Predicting the risk of heart disease:

The 'heartdisease' dataset is a collection of patient data related to heart disease risk factors. It typically consists of 918 observations, each corresponding to a different patient. The dataset includes both numerical and categorical variables, with the target variable indicating whether each patient is at risk of heart disease or not.

Variables

Age: The age of the patient.

Sex: The gender of the patient (M or F).

RestingBP: The resting blood pressure of the patient (in mm Hg).

Cholesterol: The cholesterol level of the patient (in mg/dL).

FastingBS: Fasting blood sugar level (> 120 mg/dL or <= 120 mg/dL).

RestingECG: Resting electrocardiographic results (normal, having ST-T wave abnormality, or left ventricular hypertrophy).

MaxHR: Maximum heart rate achieved by the patient. Angina: Exercise-induced angina (yes or no).

HeartPeakReading: A variable representing a measurement or reading related to the peak activity or performance of the heart during a specific event or condition.

HeartDisease: A binary categorical variable indicating the presence (1) or absence (0) of heart disease in patients.

Installing packeges and calling the data:

```
install.packages("MLDataR")
```

```
Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

```
library(MLDataR)
library(caret)
```

```
Loading required package: ggplot2
```

```
Loading required package: lattice
```

```
data("heartdisease")
```

Check the structure of the data:

```
str(heartdisease)
```

```
tibble [918 x 10] (S3: tbl_df/tbl/data.frame)
 $ Age             : num [1:918] 40 49 37 48 54 39 45 54 37 48 ...
 $ Sex             : chr [1:918] "M" "F" "M" "F" ...
 $ RestingBP       : num [1:918] 140 160 130 138 150 120 130 110 140 120 ...
 $ Cholesterol     : num [1:918] 289 180 283 214 195 339 237 208 207 284 ...
 $ FastingBS       : num [1:918] 0 0 0 0 0 0 0 0 0 0 ...
 $ RestingECG      : chr [1:918] "Normal" "Normal" "ST" "Normal" ...
 $ MaxHR           : num [1:918] 172 156 98 108 122 170 170 142 130 120 ...
 $ Angina          : chr [1:918] "N" "N" "N" "Y" ...
 $ HeartPeakReading: num [1:918] 0 1 0 1.5 0 0 0 0 1.5 0 ...
 $ HeartDisease    : num [1:918] 0 1 0 1 0 0 0 0 1 0 ...
```

Print summary statistics of data:

```
summary(heartdisease)
```

```
      Age            Sex              RestingBP      Cholesterol
 Min.   :28.00   Length:918         Min.   :  0.0   Min.   :  0.0
 1st Qu.:47.00   Class :character   1st Qu.:120.0   1st Qu.:173.2
 Median :54.00   Mode  :character   Median :130.0   Median :223.0
 Mean   :53.51                      Mean   :132.4   Mean   :198.8
 3rd Qu.:60.00                      3rd Qu.:140.0   3rd Qu.:267.0
 Max.   :77.00                      Max.   :200.0   Max.   :603.0
   FastingBS       RestingECG           MaxHR          Angina
 Min.   :0.0000   Length:918         Min.   : 60.0   Length:918
 1st Qu.:0.0000   Class :character   1st Qu.:120.0   Class :character
 Median :0.0000   Mode  :character   Median :138.0   Mode  :character
 Mean   :0.2331                      Mean   :136.8
 3rd Qu.:0.0000                      3rd Qu.:156.0
 Max.   :1.0000                      Max.   :202.0
 HeartPeakReading   HeartDisease
 Min.   :-2.6000   Min.   :0.0000
 1st Qu.: 0.0000   1st Qu.:0.0000
 Median : 0.6000   Median :1.0000
```

```
 Mean   : 0.8874   Mean   :0.5534
 3rd Qu.: 1.5000   3rd Qu.:1.0000
 Max.   : 6.2000   Max.   :1.0000
```

Splitting the data into train and test set: It allows us to train the model on one part and evaluate its performance on another, ensuring that the model generalizes well to new data and avoids overfitting.

