**Logistic Regression:**

Definition:

In [statistics](https://en.wikipedia.org/wiki/Statistics), the logistic model (or logit model) is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that is usually taken to apply to a **binary dependent variable**. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), logistic regression or logit regression is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model. More formally, a logistic model is one where the [log-odds](https://en.wikipedia.org/wiki/Log-odds) of the probability of an event is a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of independent or predictor variables. The two possible dependent variable values are often labelled as **"0"** and **"1",** which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick

**R CODE:**

data<−read.csv(file="R/data/pima.csv",stringsAsFactors=TRUE,head=TRUE):

To read data from the csv file from our local directory.

library(dplyr):

**dplyr** provides a flexible grammar of data manipulation. It's the next iteration of plyr, focused on tools for working with data frames (hence the *d* in the name).

data = select(data,-X):

This command will remove the name of the column from the select dataset .

In our case we want to remove the column with the name X as it is unwanted column or us.

library(caTools):

Contains several basic utility functions including: moving (rolling, running) window statistic functions, read/write for GIF and ENVI binary files, fast calculation of AUC, LogitBoost classifier, base64 encoder/decoder, round-off error free sum and cumsum, etc.

split = sample.split(data,SplitRatio = 0.7):

Used to split the whole dataset into a desired SplitRatio datasets .In our case it is 70%.

train\_data = subset(data,split == "TRUE"):

As the split is TRUE we will get a train\_data of 70% of data set.

test\_data = subset(data,split =="FALSE"):

As the split is FALSE we will get a test\_data of remaining 30% of data set.

model = glm(type~.,train\_data,family = "binomial"):

**glm** is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

**GLM = Generalized Linear Model**

type~., = This argument takes all the predictors from opur dataset

train\_data = This is our testing data set

family = Family objects provide a convenient way to specify the details of the models used by functions such as [glm](http://127.0.0.1:34326/help/library/stats/help/glm).

summary(model):

**Call:**

glm(formula = type ~ ., family = "binomial", data = train\_data)

**Deviance Residuals**:

Min 1Q Median 3Q Max

-2.7148 -0.6993 -0.4321 0.6956 2.3495

**Coefficients:**

Estimate Std. Error z value Pr(>|z|)

(Intercept) -7.442759 1.459490 -5.100 3.40e-07 \*\*\*

npreg 0.167196 0.072027 2.321 0.0203 \*

glu 0.033512 0.006469 5.181 2.21e-07 \*\*\*

bp -0.016167 0.014539 -1.112 0.2662

skin 0.000295 0.024674 0.012 0.9905

bmi 0.066042 0.034309 1.925 0.0542 .

ped 1.018145 0.554172 1.837 0.0662 .

age 0.005060 0.022529 0.225 0.8223

---

**Signif. codes:** 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

**Null deviance:** 256.77 on 207 degrees of freedom

**Residual deviance:** 190.82 on 200 degrees of freedom

**AIC:** 206.82

Number of Fisher Scoring iterations: 5

**Call:** This will give us the information that we used for the model.

**Null Deviance & Residual Deviance:** The **Null Deviance** shows how well the response variable is predicted by a model that includes only the intercept (grand mean) where as **Residual Deviance** with inclusion of independent variables.

**AIC**: The **Akaike information criterion (AIC)** is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models.

**It should be as low as possible.**

model = glm(type~.-age,train\_data,family = "binomial"):

As highlighted in the above command those predictor variables have less significance so we will remove one by one and see the summary values of each and decide whether that particular variable can be removed or not

summary(model):

Null deviance: 256.77 on 207 degrees of freedom

Residual deviance: 190.87 on 201 degrees of freedom

AIC: 204.87

model = glm(type~.-bp,train\_data,family = "binomial"):

summary(model):

Null deviance: 256.77 on 207 degrees of freedom

Residual deviance: 192.05 on 201 degrees of freedom

AIC: 206.05

model = glm(type~.-skin,train\_data,family = "binomial")

summary(model):

Null deviance: 256.77 on 207 degrees of freedom

Residual deviance: 190.82 on 201 degrees of freedom

AIC: 204.82

As we can see that we have the less value of AIC when the skin predictor is removed.

