Tech gig

2018

Approach document

Data science Semifinal hackathon

Kukreja, Bharti

Techgig's Geek goddess 2018

**Model approach**

**Data understanding**

First of all, it took time for understanding the data definition of fields present in data. There were fields that were not self-explanatory, but data dictionary file helped there.

**Data loading**

Data files were loaded into R studio after converting them into CSV format.

**Data Exploration**

The data contains 500 training records for each of the 5 deodorants giving 2500 records in all. The data was combined together as single data file, to increase the strength of training data for model building. More data and simple model always beats the complex model with less training data.

Based on explanation in data dictionary, data types were given to columns like factors to categorical, nominal to yes no fields, categorical ordinal to fields that had order, and numerical to numeric fields.

**Precautionary step**

Since there were many categorical factors, so to avoid the new factor inclusion in test data at prediction time, the train and test data were combined before categorizing the fields.

**SPSS analysis**

Then to choose the model to start with, the train data file was fed to SPSS and auto analyzer was applied.

I got decision trees, neural network and logistic regression as 3 options, with logistic regression on top with very less accuracy.

Step 5->

I applied logistic regression with threshold of 0.50 and logit binomial family.

On running the model I got this

**Warning message: glm.fit: fitted probabilities numerically 0 or 1 occurred.**

Now the problem I was facing is imperfect separation in logistic regression. How to deal with perfect separation in logistic regression? All I needed was to utilize a form of penalized regression. On googling I found that, this is the original reason some of the penalized regression forms were developed (although they turned out to have other interesting properties.

But for glmnet you have to only feed matrices, not formulas as we do usually. So I used model.matrix() to construct this matrix from the dataframe.

This was the major turn when I switched from glm to glmnet with alpha=0.755 to blend with Lasso and Ridge.

Tried various values of lambda and then decide to go with best lambda suggested at alpha=1 for lasso only. Previously my model was overloaded with so many factors and less information, with glmnet that also got sorted out.

Step 6->

Predicted the results and attached them with test data to submit results.

On the first attempt all the predictions turned out to be right.

After this I gave a try to decision tree and ensemble techniques but used simple to avoid overfitting and bias problems.

**MODEL**

List of 13

$ a0 : Named num [1:873] -1.11 -1.11 -1.11 -1.1 -1.1 ...

..- attr(\*, "names")= chr [1:873] "s0" "s1" "s2" "s3" ...

$ beta :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots

.. ..@ i : int [1:4711] 2 2 2 2 2 2 2 2 2 2 ...

.. ..@ p : int [1:874] 0 0 1 2 3 4 5 6 7 8 ...

.. ..@ Dim : int [1:2] 178 873

.. ..@ Dimnames:List of 2

.. .. ..$ : chr [1:178] "Train\_filtered.frag\_usd\_past\_6\_mths\_23" "X.Intercept." "Train\_filtered.personal\_opinion\_3.L" "Train\_filtered.personal\_opinion\_3.Q" ...

.. .. ..$ : chr [1:873] "s0" "s1" "s2" "s3" ...

.. ..@ x : num [1:4711] -0.0195 -0.0391 -0.0587 -0.0784 -0.0981 ...

.. ..@ factors : list()

$ df : int [1:873] 0 1 1 1 1 1 1 1 1 1 ...

$ dim : int [1:2] 178 873

$ lambda : num [1:873] 0.13 0.129 0.128 0.127 0.126 ...

$ dev.ratio : num [1:873] -1.51e-14 3.43e-03 6.86e-03 1.03e-02 1.37e-02 ...

$ nulldev : num 2796

$ npasses : int 3377

$ jerr : int 0

$ offset : logi FALSE

$ classnames: chr [1:2] "1" "2"

$ call : language glmnet(x = x, y = y, family = "binomial", alpha = 0.755, nlambda = 1000, standardize = FALSE, maxit = 1e+05)

$ nobs : int 2500

- attr(\*, "class")= chr [1:2] "lognet" "glmnet"

And the best lambda is

> cv.glmmod <- cv.glmnet(x, y=as.double(Train\_filtered$Instant\_liking\_target), alpha=1)

> plot(cv.glmmod)

> (best.lambda <- cv.glmmod$lambda.min)

[1] 0.005342969

