# Logistic Regression Using R

---

## Introduction:

This document describes a high level process for building Logistic regression model in R. Document has been prepared in relevance to contest "Business Analytics for Beginners Using R-Part II"
https://www.crowdanalytix.com/contests/business-analytics-for-beginners-using-r---part-ii

R codes are also given as an example working with some variable for the solvers to get them started. Below mentioned sections describe the process in brief:

## Contents





CrowdANALYTIX

## Section 1: Understanding the business Objective and variables

Solvers should start with downloading all relevant data and data dictionary and reading through the contest description thoroughly. It is important to keep end objective in mind while preparing your data for analysis. Solvers should go through each of the variables to understand its meaning and develop a general hypothesis as how the independent variable could impact the dependent variable in question. Types of technique for modeling as well as data treatment can vary depending on, if variable is continuous, nominal or ordinal.

Here the dependent variable is 'Dichotomous', so will focus on developing a Logistic regression model. Further process has been outlined under the assumption that data is obtained after missing value treatment and ready for modeling (except recoding of variables).

## Section 2: Importing data in R and recoding ordinal variables

First we need to import data and row bind them so any transformation can be applied together. Although R considers character variable and will not give error during analysis. It internally creates level for character variables by giving numerical level in alphabetical order and R reads it as factor variable which can be a problem if variable is ordinal in nature. So for variable 'Founders_previous_company_employee_count' where levels are Small, Medium and Large; Large will become 1, Medium will become 2 and Small will become 3 alphabetically. This can create problem with ordinal variables. It is always a good idea to recode ordinal variables as numeric ordered variables.

```
# importing train and test data
train<- read.csv(file="CAX_Startup_Train.csv", header=TRUE,as.is=T)
test<- read.csv(file="CAX_Startup_Test.csv", header=TRUE,as.is=T)
train$Category<-c("Train")
test$Category<-c("Test")

# row binding imported train and test data
d<-rbind(train,test)

# recoding ordinal variables
library(plyr)
d$employee_count_code<-
as.numeric(revalue(d$Founders_previous_company_employee_count,
            c("Small"=1, "Medium"=2, "Large"=3)))
```

If you don't want to rely on plyr, you can do the following with R's built-in functions:

```
# recoding with R built-in functions
d$employee_count_code[d$Founders_previous_company_employee_count==
"Small"] <- 1
d$employee_count_code[d$Founders_previous_company_employee_count==
"Medium"] <- 2
```

```
d$employee_count_code[d$Founders_previous_company_employee_count==
"Large"] <- 3
```

Here variable will be taken as numeric which you can verify using str(). Then you can set the original variable to NULL.

```
# removing original variable from data frame
d$Founders_previous_company_employee_count = NULL
```

Once solver have recoded all the necessary variable data can be used for modeling.

## Section 3: Making Your Own Test data

Before proceeding with variable selection and model building process solver should make their own test set to check the performance of the model they will develop. The percentages of observation kept for testing depends on total number of observations available. For large dataset you can keep 40% for testing and for small dataset even 10% can do the job. Idea is to keep enough number of observation to train your model and perform testing simultaneously.

Here you can keep out 10% of data for testing your model. Care should be taken to randomly select the 0 and 1 cases in test set proportional to your training dataset. Here 0 and 1 have same number of cases, 118 and 116 observations respectively. So, a random sampling of 10% observation from each of the levels will do.

```
# partitioning of test and train set for own evaluation of models
# seprating out 0 and 1 level
train_0 <- train[train$Dependent==0,]
train_1 <- train[train$Dependent==1,]
train_0$Category<-NULL
train_1$Category<-NULL

# randomly choosing test and train set for each level
library(caTools)
sample_0 = sample.split(train_0, SplitRatio = .9)
train_0_new = subset(train_0, sample_0 == TRUE)
test_0_new = subset(train_0, sample_0 == FALSE)

sample_1 = sample.split(train_1, SplitRatio = .9)
train_1_new = subset(train_1, sample_1 == TRUE)
test_1_new = subset(train_1, sample_1 == FALSE)

# final new train and test set
train_new<- rbind(train_1_new,train_0_new)
test_new<- rbind(test_1_new,test_0_new)
```

## Section 4: Variable Selection Using Information Value

Information Value ('IV') for logistic regression is analogous to correlation for linear regression. Information value tells us how well an independent variable is able to distinguish two categories of dependent variables. Please refer to below links for more details:
http://ucanalytics.com/blogs/information-value-and-weight-of-evidencebanking-case/

