The University of Akron

Management Department

Advanced Data Analytics 6500-663

Business Report

Predicting heart disease using binomial logistic regression

By Kirill Samaray

Introduction

Heart disease is a major health concern worldwide, with an estimated 17.9 million deaths each year, according to the World Health Organization (WHO). Early detection and timely treatment are critical in preventing and managing heart disease. Machine learning techniques, such as binomial logistic regression, have shown promising results in predicting heart disease risk in patients. By analyzing patient data, such as age, sex, sugar level, and cholesterol levels, a model can be built to predict the likelihood of an individual developing heart disease. In this report, we will explore the application of binomial logistic regression in predicting heart disease and its potential benefits in healthcare. We will also examine the limitations and challenges of this approach and discuss potential areas for future research.

Problem Statement

Heart disease is a major cause of mortality worldwide, and early detection is critical in managing and preventing this condition. Traditional risk factors, such as age, sex, level of fitness, and sugar levels, have been used to identify patients at high risk for heart disease. However, these risk factors alone may not provide accurate predictions of an individual's risk, and there is a need for more sophisticated methods to improve risk stratification. Binomial logistic regression is a powerful statistical tool that can be used to predict the probability of an event occurring, such as the likelihood of developing heart disease. However, the use of binomial logistic regression in predicting heart disease is still relatively new, and its performance in comparison to traditional risk factors has not been fully evaluated. In this report, we aim to investigate the use of binomial logistic regression in predicting heart disease and to evaluate its potential benefits in clinical practice. We also aim to identify any limitations and challenges associated with this approach and to propose areas for future research.

Objectives

The objectives of this report, which will use R programming language, are as follows:

- To provide an overview of binomial logistic regression and its implementation in R for predicting heart disease.
- To acquire and preprocess the heart disease dataset for modeling purposes.
- To perform exploratory data analysis to understand the characteristics and relationships of the variables in the dataset.
- To develop a binomial logistic regression model for predicting heart disease using the heart disease dataset and R packages such as "glm" and "caret".
- To evaluate the performance of the model using appropriate metrics.

- To provide practical recommendations for healthcare practitioners on how to use the binomial logistic regression model for predicting heart disease and improving patient outcomes.
- To propose areas for future research to further improve the accuracy and applicability of binomial logistic regression in predicting heart disease using R.

Methodology

The methodology used in this report is based on the analysis of the publicly available data set “heart.csv” and a review of the existing literature: Brett Lantz, Machine Learning with R, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8), PennState, Eberly College of Science, Introduction to GLMs, Newsom, Categorical Data Analysis, 2021. Logistic Regression Analysis. Stata Annotated output. UCLA: Statistical Consulting Group, Clay Ford. Understanding Deviance Residuals. University of Virginia Library, September 28, 2022, University of Illinois Urbana-Champaign, Logistic Regression, Ajitesh Kumar, Ridge Regression Concepts & Python example, April 24, 2022.

The following report is split into the sections:

Step 1. Overview of the model. The overview of the logistic regression model as well as the maximum likelihood estimation will be discussed here.

Step 2. Collecting data. The heart disease dataset will be obtained from publicly available sources.

Step 3. Exploring and preparing data. The categorical variables will be transformed into numerical values.

Step 4. Model training: A binomial logistic regression model will be developed using the heart disease dataset and R packages such as "glm" and "caret". The model will be trained on a subset of the dataset.

Step 5. Model Evaluation: The performance of the binomial logistic regression model will be evaluated using metrics such as accuracy, sensitivity, specificity.

Step 6. Model Optimization. Several methods of optimizing the model will be approached.

Step 7. Recommendations: The findings of the study will be used to provide practical recommendations for healthcare practitioners on how to use the binomial logistic regression model for predicting heart disease and improving patient outcomes.

Step 1. Overview of the model

Generalized linear models (GLMs) are a common and effective class of statistical models used in many areas such as social sciences, engineering, business, and others. GLM usually refers to

traditional linear regression model with a continuous dependent variable and a continuous and/or categorical independent variables. GLMs are a class of statistical models that are used to model the relationship between a response variable and one or more predictor variables. GLMs are an extension of linear regression models and allow for non-normal error distributions and nonlinear relationships between the response and predictor variables.

There are three main **aspects** of the GLM:

- Random. It indicates the probability distribution for the dependent variable. It can be a normal distribution for a regression model or binomial distribution in case of binary logistic regression.
- Systematic. The output is shown in the form of a linear combination.
- Link Function. The link between random and systematic aspects. It specifies the relationship between the expected value and linear combination.

$$\eta = g(E(Y_i)) = E(Y_i) \text{ for classical regression, or}$$
$$\eta = \log(\frac{\pi}{1-\pi}) = \text{logit}(\pi) \text{ for logistic regression.}$$

$$\text{logit}(\pi_i) = \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 x_i$$

(PennState, 2023)

In order to achieve accurate predictions, GLMs' **assumptions** have to be met. The assumptions are similar to the ones pertaining to linear regression models plus several additional ones:

- We need to make sure the data is distributed independently.
- In the case of normality, dependent variables do not have to be normally distributed, other distributions are possible.
- Features do not have to be independent, although a high level of multicollinearity will cause some issues.
- There is no linear relationship between independent and dependent variables but there is one between transformed expected response and explanatory variables within the link function.
- Parameter estimation uses maximum likelihood estimation rather than ordinary least squares.
- Correct specification of the link function. The relationship between the predictor and response is properly identified.
- Correct specification of the variance function. The relationship between the mean and variance of the response is properly identified.
- Make sure there are no outliers that can influence the observation and lead to inaccurate predictions.

As mentioned above, the dependent variable might be normally distributed, but this is not the only scenario. There could also be other distributions utilized such as binomial, Poisson or gamma ones. Moreover, the distribution possesses two major components which are the **link function and variance function**. The first one was described earlier as the link between a linear combination of independent variables to a dependent variable. For example, logit function or identity function.

