the null hypothesis that errors are normally distributed.

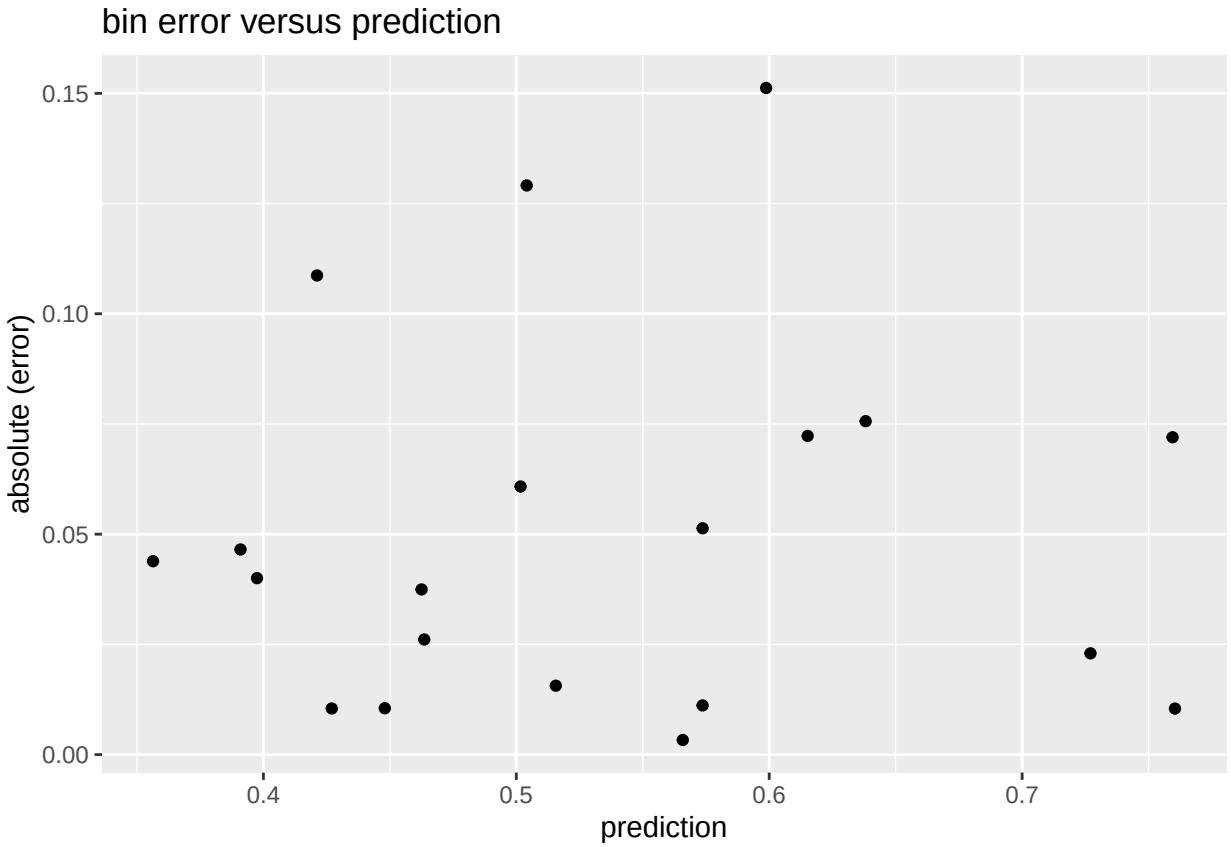
```
shapiro.test(bin.e)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  bin.e
## W = 0.97658, p-value = 0.8827
```

The p-value of 0.88 says we cannot reject the null hypothesis meaning the errors appear to be normally distributed.