So we will finalize that this is our model.

result = predict(model,test\_data,type = "response"):

This predict command will predict the results of test data with the model that we have created.

install.packages("ROCR"):

library(ROCR): Used for the ROC curve inorder to get the threshold value,

result = predict(model,train\_data,type = "response"):

In order to get the threshold value we need the training data so we will predict based on the training data

ROCRpred = prediction(result , train\_data$type):

Every classifier evaluation using ROCR starts with creating a prediction object. This function is used to transform the input data (which can be in vector, matrix, data frame, or list form) into a standardized format.

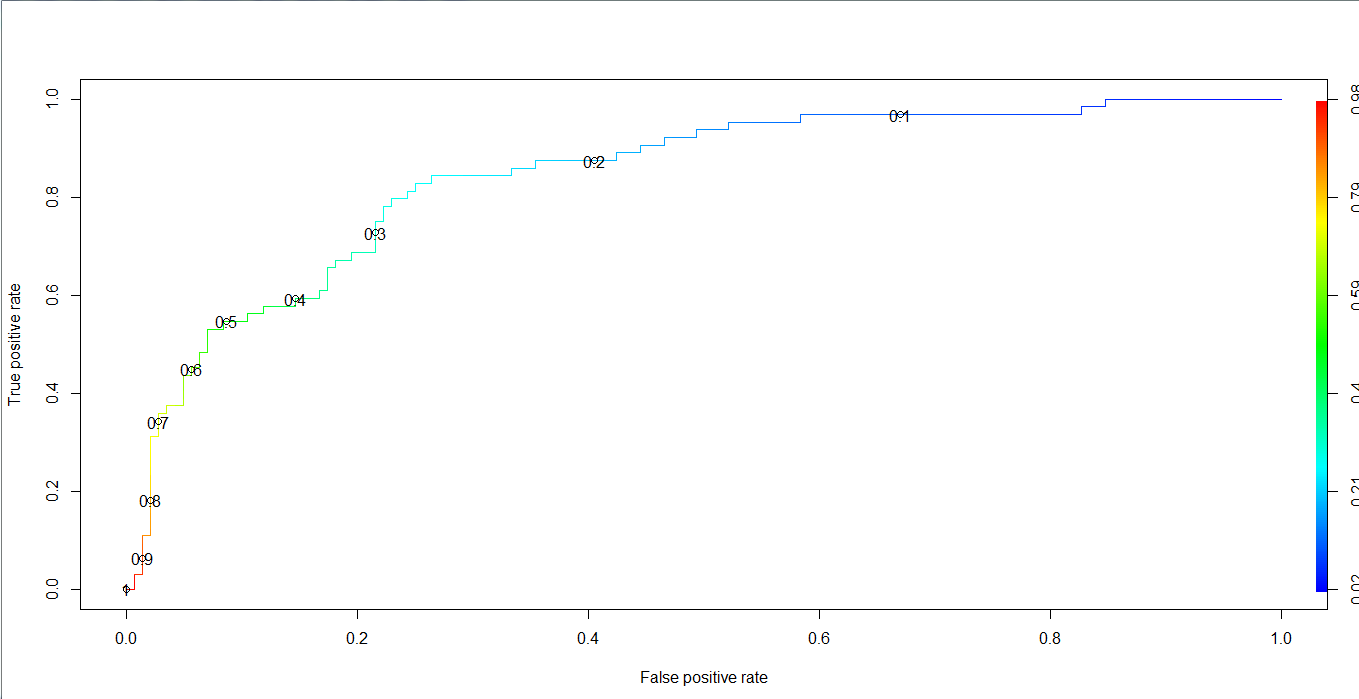
ROCRperf = performance(ROCRpred,"tpr","fpr"):

All kinds of predictor evaluations are performed using this function.

Tpr = true positives

Fpr = False positives

plot(ROCRperf,colorize=TRUE,print.cutoffs.at=seq(0.1,by=0.1)):



**From the plot we will come to know that 0.3 is our threshold value**

result = predict(model,test\_data,type = "response"):

Now we predict the values of our test\_data.

conf\_matrix =table(ActualValue = test\_data$type,PredictedValue = result>0.3):

Create a confusion matrix which will tell us the correct and wrong predicted values

conf\_matrix:

PredictedValue

ActualValue FALSE TRUE

No 72 7

Yes 10 35

accuracy = (sum(diag(conf\_matrix)))/(nrow(test\_data)):

This command will calculate the accuracy of the model.

accuracy :

0.8629032