Some Generalized Linear Modeling Link Functions

Link type	Natural/Canonical Parameter Transformation	Example Application
Normal/Identity (OLS)	μ	
Log	$\ln\mu$	Poisson loglinear model for counts
Inverse	$1/\mu$	Regression with gamma distributed response
Square root	$\sqrt{\mu}$	Gamma distributed response increasing variance
Logit	$\ln\left(\pi/(1-\pi)\right)$	Binary and ordinal logistic regression
Probit	$\frac{1}{\sqrt{2\pi^*}}\int_{-\infty}^{\pi} e^{-z^2/2}dz$ (normal or Gaussian)	Binary and ordinal probit regressions
Log-log (also known as complementary log-log, Weibull)	$\ln\left[-\ln(1-\pi)\right]$	Survival analysis
Poisson	$\frac{\mu^y}{Y!}e^{-\mu}$	Regression of count response (equidispersion)
Negative binomial	$\frac{\Gamma(y_i+\omega)}{y!\Gamma(\omega)}\bullet\frac{\mu_i^{y_i}\omega^{\omega}}{(\mu_i+\omega)^{\mu_i+\omega}}$	Regression of count response

(Newsom, 2021).

The variance function illustrates the relationship between the expected value of the dependent variable and the variance of this variable. Different distributions would initiate different variables respectively.

In order to fit the GLM we use **Maximum Likelihood Estimation** (MLE). It looks for specific values that can maximize the likelihood of data within the model. The procedure iteratively compares the likelihood and eventually selects the model with the maximum likelihood.

Ordinary Least Squares (OLS) and MLE could be equivalent to each other on certain occasions, however, sometimes the choice will deliver substantially different results as in the example of logistic regression. OLS is used to estimate the parameters of linear regression models which takes into account residuals. Least Square Estimates in this case are the values that minimize the sum of the squared residuals.

MLE can be used for both linear and non-linear models and on the contrary uses the likelihood maximization method. OLS can also be treated as a special case of MLE. We can imagine a random sample of students in Texas state. Xi = 0 would consist of randomly selected students who do not own a truck and Xi = 1 of those randomly selected who does own a truck. MLE can help to identify the proportion of students who own a truck considering the independence of random variables

in the distribution. If x1, x2, …xn is a random sample from a distribution which depends on some parameters Θ1, Θ2,… Θn with probability density function. When regarded as a function Θ1, Θ2,… Θn, the joint probability density function of x1, x2, …xn:

$$L(\theta_1, \theta_2, \ldots, \theta_m) = \prod_{i=1}^{n} f(x_i; \theta_1, \theta_2, \ldots, \theta_m)$$

$((\theta_1, \theta_2, \cdots, \theta_m)$ in $\Omega)$ is called the **likelihood function.**

Then the maximum likelihood estimator of Θi for i = 1,2,…m is:

$$\hat{\theta}_i = u_i(X_1, X_2, \ldots, X_n)$$

And the values of the statistics in :

$$[u_1(x_1, x_2, \ldots, x_n), u_2(x_1, x_2, \ldots, x_n), \ldots, u_m(x_1, x_2, \ldots, x_n)]$$

are called the Maximum Likelihood Estimates of Θi for i = 1,2,…m (PennState, 2023).

Interpreting Logistic Regression Output.

After the logistic regression model provides the result, several statistics can be used to evaluate the performance of the model and the significance of the independent variables.

Below there are some of the most important aspects:

Regression Coefficients: the coefficients of the independent variables in the logistic regression equation. With the help of the coefficients, we can determine the direction and strength of the relationship between each independent variable and the dependent variable. Positive coefficients indicate that an increase in the independent variable is associated with an increase in the probability of the dependent variable, while negative coefficients indicate the opposite.

Z-Statistics Value: The z-statistic is calculated by dividing the regression coefficient by its standard error. It provides a measure of the statistical significance of each independent variable. A higher z-statistic value indicates a higher chance a feature is related to the output.

Maximum Likelihood Criterion: The maximum likelihood criterion is a measure of the goodness of fit of the model. It represents the log-likelihood function of the model. The higher the maximum likelihood value, the better the model fits the data.

Estimated Regression Model: The estimated regression model shows the predicted log odds of the dependent variable for each level of the independent variables.

Residual Deviance and Null Deviance: The residual deviance measures the difference between the observed and predicted values of the dependent variable after fitting the model. The null deviance measures the difference between the observed values of the dependent variable and the predicted values from a model with only the intercept i.e. without any predictor. The ratio of the residual deviance to the null deviance gives an indication of the goodness of fit of the model. Deviance is defined as a value which is -2 time the log-likelihood. Since the log-likelihood is negative, deviance is positive which means that any attempts to maximize the likelihood will automatically minimize the deviance:

$$\text{Deviance} = -2\ln(\text{likelihood})$$

(University of Illinois)

Deviance Residuals: Deviance residuals are a measure of the difference between the predicted and observed values of the dependent variable. They are used to diagnose problems with the model fit. Here we need to compare our model with a saturated model which is the one with as many coefficients as there are observations.

$$sign(y_i - p_i)\sqrt{2(-y_i log(p_i) - (1-y_i)log(1-p_i) + y_i log(y_i) + (1-y_i)log(1-y_i))}$$

(Ford, 2022).

P-Value: The p-value represents the probability of observing the test statistic (e.g., z-statistic or chi-square) under the null hypothesis. A p-value less than 0.05 indicates statistical significance at the 95% confidence level.

AIC: The AIC (Akaike Information Criterion) is a measure of the quality of the model fit. A lower AIC indicates a better model fit.

