Data Science

Chapter 5 Exercise

Student:

Solmaz Mohammadi

Teacher:

Prof. Faraji

CHAPTER 5 EXERSICE Media 6 Degrees

Media 6 Degrees kindly provided a dataset that is perfect for exploring logistic regression models, and evaluating how good the models are. Follow along by implementing the following R code. The dataset can be found at https://github.com/oreillymedia/doing_data_science.

At first, we load the dataset with “readr” library. We have 22 columns and 54584 observations in the dataset. By adding the "rand" column and taking a threshold of .65, the dataset is divided to test and train sets. Then we take a copy of column "y_buy" and call it "y" in the original dataset.

```
> names(set)
 [1] "at_buy_boolean"            "at_freq_buy"               "at_freq_last24_buy"
 [4] "at_freq_last24_sv"         "at_freq_last24_sv_int_buy" "at_freq_sv"
 [7] "at_freq_sv_int_buy"        "at_interval_buy"           "at_interval_sv"
[10] "at_interval_sv_int_buy"    "expected_time_buy"         "expected_time_sv"
[13] "expected_time_sv_int_buy"  "last_buy"                  "last_sv"
[16] "last_sv_int_buy"           "multiple_buy"              "multiple_sv"
[19] "multiple_sv_int_buy"       "uniq_content_links"        "num_checkins"
[22] "y_buy"
```

Now we want to look at some basics used in functions and understand them.

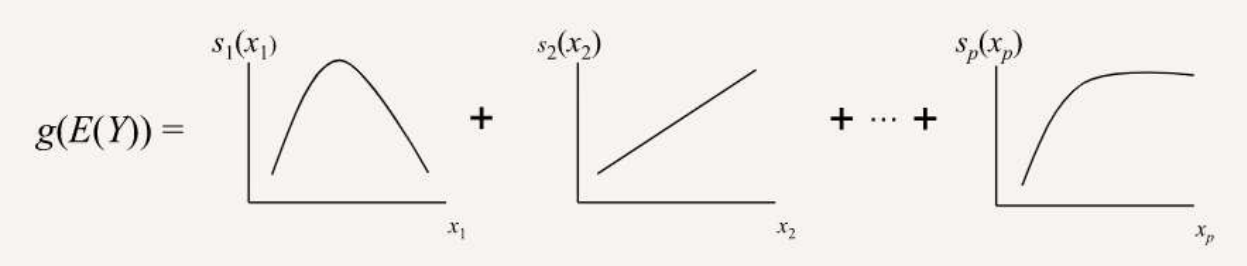
What is GAM?

A generalized additive model (GAM) is a generalized linear model (GLM). This is a good solution to the problem of putting *non-linear functions* in a *linear predictor*. Think of a scatter plot that requires a curvy and non-linear fitting model. By using a generalized additive model, parameters are treated as random effects.

The GAM framework is based on an appealing and simple model:

- Relationships between the **individual predictors** and the **dependent variable** follow smooth patterns that can be linear or nonlinear.
- We can estimate these smooth relationships simultaneously and then predict g(E(Y))) by simply adding them up.

GAM is an **additive** modeling technique where the impact of the predictive variables is captured through smooth functions which—depending on the underlying patterns in the data—can be nonlinear:



We can write the GAM structure as:

$$g(E(Y)) = \alpha + s1(x1) + \cdots + sp(xp),$$

where Y is the dependent variable (i.e., what we are trying to predict), E(Y) denotes the expected value, and g(Y) creates the link function that links the expected value to the predictor variables x1, ..., xp.

Why use GAM?

- Easy to interpret.
- Flexible predictor functions can uncover **hidden patterns** in the data.
- Regularization of predictor functions helps **avoid overfitting**.

We don't have to know what type of predictive functions we will eventually need. From an estimation standpoint, the use of regularized, nonparametric functions avoids the pitfalls of dealing with higher order polynomial terms in linear models.

Fitting GAMs in R

The book code uses the "gam" function from **"mgcv"** library to fit a gam model. In the first function of the code "**plotrel"**, gam is used to produce a smooth representation of the data. Although, the processes don't seem to use this function.

- By considering the family binominal, we are in fact generalizing the parameters.
- The type="response" in **predict** tells R to output probabilities as opposed to other information such as the logit.
- Empirical estimation techniques are based on the data taken from the previous project and some based on guesses and assumptions.

What is MAE?