```
index <- sample(1: nrow(heartdisease), round(nrow(heartdisease)*0.80))
train <- heartdisease[index,]
test <- heartdisease[-index,]
```

Building logistic regression model:

```
lr_model <- glm(HeartDisease~.,data=train, family = "binomial")
lr_model
```

```
Call:  glm(formula = HeartDisease ~ ., family = "binomial", data = train)

Coefficients:
     (Intercept)               Age               SexM          RestingBP
        1.392585          0.012494          1.266126          -0.002508
     Cholesterol         FastingBS  RestingECGNormal      RestingECGST
       -0.002444          1.405342         -0.288841          -0.432851
           MaxHR           AnginaY  HeartPeakReading
       -0.022852          1.494141          0.636296

Degrees of Freedom: 733 Total (i.e. Null);  723 Residual
Null Deviance:      1012
Residual Deviance: 636.2    AIC: 658.2
```

Print the summary of the model:

```
summary(lr_model)
```

```
Call:
glm(formula = HeartDisease ~ ., family = "binomial", data = train)
```

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.392585   1.300017   1.071   0.2841
Age               0.012494   0.012498   1.000   0.3175
SexM              1.266126   0.258067   4.906 9.29e-07 ***
RestingBP        -0.002508   0.005470  -0.459   0.6466
Cholesterol      -0.002444   0.001043  -2.343   0.0191 *
FastingBS         1.405342   0.266197   5.279 1.30e-07 ***
RestingECGNormal -0.288841   0.267098  -1.081   0.2795
RestingECGST     -0.432851   0.331862  -1.304   0.1921
MaxHR            -0.022852   0.004590  -4.978 6.41e-07 ***
AnginaY           1.494141   0.230681   6.477 9.35e-11 ***
HeartPeakReading  0.636296   0.112762   5.643 1.67e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1012.30  on 733  degrees of freedom
Residual deviance:  636.16  on 723  degrees of freedom
AIC: 658.16

Number of Fisher Scoring iterations: 5
```

Calculate the probabilities and dividing the possibilities into two groups:

```
summary(lr_model)
```

```
Call:
glm(formula = HeartDisease ~ ., family = "binomial", data = train)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.392585   1.300017   1.071   0.2841
Age               0.012494   0.012498   1.000   0.3175
SexM              1.266126   0.258067   4.906 9.29e-07 ***
RestingBP        -0.002508   0.005470  -0.459   0.6466
Cholesterol      -0.002444   0.001043  -2.343   0.0191 *
FastingBS         1.405342   0.266197   5.279 1.30e-07 ***
RestingECGNormal -0.288841   0.267098  -1.081   0.2795
RestingECGST     -0.432851   0.331862  -1.304   0.1921
```

```
MaxHR            -0.022852   0.004590  -4.978 6.41e-07 ***
AnginaY           1.494141   0.230681   6.477 9.35e-11 ***
HeartPeakReading  0.636296   0.112762   5.643 1.67e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1012.30  on 733  degrees of freedom
Residual deviance:  636.16  on 723  degrees of freedom
AIC: 658.16

Number of Fisher Scoring iterations: 5
```

```r
predicted_probs <- predict(lr_model, test[, -10], type= "response")
head(predicted_probs)
```

```
        1         2         3         4         5         6
0.35524898 0.81047220 0.07443053 0.52968708 0.25363737 0.43299045
```

```r
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
head(predicted_classes)
```

```
1 2 3 4 5 6
0 1 0 1 0 0
```