There is no standard package for 'IV' calculation in R but a github implementation can be downloaded and used:

```
# install developer tool and then woe package from gitub
#  refer  http://www.r-bloggers.com/r-credit-scoring-woe-information-value-in-woe-package/
install.packages("devtools")
library(devtools)
install_github("tomasgreif/woe")
library(woe)

# only dependent and independent variable of training set
# should be there in data frame
train_new$CAX_ID<-NULL

# calculation of information value
IV<-iv.mult(train_new,y="Dependent",TRUE)

# Ignore warnig message for variable which have 0 WOE
# (anyway you will remove these before modeling)
```

Now the 'IV' of variables can be used to select variables for modeling. Generally we select variable with 'IV' of 0.1 to .5 for modeling. Another approach could be to do a grid search for combination of variables which gives best AUC on test set. Here I will describe the modeling process only for variable selection using 'IV'. Solver should try another approach also.

```
# selecting variables with 0.1<IV<0.5
var<-IV[which(IV$InformationValue>0.1),]
var1<-var[which(var$InformationValue<0.5),]
final_var<-var1$Variable

x_train<-train_new[final_var]
Dependent<-train$Dependent
train_final<-cbind(Dependent,x_train)
```

## Section 5: Model Building

Now a stepwise logistic regression model can be fitted with selected variables. The stepwise procedure will select the best combination of variables fitting the training data.

```
# fitting stepwise binary logistic regression with logit link function
mod<-step(glm(Dependent~., family = binomial(link=logit),data = train_final))

# model summary
summary(mod)
```

output:

```
Call:
glm(formula = Dependent ~ Company_business_model +
Company_competitor_count + Company_1st_investment_time +
Founders_Data_Science_skills_score + Company_big_data +
Founders_publications + Founders_global_exposure,
family = binomial(link = logit), data = train_final)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1920  -0.9813  -0.2354   0.9389   2.2444

Coefficients:
                                 Estimate Std. Error z value Pr(>|z|)
(Intercept)                       0.27177    0.48369   0.562 0.574212
Company_business_modelB2C        -1.39215    0.42061  -3.310 0.000934 ***
Company_business_modelBoth        1.30826    1.17252   1.116 0.264521
Company_competitor_count         -0.11438    0.05757  -1.987 0.046928 *
Company_1st_investment_time       0.01859    0.01007   1.847 0.064805 .
Founders_Data_Science_skills_score  0.03634    0.02173   1.672 0.094464 .
Company_big_dataYes               2.25752    0.92881   2.431 0.015076 *
Founders_publicationsMany        -0.81199    0.57544  -1.411 0.158226
Founders_publicationsNone        -1.00840    0.47483  -2.124 0.033695 *
Founders_global_exposureYes       0.65902    0.35625   1.850 0.064329 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 288.33  on 207  degrees of freedom
Residual deviance: 232.30  on 198  degrees of freedom
AIC: 252.3

Number of Fisher Scoring iterations: 4
```

There could be some variables which are insignificant at very high level of significance like Company_business_model with p-value of 0.26. This variables can be removed from equation.

Please note although intercept is insignificant it should not be removed from equation until no physical interpretation is associated with intercept representing the real world scenario. Here even if you don’t consider any variable in model intercept denote the minimum level of probability of success of company which represent the true nature of real life scenario. So, it will makes perfect sense to keep the intercept.

Solver can retain all variable below 5%, 10% or 20% level of significance based on nature of variable coming in model, their business interpretation and if they are improving the model accuracy.

Please note the result given here are just for illustration as I have not recoded all variables and result may not be replicated when solvers will try their model after recoding all variable.

Now the final model can be fitted with significant variable at 20% level of significance.

```
# final logistic regression model
model<-glm(formula = Dependent ~ Company_competitor_count +
Company_1st_investment_time + Founders_Data_Science_skills_score +
Company_big_data + Founders_publications + Founders_global_exposure,
family = binomial(link = logit), data = train_final)

# model summary
summary(model)
```

output:

```
Call:
glm(formula = Dependent ~ Company_competitor_count +
Company_1st_investment_time + Founders_Data_Science_skills_score +
Company_big_data + Founders_publications + Founders_global_exposure,
family = binomial(link = logit), data = train_final)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3131  -0.9925  -0.4178   1.0408   1.9929

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -0.12638    0.45101  -0.280  0.77931
Company_competitor_count        -0.09980    0.05351  -1.865  0.06220 .
Company_1st_investment_time      0.02514    0.01009   2.491  0.01273 *
Founders_Data_Science_skills_score  0.05174    0.02196   2.357  0.01844 *
Company_big_dataYes              1.98897    0.88167   2.256  0.02408 *
Founders_publicationsMany       -1.04961    0.55778  -1.882  0.05987 .
Founders_publicationsNone       -1.26650    0.45946  -2.756  0.00584 **
Founders_global_exposureYes      0.83353    0.34272   2.432  0.01501 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 288.33  on 207  degrees of freedom
Residual deviance: 247.16  on 200  degrees of freedom
AIC: 263.16

Number of Fisher Scoring iterations: 4
```