We can check for constancy of bin error variance as we did for residuals in linear regression models. Let's plot the absolute value of bin errors against the predicted values.

```
abs.bin.e = abs(bin.e)
ggplot(df) + geom_point(aes(x = y2, y = abs.bin.e)) +
labs(x = 'prediction', y = 'absolute (error)', title ='bin error versus prediction')
```



The plot suggests that variance is more or less the same across all bins, but again, to be sure, we can conduct Brown-Forsythe test to test the bin errors for heteroscedasticity (non-constant error variance).

```
#Brown Forsythe Test
#e is a vector of residuals, and x is some vector (usually yhat or one of the x_j's)

brown.forsythe=function(e,x){
  m=median(x)
  e1=e[x<=m]
  e2=e[x>m]
  e1med=median(e1)
  e2med=median(e2)
  d1=abs(e1-e1med)
  d2=abs(e2-e2med)
  d1bar=mean(d1)
  d2bar=mean(d2)
  n1=length(e1)
  n2=length(e2)
  sp=sqrt((sum((d1-d1bar)^2)+sum((d2-d2bar)^2))/(n1+n2-2))
  t=(d1bar-d2bar)/(sp*sqrt(1/n1+1/n2))
  return(2*pt(abs(t),n1+n2-2,lower.tail=FALSE))
}
xtest=1:100
etest=c(1:50,(1:50)*10)
brown.forsythe(etest,xtest) ##output must be 1.610149e-18
```

```
## [1] 1.610149e-18
```

```
brown.forsythe(bin.e,y2)
```

```
## [1] 0.8649346
```

The test says that the bin errors are homoscedastic.

We are not sure if an analysis like this one makes complete sense, but the results do agree with our intuition about how the bin errors should behave.

Cross-Validation

To cross-validate our model, we will evaluate its performance on the test data that it has not seen before. Our performance metric will be the classification accuracy score which is the fraction of the total targets in the test data that are correctly predicted (1 as 1 and 0 as 0).

```
test.hat = predict(my.model,newdata=heart.test,type='response')
test.hat = ifelse(test.hat> 0.5,1,0)
miss= mean(test.hat != heart.test$target)
print(paste('Classification Accuracy:',1-miss))
```

```
## [1] "Classification Accuracy: 0.852459016393443"
```

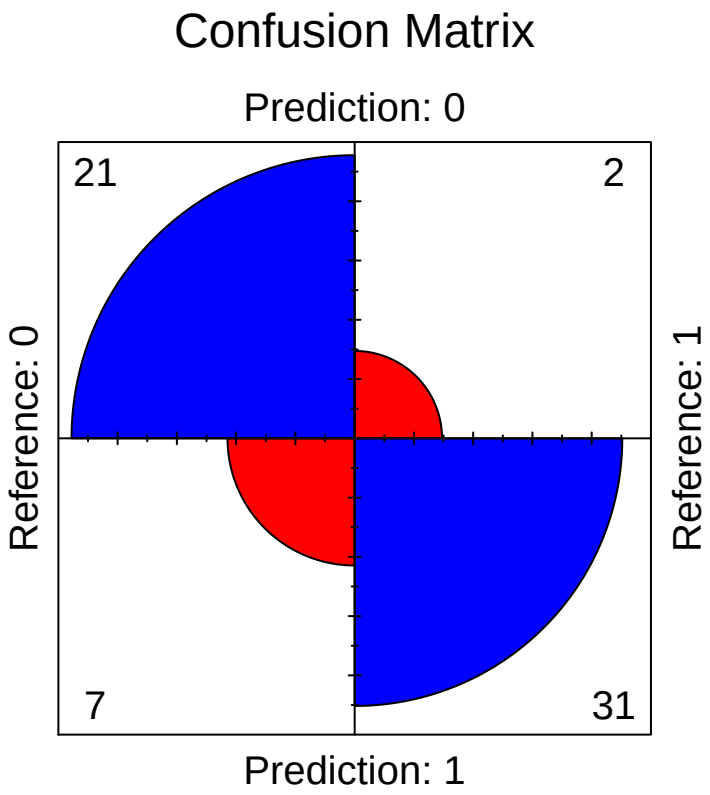
The classification accuracy is around 85.25 %, which is not bad. The model certainly does a decent job in predicting the presence or absense of heart disease given the required variables.