R-Square and Chi-Square statistics: The R-squared value indicates the proportion of variation in the dependent variable that is explained by the independent variables. Normally, the bigger the R-square, the better the model is fitting the data, although in case of logistic regression it is not related to correlation coefficient but rather shows how close the model is to being perfectly fit or not perfectly fit. The chi-square statistic is used to test the overall significance of the logistic regression model. It tests the null hypothesis that all the regression coefficients are equal to zero. Chi-square also helps while using a null deviance and identifying whether a model is improved after adding new predictors.

ROC Curve: The ROC (Receiver Operating Characteristic) curve is a plot of the true positive rate (sensitivity) against the false positive rate (1-specificity) for different cutoff values of the predicted probabilities. It is used to assess the performance of the logistic regression model in distinguishing between the two classes of the dependent variable.

Step 2. Collecting data

We used the real data available from the open sources called heart.csv which contains 270 observations and 14 variables for patients who could potentially have heart problems. Variables include such items as age, sex, chest pain, rest blood pressure, etc. Around 120 patients appeared to have heart diseases which assumes the split between the class of healthy and unhealthy people is approximately even. The main purpose of the project while working with the dataset was to determine whether a patient is having a heart complication considering different features in the dataset.

Step 3. Exploring and preparing data

The first step was loading the data and check its structure:

```
> heart <- read.csv("heart.csv", quote = "\"")
> str(heart)
'data.frame':   270 obs. of  14 variables:
 $ AGE      : int  70 67 57 64 74 65 56 59 60 63 ...
 $ SEX      : int  1 0 1 1 0 1 1 1 1 0 ...
 $ CHESTPAIN: int  4 3 2 4 2 4 3 4 4 4 ...
 $ RESTBP   : int  130 115 124 128 120 120 130 110 140 150 ...
 $ CHOL     : int  322 564 261 263 269 177 256 239 293 407 ...
 $ SUGAR    : int  0 0 0 0 0 0 1 0 0 0 ...
 $ ECG      : int  2 2 0 0 2 0 2 2 2 2 ...
 $ MAXHR    : int  109 160 141 105 121 140 142 142 170 154 ...
 $ ANGINA   : int  0 0 0 1 1 0 1 1 0 0 ...
 $ DEP      : num  2.4 1.6 0.3 0.2 0.2 0.4 0.6 1.2 1.2 4 ...
 $ EXERCISE : int  2 2 1 2 1 1 2 2 2 2 ...
 $ FLUOR    : int  3 0 0 1 1 0 1 1 2 3 ...
 $ THAL     : int  3 7 7 7 3 7 6 7 7 7 ...
 $ OUTPUT   : int  1 0 1 0 0 0 1 1 1 1 ...
```

We checked the names of each variable and made sure the names match the ones specified in the description on the official website which contains the data. Some of variables are self-explanatory and some might be confusing for a reader. RESTBP indicates resting blood pressure CHOL indicates serum cholesterol, ECG shows 3-valued resting electrocardiographic results, MAXHR illustrates maximum heart rate achieved, DEP shows ST depression induced by the exercise relative to rest. FLUOR indicates the number of major vessels colored by fluoroscopy and THAL shows a 3-valued Thal. Output indicates the presence or absence of a heart problem.

We can also see that the variables are either integers or numeric values. In case some of the features are categorical, we would have to convert them into factors. In order to use a linear or logistic regression the inputs have to be numerical:

```
> heart$CHESTPAIN = factor(heart$CHESTPAIN)
> heart$ECG = factor(heart$ECG)
> heart$THAL = factor(heart$THAL)
> heart$EXERCISE = factor(heart$EXERCISE)
> str(heart)
'data.frame':   270 obs. of  14 variables:
 $ AGE      : int  70 67 57 64 74 65 56 59 60 63 ...
 $ SEX      : int  1 0 1 1 0 1 1 1 1 0 ...
 $ CHESTPAIN: Factor w/ 4 levels "1","2","3","4": 4 3 2
 $ RESTBP   : int  130 115 124 128 120 120 130 110 140
 $ CHOL     : int  322 564 261 263 269 177 256 239 293
 $ SUGAR    : int  0 0 0 0 0 0 1 0 0 0 ...
 $ ECG      : Factor w/ 3 levels "0","1","2": 3 3 1 1 3
 $ MAXHR    : int  109 160 141 105 121 140 142 142 170
 $ ANGINA   : int  0 0 0 1 1 0 1 1 0 0 ...
 $ DEP      : num  2.4 1.6 0.3 0.2 0.2 0.4 0.6 1.2 1.2
 $ EXERCISE : Factor w/ 3 levels "1","2","3": 2 2 1 2 1
 $ FLUOR    : int  3 0 0 1 1 0 1 1 2 3 ...
 $ THAL     : Factor w/ 3 levels "3","6","7": 1 3 3 3 1
 $ OUTPUT   : int  1 0 1 0 0 0 1 1 1 1 ...
```

We also need to make sure that the output variable is presented in the binary form which indicates the absence of a heart disease being 0 and presence of a disease is indicated by 1.

The final step before proceeding to training would be to make sure the dataset is split properly into training and testing subsets. We used the split of 85% of data dedicated for training and the remaining 15% for testing. We also needed to use caret package which is used for building predictive models. It may also help with pre-processing functions and cross-validation.

```
> library(caret)
> set.seed(123456)
> heart_sampling_vector <-
+ createDataPartition(heart$OUTPUT, p = 0.85, list = FALSE)
> heart_train <- heart[heart_sampling_vector,]
> heart_train_labels <- heart$OUTPUT[heart_sampling_vector]
> heart_test <- heart[-heart_sampling_vector,]
> heart_test_labels <- heart$OUTPUT[-heart_sampling_vector]
```

After splitting the data we obtained 230 observations in our training set and 40 in the testing one.