In statistics, **mean absolute error** is a measure of errors between paired observations expressing the same phenomenon. **MAE** measures the average of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j|$$

The function **"getmae"** first normalizes the given dataset, blocks the data into
b parts (a given number) to ease the process of calculating MAE and analyzing data.
And plots the results after getting the MAE.

AUC, a performance measurement.

"In Machine Learning, performance measurement is an essential task. So, when it
comes to a classification problem, we can count on an AUC Curve. When we need to check or visualize the performance of the multi-class classification problem, we use AUC (Area Under the Curve). It is one of the most important evaluation metrics for checking any classification model's performance."

It tells how much model is capable of distinguishing between classes.
Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.
It can be calculated by true and false positive and negative values.

Defining terms used in AUC

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

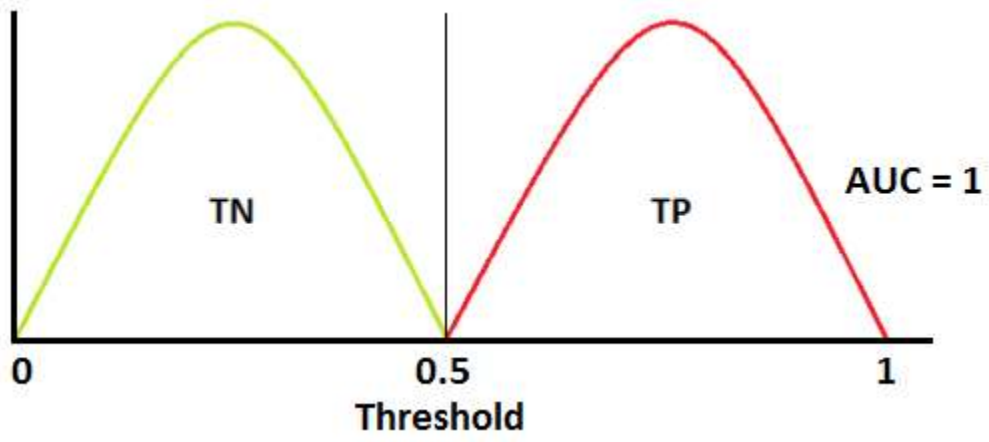
$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{FPR} = 1 - \text{Specificity}$$

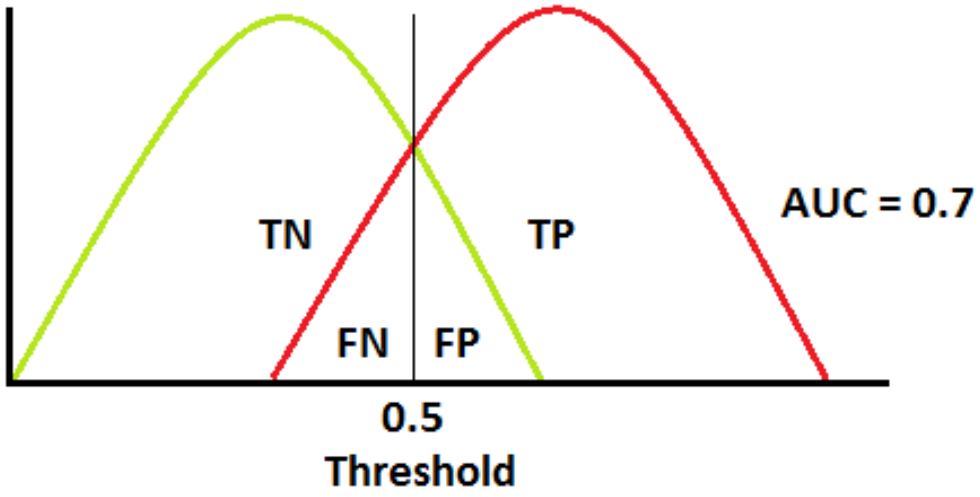
$$= \frac{FP}{TN + FP}$$

How to speculate the performance of the model?