Here the coefficient estimate given are of log(odds) of variable. Also please note log(odds) are given for each level of categorical variable against the base value. For variable ‘Founders_publications’ , ‘Few’ is base level since it did not appear in coefficients and estimated value of log(odds) for each level as compared to base level is given. Here sign of

estimate will give idea of directionality and a sense check should be applied to see if the directionality makes business sense and can be used to give insights while reporting.

As stated earlier the estimate of coefficients are log(odds). To report the coefficient in terms of odds ratio we should use

```
# odds ratios and 95% CI
exp(cbind(OR = coef(model), confint(model)))
```

output:

```
                                       OR     2.5 %      97.5 %
(Intercept)                     0.8812767 0.3652939   2.1804303
Company_competitor_count        0.9050210 0.8071233   0.9962430
Company_1st_investment_time     1.0254594 1.0066382   1.0473617
Founders_Data_Science_skills_score 1.0531047 1.0124638  1.1028928
Company_big_dataYes             7.3079721 1.5131251 55.6651728
Founders_publicationsMany       0.3500743 0.1136074   1.0238319
Founders_publicationsNone       0.2818160 0.1101670   0.6761404
Founders_global_exposureYes     2.3014250 1.1864030   4.5686515
```

Here odds ratio of above 1 indicate, with change in level of independent variable the odds of success of company increases by 'odds ratio estimated value' times and for odds ratio below 1 odds of success decreases by that much times. For details of logistic regression and variable interpretation please refer to http://www.ats.ucla.edu/stat/r/dae/logit.htm

You can include an interaction effect of two main effect variable also by simply adding 'variable1*variable2' in equation. This can improve the model drastically if the effect of one predictor variable on the response variable is different at different values of the other predictor variable. Interaction is slightly complicated concept and solvers are encouraged to read about it and make some more research to become familiar as it can help a lot in improving your model accuracy
http://www.theanalysisfactor.com/interpreting-interactions-in-regression/

To test for overall goodness of fit of your logistic regression model 'Hosmer-Lemeshow' test can be done.

```
# model fit (Hosmer and Lemeshow goodness of fit (GOF) test)
library(ResourceSelection)
hoslem.test(train_new$Dependent,model$fitted.values, g=10)
```

output:

```
Hosmer and Lemeshow goodness of fit (GOF) test

data:  train_new$Dependent, model$fitted.values
X-squared = 6.3271, df = 8, p-value = 0.6106
```

p-value is well above 0.05 indicating there is no evidence of poor fit. In other words, model is fitting the data well.

## Section 6: Predicting Test Score and Model Evaluation

Once you have built your first model, you can start evaluating its predictive power using model accuracy measures like AUC, ROC and confusion matrix with respect to your own test set. Make sure to check the accuracy using multiple metrics to select the best model and tune your models to improve the accuracy.

```
# Prediction on test set
pred_prob<-predict (model, newdata=test_new, type="response")

# model accuracy measures
library (ROCR)
pred <- prediction (pred_prob, test_new$Dependent)
# Area under the curve
Performance (pred, 'auc')

# creating ROC curve
roc <- performance (pred,"tpr","fpr")
plot (roc)
```

There are several ways you can choose the optimal cut-off for predicted probability (say 0.3, any predicted value below 0.3 will become '0' and >= 0.3 will become '1'). One of the way is using ROC. We will get value of "True Positive Rate" and "False Positive Rate" at each of the cut-off of predicted probability to choose the best cut-off meeting our criteria.