This code block is used to measure the performance of a classification model. The first four lines calculate the values of TP, FP, TN, and FN based on the model's predictions. The next line shows the class distribution in the training set. Finally, the confusionMatrix() function evaluates the model's performance by comparing predicted and actual classes, providing metrics such as accuracy, precision, recall scores.

```r
confusionMatrix(table(ifelse(test$HeartDisease == "1", "1", "0"),
predicted_classes), positive = "1")
```

```
Confusion Matrix and Statistics

   predicted_classes
     0  1
  0 58 16
```

```
 1 19 91

               Accuracy : 0.8098
                 95% CI : (0.7455, 0.8638)
    No Information Rate : 0.5815
    P-Value [Acc > NIR] : 3.969e-11

                  Kappa : 0.607

 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.8505
            Specificity : 0.7532
         Pos Pred Value : 0.8273
         Neg Pred Value : 0.7838
             Prevalence : 0.5815
         Detection Rate : 0.4946
   Detection Prevalence : 0.5978
      Balanced Accuracy : 0.8019

       'Positive' Class : 1
```

Accuracy (Accuracy): It expresses the ratio of correctly predicted samples. In this case, the accuracy rate is approximately 83.15%. So, the model's correct classification rate is quite high.

95% CI (Confidence Interval): It is the 95% confidence interval of the accuracy rate. This indicates the probability that the accuracy estimate will be within a certain range.

```
(0.7695, 0.8826)
```

Kappa (Kappa Statistic): It indicates how much better the model performs compared to a random model. The higher the value, the better the model performs. In this case, the kappa value is 0.6578.

Sensitivity (Sensitivity): It expresses the true positive rate. That is, it shows how many of the true positives were correctly detected. In this case, the sensitivity rate is approximately 83.81%.

Specificity (Specificity): It expresses the true negative rate. That is, it shows how many of the true negatives were correctly identified. In this case, the specificity rate is approximately 82.28%.

Pos Pred Value (Positive Predictive Value): It expresses the accuracy of positive predictions. That is, it shows how many of the predicted positives were actually positive. In this case, the positive predictive value is approximately 86.27%.

Neg Pred Value (Negative Predictive Value): It expresses the accuracy of negative predictions. That is, it shows how many of the predicted negatives were actually negative. In this case, the negative predictive value is approximately 79.27%.

Prevalence (Prevalence): It expresses the rate of the positive class in the dataset. In this case, the prevalence of the positive class in the dataset is approximately 57.07%.

Detection Rate (Detection Rate): It expresses the rate at which the model correctly detects the positive class. In this case, the detection rate is approximately 47.83%.

Balanced Accuracy (Balanced Accuracy): It is the weighted average of sensitivity and specificity. Balanced accuracy is calculated based on the number of samples in each class and better reflects performance on imbalanced datasets. In this case, the balanced accuracy rate is approximately 83.04%.

Brier Score

The Brier score is a metric that measures the accuracy of probabilistic predictions made by a classification model. It is particularly used in models that predict the probability of an event occurring. The Brier score measures how well the model's predicted probabilities align with the observed outcomes.

The Brier score is the average squared difference between the predicted probability and the observed outcome for each example. A lower Brier score indicates that the model's probability predictions are better aligned with the actual events, suggesting better calibration. Thus, a lower Brier score indicates better model performance.

Therefore, the Brier score is an important metric for evaluating the accuracy and calibration of probability predictions made by a model.

To calculate Brier Score we should install following package:

```
install.packages("ModelMetrics")
```

```
Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

```
library(ModelMetrics)
```

```
Attaching package: 'ModelMetrics'

The following objects are masked from 'package:caret':

    confusionMatrix, precision, recall, sensitivity, specificity

The following object is masked from 'package:base':

    kappa
```

Calculating Brier score:

```
{brier(lr_model)}
```

A Brier score of 0.1458128 indicates the accuracy of probabilistic predictions made by a classification model. This score represents the average squared difference between the predicted probabilities and the actual outcomes for each example.

In practical terms, a Brier score closer to 0 indicates better calibration and accuracy of the model's predicted probabilities. Therefore, a Brier score of 0.1458128 suggests that the model's probability predictions are reasonably well-calibrated, but there is still room for improvement.