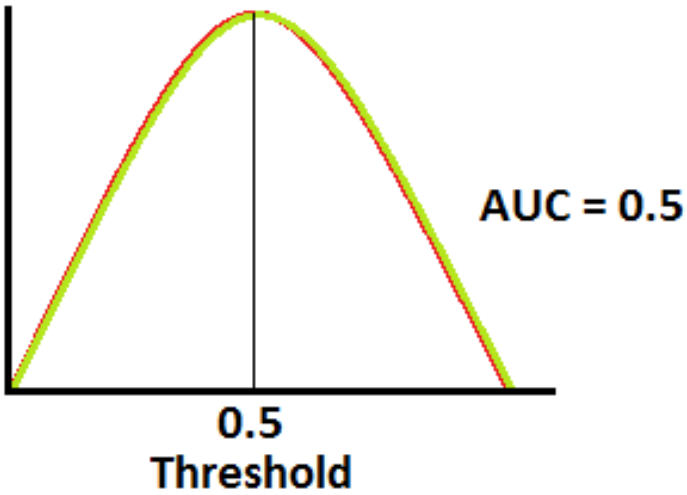
1. This is an ideal situation. When two curves don't overlap at all means model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class.



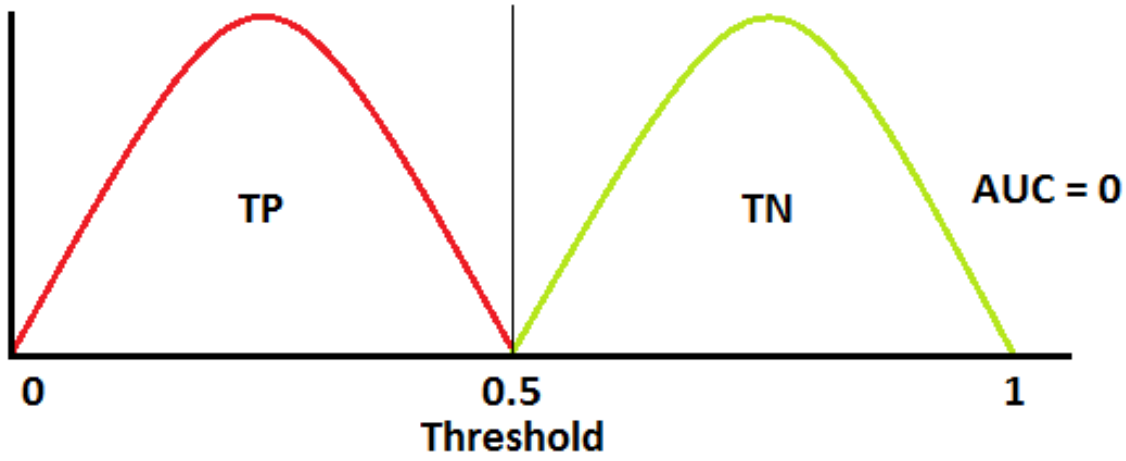
2. When two distributions overlap, we introduce type 1 and type 2 error. Depending upon the threshold, we can minimize or maximize them. When AUC is 0.7, it means there is 70% chance that model will be able to distinguish between positive class and negative class.



3. When AUC is approximately 0.5, model has no discrimination capacity to distinguish between positive class and negative class.



4. When AUC is approximately 0, model is actually reciprocating the classes. It means, model is predicting negative class as a positive class and vice versa.



AUC in R

The **"get_auc"** function's purpose is to evaluate the given model's performance using the **"rocr"** package.

K-fold Cross Validation

Cross-validation is a statistical method used to estimate the skill of machine learning models.

It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    - Take the group as a hold out or test data set
    - Take the remaining groups as a training data set
    - Fit a model on the training set and evaluate it on the test set
    - Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Fitting Process