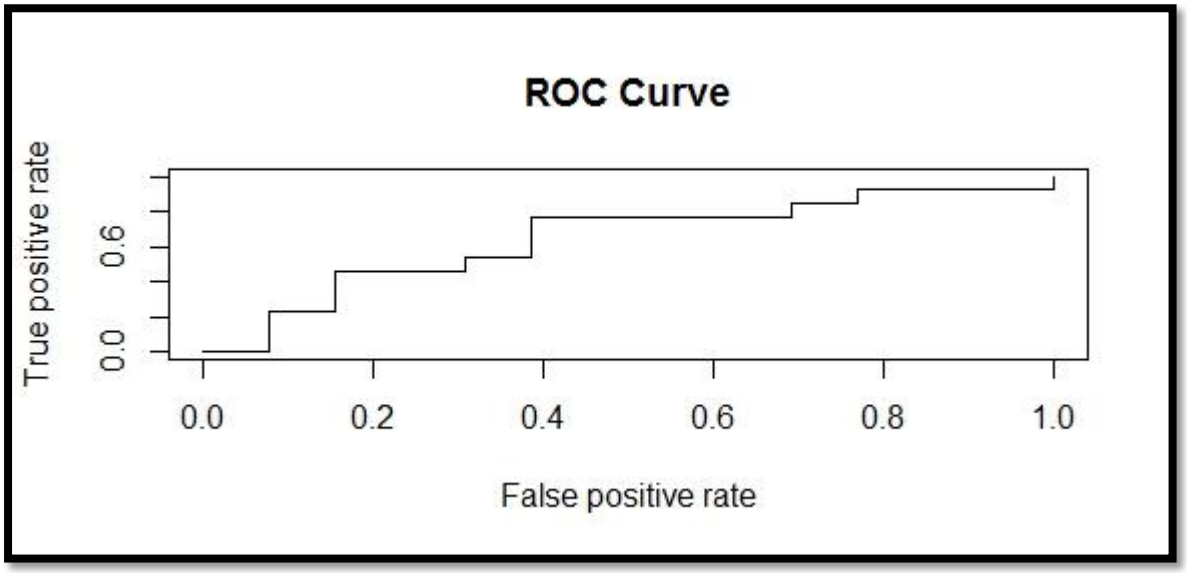


*Figure 1 ROC Curve of model. Optimal point seem to be at cut-off probability of 0.4*

The ROC Curve can be seen for probability point causing largest separation between 'TPR' and 'FPR'. For this we can plot the values as given below:

```
# create data frame of values
perf <-as.data.frame(cbind(roc@alpha.values[[1]], roc@x.values[[1]],
roc@y.values[[1]]))
colnames(perf) <-c("Probability","FPR","TPR")

# removing infinity value from data frame
perf <-perf[-1,]

# reshape the data frame
library(reshape)
perf2<- melt(perf, measure.vars = c("FPR", "TPR"))

# plotting FPR, TPR on y axis and cut-off probability on x axis
library(ggplot2)
ggplot(perf2, aes(Probability, value, colour = variable)) +
  geom_line()+ theme_bw()
```

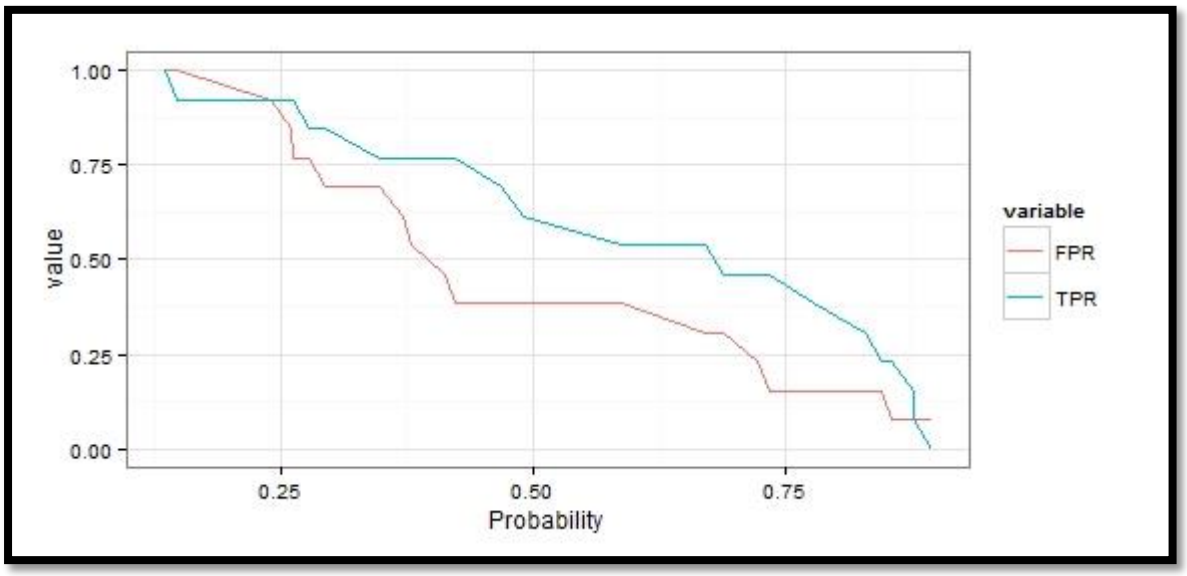


*Figure 2 Chart showing separation between FPR and TPR. Largest separation seem to be at cut-off probability of 0.4*

