CURSO 19-20

Data Analytics with R MIBA, 2019

Challenge 2: Introduction to Probability Models

Gonzalo de la Torre
Rubén Coca

Predicting machine failure

In this exercise we will build a machine failure model. This model will be the basis for an EWS (Early Warning System) for Vendex that fires periodic alarms with a list of machines that are probably broken to send operators to repair them, hence boosting revenues from the previously "lost sales".

Vendex is currently recording whether the machine was broken by an indicator called **"failure" which is equal to 1 on the first sale of a machine after a broken period** and 0 otherwise.

The probability that a given machine is broken at a given time (that is, will end up the day without sales) depends conceptually on two main factors: a) The average daily sales of the machine and b) The hours since the last sale. We will consider both in our model in a combined fashion.

Follow the given steps in order to construct the model and provide well organized code with every step:

[3 points] Creating the relevant variable of the model:

1. Merge the transactional dataset with the machine failures data set setting failure variable to 0 when no failure is recorded
   Hint: The syntax to do merges is merge(dt1,dt2, by=c('field1','field2'),all.x=T) and to set to 0 try dt[is.na(failure),failure:=0]
2. In the transactional data table, create a variable called "last_vend" containing the timestamp of the previous sale of each machine
   Hint: Remember you can use the function "shift" once the data is ordered according to machine and date with function "order" i.e. dt = dt[order(x,y)] where x and y are column names of dt
3. Create a new variable in the transactional data table called "deltahours" containing, for every sale, the hours that passed since the last sale
   Hint: Check function "difftime"
4. Create an auxiliary data table called "machine_daily_average" with the average daily sales per machine. Use this auxiliary table to attach to every row of the transactional data table the the average daily sales per machine. You can do this by doing a merge.
   Hint: Check function merge again via help(merge)
5. Create a new variable called "delta" in the transactional data table containing a **normalized version of deltahours** consisting on the deltahours associated with each sale divided by the average deltahours of each machine i.e. delta = deltahours /(24/daily_sales_machine). The interpretation of delta is the amount of "missed sales" if the machine was selling at a constant rate

[2 points] Creating the model

6. Select 30% of the machines in the transactional data for testing and 70% of the machines for training and train a linear logistic regression model called "m" to **predict whether a machine has a failure as a function of variable delta**. What is the value of the intercept and the coefficient accompanying variable delta?
   *Hint: Recall the syntax for the linear model is glm(target~ variable,data,family='binomial'). You can select 70% machines out of a machine list v, for example, with function sample(v,round(0.7*length(v),0),replace=F). Use help(sample) for more information*

[5 points] Questions

Answer the following questions:

a. [1 point] What's the AUC, a measure of quality, of the model you have built on the train set? and on test set?

b. [1 point] Plot the function of probability of failure with respect to delta to gain intuition:

*Hint: You can plot a function easily with function curve i.e. curve(x^2+x,c(0,20) if the function was x^2+x. Remember the form of the logistic regression function is prob = 1/(1+exp(-(intercept + coefficient*delta))).*

c. [1 point] Let us create alarms with two levels of priority: med-risk and high-risk. Med-risk alarms will fire when the probability of failure is >=60% and High-risk when that probability is >=80%.
   i. What are the threshold deltas for each type of alarm to fire?
   Hint: You can find it in several ways:
      - *Trial and error making the model predict the failure probability on one instance with different values of delta i.e. predict(m,data.table(delta=10),type='response')*
      - *Visually with the plot of exercise (b)*
      - *(More precise) finding the "root" of the relevant function i.e. p(delta)- 0.6 for the med-risk alarms where p(delta) is the logistic function. Function uniroot from library "stats" can help you: uniroot(f, c(0,20))$root where f needs to be defined before i.e. f = function(x){x^2+x}*

   ii. How many of these alarms would be fired per day on average according to your model?
   Hint: Divide the number of alarms (transactions where the delta exceeds the threshold delta for each type of alarm) by the total number of days
   iii. What % of these will be "false alarms" i.e. failure variable is equal to 0, for each level of priority?

d. [2 points] In this exercise we will estimate **the profit impact of our EWS system vs the current system**:
   i. If we set the EWS only with the med-risk alarms, what is the annual profit we will generate vs the current system as a % of the total profit? [For simplicity, consider the total profit to be the margin per item times the number of items in the period]
   ii. And if we set the EWS only with the high-risk alarms?

   Assumptions:
   - The time to go and fix a machine is 1.5 hours after an alarm is fired
   - The cost of an operator checking if the machine is broken is 10 euros (transport + checking). You will use this to calculate the cost of "false alarms" with our system
   - Do not consider any cost of repairing the machine since this cost needs to be done also with the current system
   - Vendex reports that their current system generates 2.2 false alarms per machine and year
   - Every item sold leaves 1.7 euros of margin on average. You will use this to calculate the impact of the "saved sales" thanks to our new EWS system

Hints:
1. *To start, you will need to select only the transactions that would have generated an alarm according to the version of the EWS i.e. dt_alarm = dt[delta>delta_threshold]*
2. *You can find out the amount of hours without sales corresponding to that threshold delta for each machine ("threshold_hours") multiplying the threshold delta by the "conversion factor" from deltas to hours which is 24/daily_sales*

3. *You will need to add 1.5 hours of repairing time to that threshold hours creating "threshold_hours_fixed" and convert that back to a delta i.e. "delta_fixed" multiplying by the conversion factor from hours to delta 1/(24/daily_sales)*
4. *For each transaction you can create a variable called "won_sales" multypling the binary failure variable by (delta – delta_fixed). Recall that delta has the interpretation of "missed sales". Note that if the current system was able to detect and repair the machine before our EWS system, the won_sales will be negative (actually we lost sales).*
5. *Additional revenues will be the sum of won_sales times the average margin of every sale*
6. *Cost of the system is the cost of sending an operator to check times the number of "false alarms", that is, failure is equal to 0.*
7. *The profit increase as a % of the total profit can be estimated as the profit increase / (margin_per_item * number_of_transactions)*

Additional information

Advanced plotting
If you want to do more advanced plots you can use ggplot2. For that, install ggplot2 using install.packages('ggplot2') and use it via library(ggplot2). Check here how to plot functions https://kohske.wordpress.com/2010/12/25/draw-function-without-data-in-ggplot2/