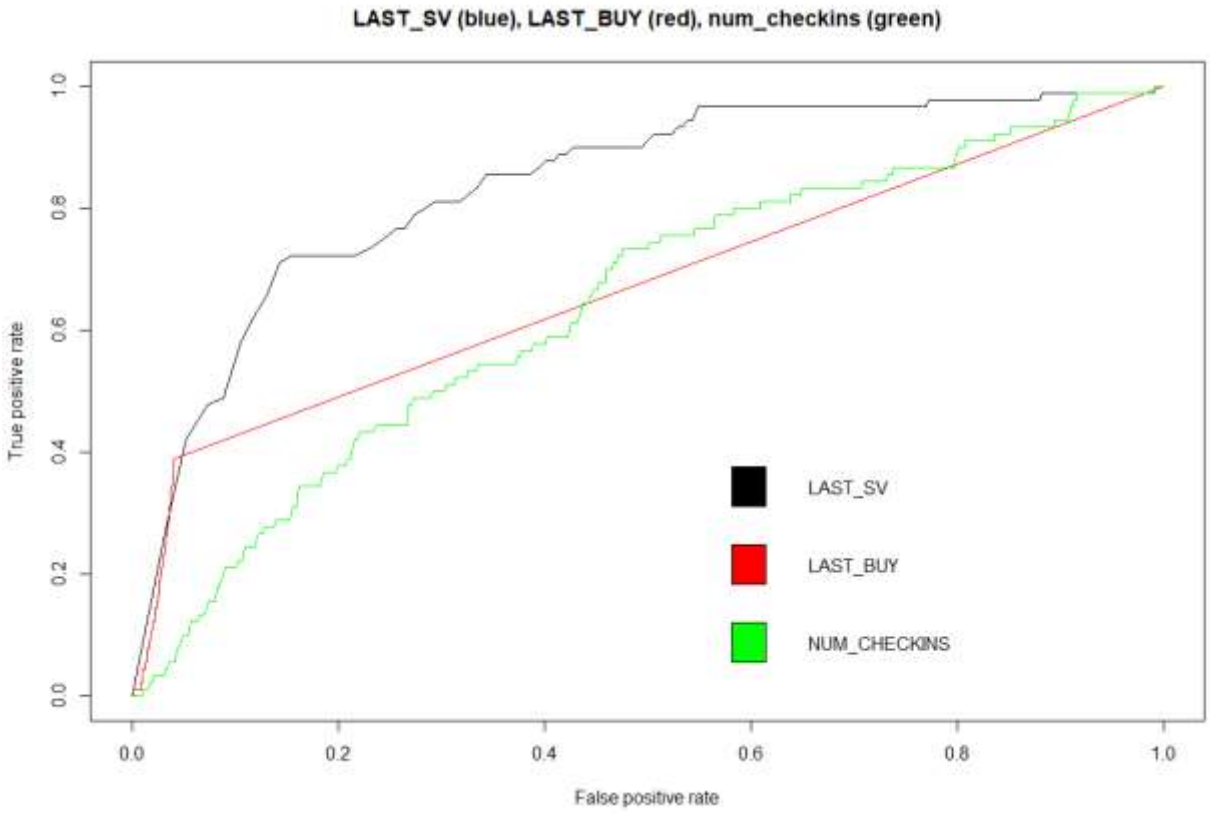
After analyzing the used functions, we need to look at the results of fitting different models on our dataset

By using GLM we fit a binominal model. There are some good features (the ones that have lower p-values) such as “AT_BUY_BOOLEAN” and “LAST_CV” and standard errors are pretty low.

After that we get each feature’s performance and save them in separate lists and combine them to a data frame. (univar) Then three different models are fitted based on three different features. Here’s the summary:

```
>
> fit <- glm(f, data = train, family = binomial(logit))
>
> summary(fit)

Call:
glm(formula = f, family = binomial(logit), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6970  -0.1003  -0.0697  -0.0374   4.2717

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -4.963e+00  1.404e-01 -35.338  < 2e-16 ***
AT_BUY_BOOLEAN     1.834e+00  2.826e-01   6.489 8.63e-11 ***
AT_FREQ_BUY        1.188e-01  1.057e-01   1.124  0.26089
AT_FREQ_LAST24_BUY -1.574e+00  6.881e-01  -2.288  0.02216 *
AT_FREQ_LAST24_SV   1.455e-01  2.841e-02   5.122 3.02e-07 ***
AT_FREQ_SV          2.822e-02  1.006e-02   2.805  0.00503 **
EXPECTED_TIME_BUY  -3.390e-03  5.639e-03  -0.601  0.54768
EXPECTED_TIME_SV   -7.145e-03  4.749e-03  -1.504  0.13245
LAST_BUY           -2.547e-03  3.495e-03  -0.729  0.46611
LAST_SV            -2.425e-02  3.760e-03  -6.448 1.14e-10 ***
NUM_CHECKINS        4.777e-05  5.873e-05   0.813  0.41599
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2080.1  on 35474  degrees of freedom
Residual deviance: 1747.8  on 35464  degrees of freedom
AIC: 1769.8

Number of Fisher Scoring iterations: 10
```

The performance of them is measured based on their prediction, and all three are plotted as below based on true positive and false positive:



It is clearer on the plot that the black line for model “LAST_CV” has the highest true positive rate and therefore, is a better model.

Sources

- https://multithreaded.stitchfix.com/blog/2015/07/30/gam/
- https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d
- https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
- https://machinelearningmastery.com/k-fold-cross-validation/