Both the above charts show optimal probability cut-off of 0.42 which can also be concluded by inspecting the 'perf' data frame.

Sometime we may wish to choose a more balanced probability cut-off optimising correct classification of both '0' and '1'. For this we will need to plot accuracy of '0' and '1' at multiple cut-offs of probability.

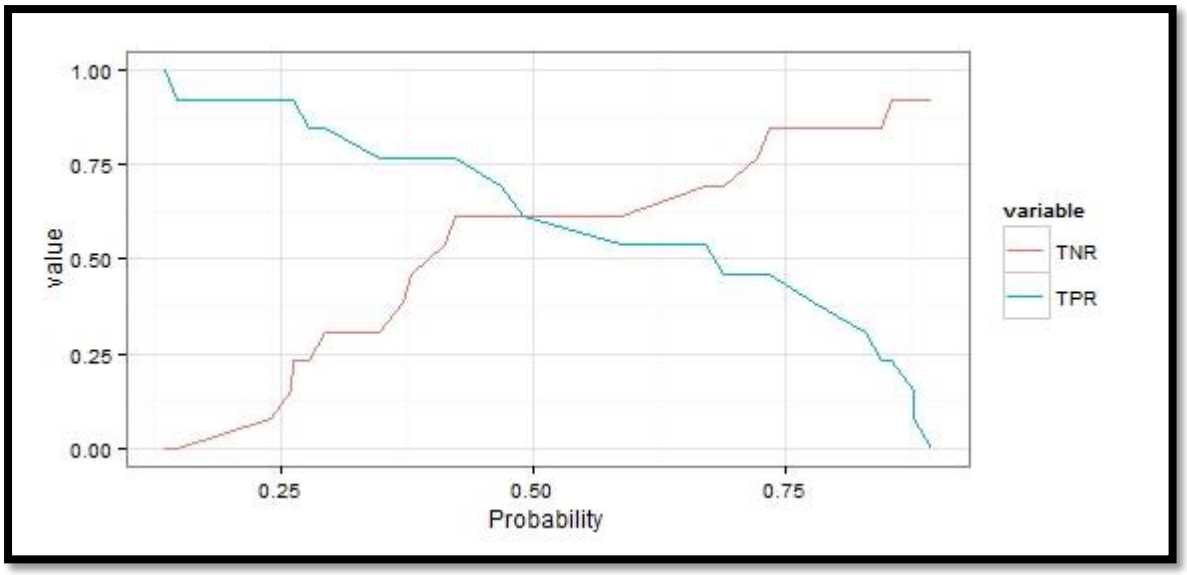


*Figure 3 TPR and TNR at various cut-off probability indicating optimal probability cut-off at 0.42*

The 'TNR' vs 'TPR' also indicated the optimal cut-off at 0.42 which can also be concluded by inspecting 'TR' data frame. Solver can also see the confusion matrix using code below

```
# model accuracy - Confusion Matrix
library(SDMTools)
confusion.matrix (test_new$Dependent, pred_prob, threshold = 0.42)
```

output:

```
    obs
pred 0  1
   0 8  3
   1 5 10
attr(,"class")
[1] "confusion.matrix"
```

Based on business requirement we can optimise our model with respect to any of the metrics. Sometime categorising or binning your continuous variable can also help in improving the overall model accuracy. Binning is particularly useful when there are lot of outliers in any continuous variables.

Once a model has been finalized the predicted probabilities for automated leaderboard can be submitted

```
# Prediction on test set of CAX
pred_CAX<- predict(model, newdata=test, type="response")
submit_CAX<- cbind(test$CAX_ID,pred_CAX)
colnames(submit_CAX)<- c("CAX_ID", "Dependent")
write.csv(submit_CAX,"Predictions.csv",row.names=F)
```

The output 'Predictions.csv' file can be submitted for automated leaderboard score.