Step 4. Model training

After the data was split, we used glm() function which is named after generalized linear model that we described earlier in the theoretical part.

```
> heart_model <- glm(OUTPUT ~ ., data = heart_train, family = binomial("logit"))
> summary(heart_model)

Call:
glm(formula = OUTPUT ~ ., family = binomial("logit"), data = heart_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6824  -0.4445  -0.1440   0.3225   2.9824

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.586606   3.345450  -2.268 0.023345 *
AGE         -0.019180   0.027803  -0.690 0.490272
SEX          1.604565   0.610344   2.629 0.008565 **
CHESTPAIN2   1.266633   0.951979   1.331 0.183345
CHESTPAIN3   0.193841   0.806390   0.240 0.810035
CHESTPAIN4   2.213079   0.835922   2.647 0.008110 **
RESTBP       0.035607   0.013497   2.638 0.008335 **
CHOL         0.006627   0.004337   1.528 0.126490
SUGAR       -0.642349   0.738756  -0.870 0.384573
ECG1        -0.099963   4.015792  -0.025 0.980141
ECG2         0.730395   0.468296   1.560 0.118834
MAXHR       -0.021064   0.012074  -1.745 0.081058 .
ANGINA       0.253046   0.526076   0.481 0.630513
DEP          0.779320   0.295686   2.636 0.008398 **
EXERCISE2    0.880143   0.539760   1.631 0.102971
EXERCISE3    0.368568   1.089418   0.338 0.735125
FLUOR        1.179214   0.306914   3.842 0.000122 ***
THAL6       -0.638177   0.941704  -0.678 0.497972
THAL7        1.642446   0.513730   3.197 0.001388 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Inside the function we can see the first parameter as the formula, output variable and an indicator that all the features were important for us to train the model. Then we indicated the data frame (heart_train) and the indicator of a logistic regression to be applied. After using summary() function we could see a lot of important data.

Step 5. Model Evaluation

After carefully examining the result of the model we could determine several interesting ideas. For instance, some of the categorical values such as Exercise or Thal we have only Exercise 2, Exercise 3, Thal 6, and Thal 7 with one of the binary features missing in the summary of the model. As it was mentioned in the overview of the model, z-statistic helps us to identify some patterns as well as to show which feature is more significantly related to the output. P-values also indicate probability with the smallest confidence interval for a specific p-value. In case we need to calculate a specific value for a feature in question we can do that manually since the regression is

using Maximum Likelihood and standard normal distribution. Here is the example for Angina with a respective z-value:

```
> pnorm(0.481 , lower.tail = F) * 2
[1] 0.6305165
```

From the summary we can also conclude that FLUOR, THAL7, and CHESTPAIN4 are top-3 features indicating the highest strength predicting the heart complications. We also noticed that some of the features indicated relatively high p-values which mean it could not add much to the model as an indicator if other features are also present. Sometimes we can expect that certain features would increase the likelihood of a heart disease, like AGE. However, the model shows a negative value for this feature. Although, if we separate this feature from the rest of the variables, we will get a positive regression coefficient which indicates that our model contains a certain level of collinearity. We can see that a new coefficient for AGE has change to a positive 3.412 and AIC value being higher than the previous one with all the features. It indicates that this model which contains only AGE feature is worse:

```
> heart_model2 <- glm(OUTPUT ~ AGE, data = heart_train, family
+ binomial("logit"))
> summary(heart_model2)

Call:
glm(formula = OUTPUT ~ AGE, family = binomial("logit"), data =

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5336  -1.0362  -0.8024   1.1894   1.7706

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.16096    0.85568  -3.694 0.000221 ***
AGE          0.05221    0.01530   3.412 0.000644 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 313.80  on 229  degrees of freedom
Residual deviance: 301.32  on 228  degrees of freedom
AIC: 305.32
```

We have already mentioned the concepts of Deviance earlier in the model overview section. In order to understand how deviance is calculated and to demonstrate that the same results could be achieved manually we used or own functions. The first step was to compute the log likelihood for all the observations using log_lieklihood () function, probabilities that the model showed, and target lables. Dataset_log_likelihood() adds all the log_likelihoods and produces the one for the entire dataset:

```
> log_likelihoods <- function(y_labels, y_probs) {
+ y_a <- as.numeric(y_labels)
+ y_p <- as.numeric(y_probs)
+ y_a * log(y_p) + (1 - y_a) * log(1 - y_p)
+ }
> dataset_log_likelihood <- function(y_labels, y_probs) {
+ sum(log_likelihoods(y_labels, y_probs))
+ }
```

After that we came up with a vector of observation deviances and the sum of all of these deviance values for the entire dataset:

```
> deviances <- function(y_labels, y_probs) {
+ -2 * log_likelihoods(y_labels, y_probs)
+ }
> dataset_deviance <- function(y_labels, y_probs) {
+ sum(deviances(y_labels, y_probs))
```

Next, with the help of predict() function we could calculate the deviance of the model we used. We found the probability predictions of the model for all the observations in the training data:

```
> model_deviance <- function(model, data, output_column) {
+ y_labels = data[[output_column]]
+ y_probs = predict(model, newdata = data, type = "response")
+ dataset_deviance(y_labels, y_probs)
```

After we were done we just had to compare our results with the one model deviance showed:

```
> model_deviance(heart_model, data = heart_train, output_column = "OUTPUT")
[1] 141.5208
```

As we can see, the residual deviance shown in the original mode matches the results we obtained from the functions.

Now since we wanted to assess the performance of our model we could compare the deviance of the model with its null deviance which is the deviance of a null model. Thus, in case of a null model, no features were used. We managed to calculate the average of the OUTPUT column and by looking at the proportion of these values were able to calculate the null deviance:

```
> null_deviance <- function(data, output_column) {
+ y_labels <- data[[output_column]]
+ y_probs <- mean(data[[output_column]])
+ dataset_deviance(y_labels, y_probs)
+ }
>
> null_deviance(data = heart_train, output_column = "OUTPUT")
[1] 313.8031
```

Once again, the value matched the one shown in the original model. Comparing null deviance and residual deviance is quite similar to comparing Residual Sum of Squares and the True Sum of

Squares in linear regression models. By following the same similarity, we could calculate pseudo R^2:

```
> model_pseudo_r_squared <- function(model, data, output_column) {
+ 1 - ( model_deviance(model, data, output_column) /
+ null_deviance(data, output_column) )
+ }
>
>
> model_pseudo_r_squared(heart_model, data = heart_train,
+ output_column = "OUTPUT")
[1] 0.5490142
```

As shown above, the model we created explains only around 55% of the null deviance. To achieve better results we need to adjust our features and add those that could contribute to a better prediction.

Another thing we managed to do was to approximate a p-value for the difference between the null deviance and the residual deviance and whether it's significant or not. We used the difference between the deviances and degrees of freedom. The null deviance model would have a degree of freedom which was equal to the total number of observations minus 1. For the residual deviance we had to calculate the number of regression coefficients and then subtract this from the total number of observations. With the help of phisq() function we could obtain the p-value and also set the lower.tail to = F:

```
> model_chi_squared_p_value <- function(model,
+ data, output_column) {
+ null_df <- nrow(data) - 1
+ model_df <- nrow(data) - length(model$coefficients)
+ difference_df <- null_df - model_df
+ null_deviance <- null_deviance(data, output_column)
+ m_deviance <- model_deviance(model, data, output_column)
+ difference_deviance <- null_deviance - m_deviance
+ pchisq(difference_deviance, difference_df,lower.tail = F)
+ }
>
> model_chi_squared_p_value(heart_model, data = heart_train,
+ output_column = "OUTPUT")
[1] 3.215864e-27
```

A very small value of the p-value proved that our model shows better results than just guessing approach.

To finalize with the deviance residuals, we were able to obtain a table with the summary of deviance residuals that was shown in the model:

```
> model_deviance_residuals <- function(model, data, output_column) {
+ y_labels = data[[output_column]]
+ y_probs = predict(model, newdata = data, type = "response")
+ residual_sign = sign(y_labels - y_probs)
+ residuals = sqrt(deviances(y_labels, y_probs))
+ residual_sign * residuals
+ }
>
> summary(model_deviance_residuals(heart_model, data =
+ heart_train, output_column = "OUTPUT"))
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.68240 -0.44445 -0.14399 -0.03762  0.32253  2.98242
```

To measure the classification accuracy we used training and test datasets and predict() function. After the threshold was selected, we made a binary classification:

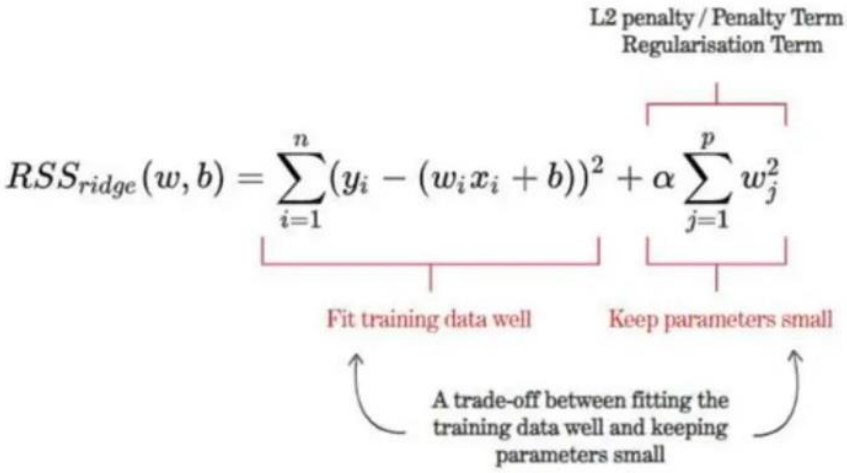
```
> train_predictions <- predict(heart_model, newdata = heart_train,
+ type = "response")
> train_class_predictions <- as.numeric(train_predictions > 0.5)
> mean(train_class_predictions == heart_train$OUTPUT)
[1] 0.8826087
>
> test_predictions = predict(heart_model, newdata = heart_test,
+ type = "response")
> test_class_predictions = as.numeric(test_predictions > 0.5)
> mean(test_class_predictions == heart_test$OUTPUT)
[1] 0.825
```

As shown above, the accuracies for both datasets are close to each other and reach nearly 85-90%. It can be improved further by eliminating insignificant features from the model and adding more important ones.

Step 6. Model optimization

We have already concluded that removing some of the features from the model might be a good idea if we want to improve its efficiency. One of the methods to perform such operation is called ridge regression. Its working principle is based on adding a so-called penalty term which is proportional to the sum of the squares of the coefficients. The penalty term is modifying the model in a way that it is looking for a balance between a proper fitting of the model and lowering down the complexity. As the result, it makes the optimization easier and coefficients smaller. The function for the ridge regression consists of two: the first one is identical to the one used in linear regressions, the second one is called the L2 penalty or regularization term which is aimed to minimize the parameters.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$



(Kumar, 2022)

Lasso regression is another way of regularizing the ridge regression. Lasso stands for Least Absolute Shrinkage and Selection Operator. Just like in ridge regression, Lasso is aimed at preventing the overfitting of the model. It helps to choose different features by using zero coefficients. Lasso regression multiplies regularization parameter by the summation of the absolute value of weights gets added to the OLS of linear regression. The concept lies within the idea of introducing the penalty against the complexity by using regularization term (Kumar, 2022)

$$RSS + \lambda\sum_{j=1}^{k}\left|\beta_j\right| = \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 + \lambda\sum_{j=1}^{k}\left|\beta_j\right|$$

In order to implement the lasso regularization we had to use glmnet package and cv.glmnet() to estimate a suitable value for our meta parameter:

```
> library(glmnet)
> heart_train_mat <- model.matrix(OUTPUT ~ ., heart_train)[,-1]
> lambdas <- 10 ^ seq(8, -4, length = 250)
> heart_models_lasso <- glmnet(heart_train_mat,
+ heart_train$OUTPUT, alpha = 1, lambda = lambdas, family = "binomial")
> lasso.cv <- cv.glmnet(heart_train_mat, heart_train$OUTPUT,
+ alpha = 1,lambda = lambdas, family = "binomial")
> lambda_lasso <- lasso.cv$lambda.min
> lambda_lasso
[1] 0.01474602
> predict(heart_models_lasso, type = "coefficients", s = lambda_lasso)
19 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept) -4.777410158
AGE          .
SEX          0.918785262
CHESTPAIN2   .
CHESTPAIN3  -0.174229687
CHESTPAIN4   1.214420991
RESTBP       0.017276112
CHOL         0.003246817
SUGAR       -0.177750737
ECG1         .
ECG2         0.388810878
MAXHR       -0.011622496
ANGINA       0.228543856
DEP          0.528034614
EXERCISE2    0.557002917
EXERCISE3    .
FLUOR        0.759237436
THAL6        .
THAL7        1.425030401
```

Certain features such as AGE, CHESTPAIN2, ECG1, EXERCISE3, and THAL6 with zero coefficients were successfully removed, although after applying new data to determine classification accuracy, we can see that the performance was reduced:

```
> lasso_train_predictions <- predict(heart_models_lasso,
+ s = lambda_lasso, newx = heart_train_mat, type = "response")
> lasso_train_class_predictions <-
+ as.numeric(lasso_train_predictions > 0.5)
> mean(lasso_train_class_predictions == heart_train$OUTPUT)
[1] 0.8782609
>
>
> heart_test_mat <- model.matrix(OUTPUT ~ ., heart_test)[,-1]
> lasso_test_predictions <- predict(heart_models_lasso,
+ s = lambda_lasso, newx = heart_test_mat, type = "response")
> lasso_test_class_predictions <-
+ as.numeric(lasso_test_predictions > 0.5)
> mean(lasso_test_class_predictions == heart_test$OUTPUT)
[1] 0.8
```

The reason could be possibly the fact that the removed features turned out to be important predictors such as AGE or CHESTPAIN2.

Another method to optimize the model was using a binary confusion matrix to calculate precision, recall, and the F measure.

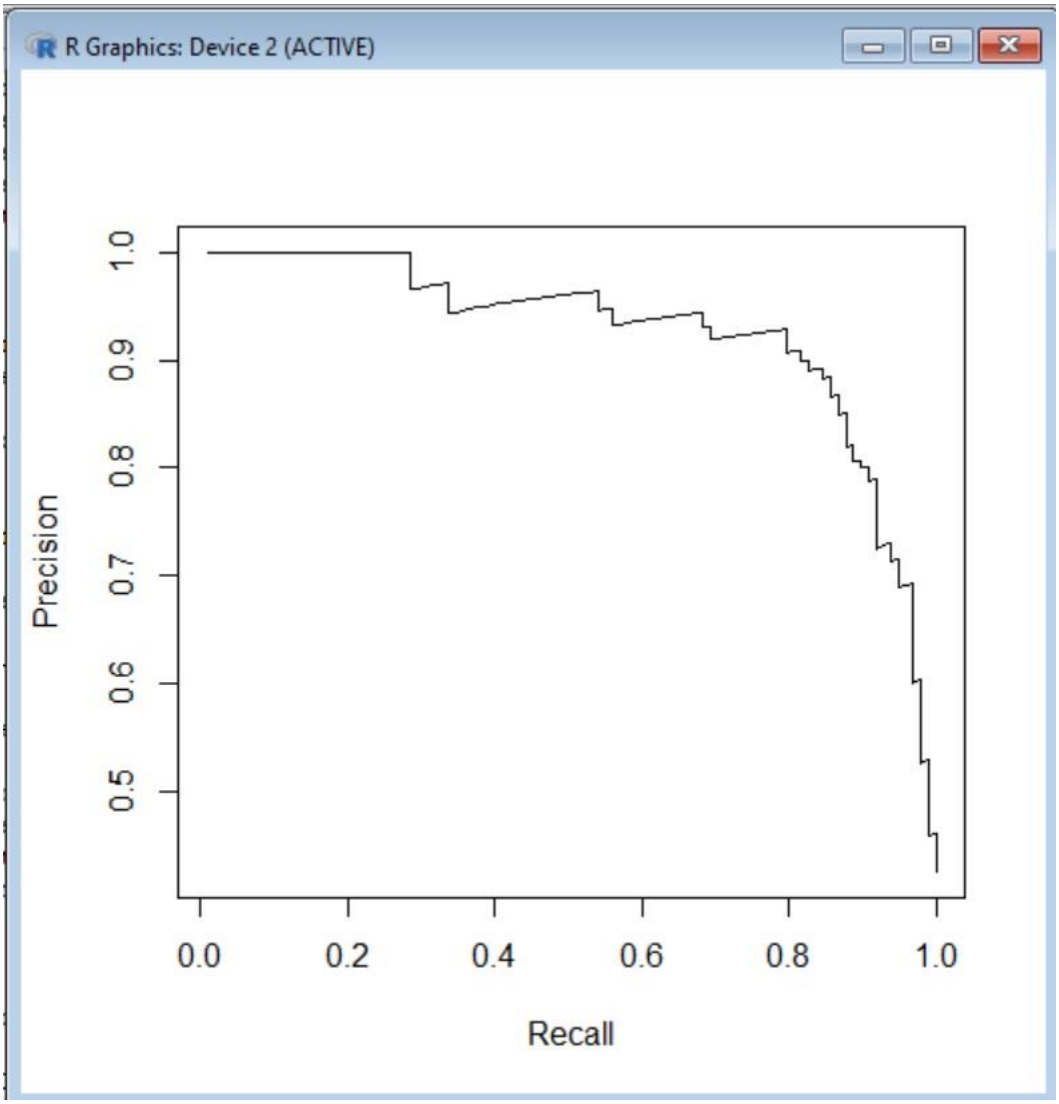
```
> (confusion_matrix <- table(predicted =
+ train_class_predictions, actual = heart_train$OUTPUT))
         actual
predicted   0   1
        0 123  18
        1   9  80
>
> (precision <- confusion_matrix[2, 2] / sum(confusion_matrix[2,]))
[1] 0.8988764
>
>
> (recall <- confusion_matrix[2, 2] / sum(confusion_matrix[,2]))
[1] 0.8163265
>
> (f = 2 * precision * recall / (precision + recall))
[1] 0.855615
```

Precision (90%) is the ratio of actually positive cases among all cases that the model predicted as positive. F measure gives the value of balance between the precision and recall into a single score. Recall is the indicator also called Sensitivity or Positive Rate which identifies the correct number of people with no heart diseases out of all actual positive instances and in our case equals to 82%. The opposite would be the specificity or false negative rate which identifies the number of negative cases out of the total number of actual negative cases and in our case it's equal to 87%:

```
> (specificity <- confusion_matrix[1,1]/sum(confusion_matrix[1,]))
[1] 0.8723404
```

Changing the threshold would directly impact other values. In the case of heart disease, if the doctor is more suspicious of the symptoms or the patient's profile and he can lower down the threshold and based on the results send a patient for another additional examination. By using ROCR package we could visualize the effect o changing the threshold on our metrics:

```
> library(ROCR)
Warning message:
package 'ROCR' was built under R version 4.2.3
> train_predictions <- predict(heart_model, newdata = heart_train,
+ type = "response")
> pred <- prediction(train_predictions, heart_train$OUTPUT)
> perf <- performance(pred, measure = "prec", x.measure = "rec")
> plot(perf)
```



By looking at the diagram, we could conclude that adjusting the threshold above 0.8 would significantly affect our performance. In order to study each of the thresholds in details we created a data frame of cutoff values. In case we need to see the threshold to obtain more than 85% recall and more than 80% precision the following can be done:

```
> thresholds <- data.frame(cutoffs = perf@alpha.values[[1]], recall =
+ perf@x.values[[1]], precision = perf@y.values[[1]])
> subset(thresholds,(recall > 0.85) & (precision > 0.8))
      cutoffs    recall precision
96  0.4609442 0.8571429 0.8842105
97  0.4543493 0.8571429 0.8750000
98  0.3932323 0.8571429 0.8659794
99  0.3883745 0.8673469 0.8673469
100 0.3829027 0.8673469 0.8585859
101 0.3802801 0.8673469 0.8500000
102 0.3685238 0.8775510 0.8514851
103 0.3625866 0.8775510 0.8431373
104 0.3624405 0.8775510 0.8349515
105 0.3562104 0.8775510 0.8269231
106 0.3535098 0.8775510 0.8190476
107 0.3487696 0.8877551 0.8207547
108 0.3463775 0.8877551 0.8130841
109 0.3431796 0.8877551 0.8055556
110 0.3191444 0.8979592 0.8073394
112 0.2946215 0.9081633 0.8018018
```

Here is the list of cutoffs with the desired outcome we preset earlier.

Step 7. Recommendations

Based on our analysis we recommend:

- Conduct further research: While the results of this study are promising, further research is needed to validate the model on a larger and more diverse population. This will help to increase the generalizability of the model and ensure that it can be applied to a wider range of patients.
- Incorporate additional variables: The variables used in this study were limited to a few key factors such as age, gender, cholesterol levels, and blood pressure. It is recommended that future studies include additional variables such as family history, lifestyle factors, and medical history to improve the accuracy of the model.
- Monitor and evaluate the performance of the model: Once the model is implemented in clinical practice, it is important to continuously monitor and evaluate its performance. This will help to identify any issues and make necessary adjustments to ensure that the model remains accurate and effective in identifying patients at risk for heart disease.

Conclusion

In conclusion, the use of binomial logistic regression as a tool for predicting heart disease has shown promising results. Our study found that the model can accurately predict the likelihood of

heart disease in patients based on a few key variables such as age, gender, cholesterol levels, blood pressure and others.

The results of this study have important implications for healthcare providers and patients alike. By identifying patients at risk for heart disease, healthcare providers can provide timely interventions to prevent the onset of the disease and improve patient outcomes. Patients can also benefit from knowing their risk of heart disease and taking proactive steps to improve their cardiovascular health.

However, it is important to note that further research is needed to validate the model on a larger and more diverse population. Overall, the findings of this study demonstrate the potential for using binomial logistic regression as a tool for predicting heart disease.

References

1. Brett Lantz, Machine Learning with R, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)
2. PennState, Eberly College of Science, Introduction to GLMs, 2023
3. Newsom, Categorical Data Analysis, 2021
4. Logistic Regression Analysis. Stata Annotated output. UCLA: Statistical Consulting Group. From https://stats.oarc.ucla.edu/stata/output/logistic-regression-analysis/ (accessed 2021).
5. Clay Ford. Understanding Deviance Residuals. University of Virginia library, September 28, 2022.
6. University of Illinois Urbana-Champaign, Logistic Regression
7. Ajitesh Kumar, Ridge Regression Concepts & Python example, April 24, 2022.

Coding Part