Confusion Matrix

```
library(caret)
```

```
## Loading required package: lattice
```

```
cm = confusionMatrix(factor(test.hat),heart.test$target, positive = '1')
fourfoldplot(cm$table, color = c("red", "blue"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```



The confusion matrix shows that we are doing a pretty good job with classifying the target variable. The fasle positive and false negative rates are quite satisfactory.

ROC-AUC

Since this is a binary classification problem, another performance metric that we can use is the ROC (receiver operating characteristic) curve which helps analyze how the model reponds to changing the threshold probability. It is basically a plot of true positive rate (also called sensitivity or recall) against the false positive rate (1-specificity) for different probability thresholds while classifying the target. The most common threshold is 0.5, but we can lower or raise it depending on how we want to control false positive and/or false negative rates.

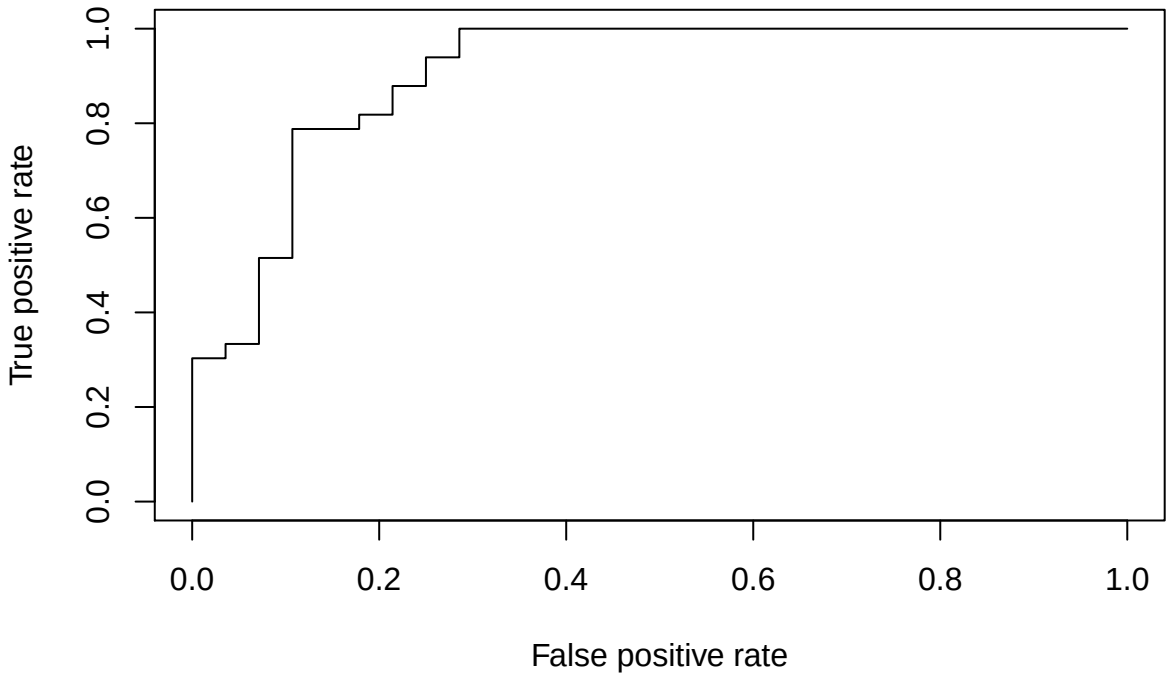
```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
test.hat = predict(my.model,newdata=heart.test,type='response')
pr = prediction(test.hat, heart.test$target)
prf = performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.9058442
```

The AUC (area under the curve) is 0.91 , which is pretty good. While an AUC score of 1 represents perfect predictive capability, AUC score close to 1 in simple terms means the model does a decent job of identifying the true positives and avoiding the false positives.

Conclusion

Looking at our cross-validation results, diagnostic plots, and performance metrics, we can claim that the logistic regression model we have developed is robust and does a pretty reasonable job of predicting heart disease correctly with a classification accuracy of approximately 85.25 %, which means it performs reasonably well. The AUC for the model is 0.91, which is close enough to 1.