***Logistic Regression***

**Aim**: Generate a model to predict the cardiovascular disease instinct(yes/no) based on the independent features based on their profile and tests of dataset. Early detection through machine learning techniques helps in estimating symptoms and behaviors helps in preventing the problems and providing medical assistance at the earliest. Cardiovascular disease is problem with heart, arteries, valves. Many factors like exercise, cholesterol, blood sugar levels cause cardiac issues. Finding the best correlation features among the dataset and estimating the model as follows.

**Algorithm selected**: Logistic Regression is selected for this case as the output has binary outcomes like yes/no,1/0, true/false. It comes under classification problem. It supports categorizing data into discrete classes by studying the relationship from a given set of data. It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function. It always ranges between 0 to 1.  The vertical axis stands for the probability for a given [classification](https://www.sciencedirect.com/topics/computer-science/classification) and the horizontal axis is the value of *x*.

Diagram

Description automatically generated

Logistic Regression uses maximum likelihood also known as conditional probability as a loss function in which if the probability is greater than 0.5 it is assigned to class 0, otherwise class 1 assigned,

Logit function is defined as the natural log of the odds. A probability of 0.5 corresponds to a logit of 0, probabilities smaller than 0.5 correspond to negative logit values, and probabilities greater than 0.5 correspond to positive logit values. It ranges from negative infinity to positive infinity.

Odds= p/(1-p)

GLM in R has capability to do various linear models like logistic, Poisson, gamma, Gaussian inverse Gaussian, binomial with each having their specific link function by allowing the magnitude of the variance of each measurement to be a function of its predicted value. These are classified based on the outcome variable. GLM is useful when you output variable is constrained or the variance is not constant.

1. **DATA COLLECTION**: For this, real world patient dataset Statlog is picked from UCI machine learning repository http://archive.ics.uci.edu/ml/datasets/ Statlog+%28Heart%29

We will try importing the csv file.

* heart=read.csv("heart.csv")
* 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 ...*

The dataset has 270 observations of patients with 14 variables. It has numeric, binary, and categorical features treated as integers. We must turn them into factors later. Understanding some of the columns like.

* Chest pain- chest pain type (4 values)
* Rest BP- resting blood pressure
* Chol- serum cholesterol in mg/dl
* Sugar- fasting blood sugar > 120 mg/dl
* ECG- resting electrocardiographic results (values 0,1,2)
* Max hr-maximum heart rate achieved
* Angina- exercise induced angina
* Dep-   depression induced by exercise relative to rest
* Fluor- number of major vessels (0-3) colored by fluoroscopy
* Thal- Thalassemia detection 3 = normal; 6 = fixed defect; 7 = reversable defect

**2.DATA PREPARATION**: Categorial variables must be changed to factor levels to reduce the risk of considering as numeric variables. We have 4 categorial variables in the dataset chestpain, ECG, thal and exercise.

* 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 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 : Factor w/ 3 levels "0","1","2": 3 3 1 1 3 1 3 3 3 3 ...

$ 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 : Factor w/ 3 levels "1","2","3": 2 2 1 2 1 1 2 2 2 2 ...

$ 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 3 2 3 3 3 ...

$ OUTPUT : int 1 0 1 0 0 0 1 1 1 1 ...

This changes 4 of them into factors. Output 1 indicates the presence and 0 indicates absences of heart disease.

Before training the model, we will have to split the dataset into training and test data set. We are now doing 85-15% split for both the datasets. So that we will have 230 observations in training and 40 observations in test.

Setting the seed to same number picks same instances every time you perform the test instead of random sampling. This gives us the same result every time we perform the test.

* library(caret)
* set.seed(987954)
* 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]

Now that our models are ready, lets train our model

**3.MODEL TRAIN:** In R, we use glm() function to train generalized linear regression models. Here we are fitting the data using logistic regression model. Output variable is predicted against all other independent features. As the output has only 2 factors (0,1) we consider binomial in this case and use logit as the link function.