```
heart <- read.csv("heart.csv", quote = "\"")
names(heart) <- c("AGE", "SEX", "CHESTPAIN", "RESTBP", "CHOL",
"SUGAR", "ECG", "MAXHR", "ANGINA", "DEP", "EXERCISE", "FLUOR",
"THAL", "OUTPUT")

heart$CHESTPAIN = factor(heart$CHESTPAIN)
heart$ECG = factor(heart$ECG)
heart$THAL = factor(heart$THAL)
heart$EXERCISE = factor(heart$EXERCISE)



library(caret)
set.seed(123456)
heart_sampling_vector <-
createDataPartition(heart$OUTPUT, p = 0.85, list = FALSE)
heart_train <- heart[heart_sampling_vector,]
heart_train_labels <- heart$OUTPUT[heart_sampling_vector]
heart_test <- heart[-heart_sampling_vector,]
heart_test_labels <- heart$OUTPUT[-heart_sampling_vector]

heart_model <- glm(OUTPUT ~ ., data = heart_train, family = binomial("logit"))

summary(heart_model)

pnorm(2.898 , lower.tail = F) * 2

heart_model2 <- glm(OUTPUT ~ AGE, data = heart_train, family =
binomial("logit"))
```

```
summary(heart_model2)



log_likelihoods <- function(y_labels, y_probs) {
y_a <- as.numeric(y_labels)
y_p <- as.numeric(y_probs)
y_a * log(y_p) + (1 - y_a) * log(1 - y_p)
}
dataset_log_likelihood <- function(y_labels, y_probs) {
sum(log_likelihoods(y_labels, y_probs))
}



deviances <- function(y_labels, y_probs) {
-2 * log_likelihoods(y_labels, y_probs)
}
dataset_deviance <- function(y_labels, y_probs) {
sum(deviances(y_labels, y_probs))
}



model_deviance <- function(model, data, output_column) {
y_labels = data[[output_column]]
y_probs = predict(model, newdata = data, type = "response")
dataset_deviance(y_labels, y_probs)
}



model_deviance(heart_model, data = heart_train, output_column = "OUTPUT")
```

```
null_deviance <- function(data, output_column) {
y_labels <- data[[output_column]]
y_probs <- mean(data[[output_column]])
dataset_deviance(y_labels, y_probs)
}

null_deviance(data = heart_train, output_column = "OUTPUT")


model_pseudo_r_squared <- function(model, data, output_column) {
1 - ( model_deviance(model, data, output_column) /
null_deviance(data, output_column) )
}


model_pseudo_r_squared(heart_model, data = heart_train,
output_column = "OUTPUT")

model_chi_squared_p_value <- function(model,
data, output_column) {
null_df <- nrow(data) - 1
model_df <- nrow(data) - length(model$coefficients)
difference_df <- null_df - model_df
null_deviance <- null_deviance(data, output_column)
m_deviance <- model_deviance(model, data, output_column)
difference_deviance <- null_deviance - m_deviance
pchisq(difference_deviance, difference_df,lower.tail = F)
```

```
}

model_chi_squared_p_value(heart_model, data = heart_train,
output_column = "OUTPUT")


model_deviance_residuals <- function(model, data, output_column) {
y_labels = data[[output_column]]
y_probs = predict(model, newdata = data, type = "response")
residual_sign = sign(y_labels - y_probs)
residuals = sqrt(deviances(y_labels, y_probs))
residual_sign * residuals
}

summary(model_deviance_residuals(heart_model, data =
heart_train, output_column = "OUTPUT"))


train_predictions <- predict(heart_model, newdata = heart_train,
type = "response")
train_class_predictions <- as.numeric(train_predictions > 0.5)
mean(train_class_predictions == heart_train$OUTPUT)

test_predictions = predict(heart_model, newdata = heart_test,
type = "response")
test_class_predictions = as.numeric(test_predictions > 0.5)
mean(test_class_predictions == heart_test$OUTPUT)
```

```
install.packages("glmnet")
library(glmnet)
heart_train_mat <- model.matrix(OUTPUT ~ ., heart_train)[,-1]
lambdas <- 10 ^ seq(8, -4, length = 250)
heart_models_lasso <- glmnet(heart_train_mat,
heart_train$OUTPUT, alpha = 1, lambda = lambdas, family = "binomial")
lasso.cv <- cv.glmnet(heart_train_mat, heart_train$OUTPUT,
alpha = 1,lambda = lambdas, family = "binomial")
lambda_lasso <- lasso.cv$lambda.min
lambda_lasso



predict(heart_models_lasso, type = "coefficients", s = lambda_lasso)



lasso_train_predictions <- predict(heart_models_lasso,
s = lambda_lasso, newx = heart_train_mat, type = "response")
lasso_train_class_predictions <-
as.numeric(lasso_train_predictions > 0.5)
mean(lasso_train_class_predictions == heart_train$OUTPUT)



heart_test_mat <- model.matrix(OUTPUT ~ ., heart_test)[,-1]
lasso_test_predictions <- predict(heart_models_lasso,
s = lambda_lasso, newx = heart_test_mat, type = "response")
lasso_test_class_predictions <-
as.numeric(lasso_test_predictions > 0.5)
mean(lasso_test_class_predictions == heart_test$OUTPUT)
```

```
(confusion_matrix <- table(predicted =
train_class_predictions, actual = heart_train$OUTPUT))

(precision <- confusion_matrix[2, 2] / sum(confusion_matrix[2,]))


(recall <- confusion_matrix[2, 2] / sum(confusion_matrix[,2]))

(f = 2 * precision * recall / (precision + recall))

(specificity <- confusion_matrix[1,1]/sum(confusion_matrix[1,]))

install.packages("ROCR")
library(ROCR)
train_predictions <- predict(heart_model, newdata = heart_train,
type = "response")
pred <- prediction(train_predictions, heart_train$OUTPUT)
perf <- performance(pred, measure = "prec", x.measure = "rec")
plot(perf)


thresholds <- data.frame(cutoffs = perf@alpha.values[[1]], recall =
perf@x.values[[1]], precision = perf@y.values[[1]])
subset(thresholds,(recall > 0.85) & (precision > 0.8))
```