* heart\_model=glm(OUTPUT ~ ., data = heart\_train, family = binomial("logit"))
* summary(heart\_model)

Table

Description automatically generated

Regression coefficients are represented corresponding to the z statistic. Higher the absolute value of the z statistic, the more likely the feature is related to our output variable. The p-values next to the z-statistic express the probability and are denoted with stars and dots, as they were in linear regression, indicating the smallest confidence interval that includes the corresponding p-value.

P values for fluor, chestpain4, thal7 are significant and shows strong effect of the output but others aren’t good. We can’t conclude that these are bad predictors, but they might have no effect in the presence of other features.

**Manual Computation of results in Summary:**

**Deviance:** It is same as the residual in linear regression. The deviance of an observation can be computed as the -2 times the log likelihood of that observation. The deviance of a data set is sum of all the observation deviances. The deviance residual of an observation is derived from the deviance itself and is like the residual of a linear regression

Recreating functions for deviance and null deviance.

* 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")

[1] 149.2286

Null deviance is similar to true sum of squares in linear regression. A null model has no features. Residual deviance is residual sum of squares.

* null\_deviance(data = heart\_train, output\_column = "OUTPUT")

[1] 317.1064

Pseudo r squared explains the variance observed by output variable

* 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] 0.5294053

This shows that model explains 53% of null deviance. Recreating the deviance summary we have using function

* 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.

-2.58584 -0.50992 -0.14016 -0.02765 0.33568 2.56545

**4.EVALUATING THE MODEL:** Predicting the output of our model using predit() function. This output is the probability of the input belonging to class 1. We can perform binary classification by applying a threshold. We'll do this with both our training and test data and compare them with our expected outputs to measure the classification accuracy

* 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] 0.8695652

* 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] 0.9

Classification accuracies on the training and test sets are very similar and are close to 90 percent.

* (confusion\_matrix=table(predicted =train\_class\_predictions, actual = heart\_train$OUTPUT))

A picture containing chart

Description automatically generated

* (precision= confusion\_matrix[2, 2] / sum(confusion\_matrix[2,]))

[1] 0.8947368

Recall or Sensitivity is the ratio of correctly identified instances of class 1, divided by the total number of observations that actually belong to class 1. Also called False Positive Rate. Here, labelling the patient as healthy with underlying heart ailments is a costly error we have 20 such errors made by our model.

* (recall=confusion\_matrix[2, 2] / sum(confusion\_matrix[,2]))

[1] 0.8095238

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

[1] 0.85

False Negative Rate is Specificity. Correctly identified members of class 0 over all the observations of class 0 in our data set

* (specificity= confusion\_matrix[1,1]/sum(confusion\_matrix[1,]))

[1] 0.8518519

We need consider a cutoff value to justify our classification metrics. Our cutoff values should not be biased to single class. We obtain this using precision recall curves from ROCR package in R.

* 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)

Chart, scatter chart

Description automatically generated

It is evident that to achieve recall of 0.8, we must sacrifice precision suddenly. Cutoff should depend on the priority to lose precision pr recall. To achieve the better cutoff let’s compute to achieve 90% recall and 0.8 % precision.

* thresholds=data.frame(cutoffs = perf@alpha.values[[1]],recall =perf@x.values[[1]], precision = perf@y.values[[1]])
* subset (thresholds,(recall > 0.9) & (precision > 0.8))

cutoffs recall precision

112 0.3491857 0.9019608 0.8288288

113 0.3472740 0.9019608 0.8214286

114 0.3428354 0.9019608 0.8141593

115 0.3421438 0.9019608 0.8070175

Optimum cut off value is 0.35. So, score above 0.35 is classified as 1 and below it is classified as 0.

**Conclusion:** Logistic regression model is constructed with 87% accuracy rate and cut off threshold is determined as 0.35. Out of all the feature variables available only 4 of them were significant and other variable aren’t showing any effect on output when other features presence. I think this model is bit overfitting as the model underperforms in training with 87% and in test it was predicted as 90%. Also, the pseudo r square showed that model didn’t explain enough deviance

**Limitations of Linear Regression:**

* Including independent features which have no effect on output may not yield results as this might increase the standard errors and dilute the actual associations.
* If input variables are highly correlated with one another then the effect of each on the regression model becomes less precise
* The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. This assumption may not hold true for certain associations