Lessons Learned Report - Course 3, Task 5

Blackwell Electronics Investigation

Primary Objective:

- Collect and document learned knowledge and technical pitfalls for the data mining tasks performed for Blackwell Electronics

C3T2 Lesson: Recursive Feature Engineering can significantly decrease compute times, and similar or better model performance

Problem:

As the number of attributes increases, model generation takes significantly more compute time, as the models work with many more available inputs to tune.

Advice:

RFE gives a simple way to show the relative importance of variables to model generation. Generally, it shows a very rapid decline after the first several variables, indicating that a relative few drive the majority of the model decisions. By eliminating variables without significant importance, the inputs to the model are dramatically reduced, resulting in faster compute times. This could be especially important for compute resources as a new model is deployed for ongoing commercial use.

C3T3 Lesson: Sometimes eliminate columns, not rows, when dealing with missing data

Problem:

When working to eliminate N/A values from our data set before modelling, it is most common to remove entire rows having any N/A values. However, with limited data sets, this can remove too much of the available data for modelling.

Advice:

If N/A or NaN values are isolated to a particular attribute (column), it can be far more efficient to eliminate that singular column, enabling the data set to not lose any rows in the process. This is particularly true of there are a large number of attributes, as a singular attribute is likely less important than providing more rows to the model for training.

C3T3 Lesson: When the training data set is small, big differences can occur during train/test versus when applied to new data sets.

Problem:

In this mining exercise, there only were 80 distinct rows of data, each describing a specific product. With 75% of those values going to the training set, the folding model creation was able to create models with very high R^2 values. However, when these models are applied to the 20 data points reserved for the test set, wildly different fit quality metric values can occur.

Advice:

Due to the limited number of data points in the problem, it can be beneficial to not save and apply only 1 model to the test data, but run multiple leading models to determine which works best on the test data. This can help reveal if some models are significantly overfitted to the training data.

Another tool to investigate this is to use attributes of the new (unknown) data to be fit. In this case, we knew sales volumes could not be below zero, so models returning negative values showed signs of not being as accurate as advertised.

C3T3 Lesson: Use plot() to overlay the test data from various model fittings

Problem:

With limited data sets, it's hard to get the whole story of the model fitting from statistics like R^2 and RMSE.

Advice:

Use the plot(), points(), and abline() commands to plot the actual test values of the dependent variable versus the predicted values from various models. With this, we can see if just one or two data points are significantly degrading the prediction summary statistics. It can also show if some models work better for one region of the dependent variable spectrum – it may not always be a singular model that works best, but instead multiple models working together to predict the outcome most accurately.

C3T4 Lesson: Support values must scale with the number of discrete items in the data set

Problem:

Support represents the frequency of a given rule occurring within a data set, but as the number of items and item combinations increase, the frequency becomes low even for the most popular items

Advice:

Investigate item frequency to evaluate the relative frequency of each item (not rule) as a percent of the total number of transactions. This gives a high-level metric of how many transactions must occur for an item to even arise once. Support levels above this value are very unlikely to yield new rules, as just the likelihood of the item being present in the transaction is very low (much less developing a strong dependency with another item).

It's very important to note that support is a relative percentage – so count is more intuitive for understanding how often a rule actually occurs within the data set. Count can be a much better guide to whether a rule has enough instances to be statistically significant.

C3T4 Lesson: Back-test items from the LHS of rules when looking for high-lift rules

Problem:

In general we work to create rules with high support, confidence, and lift, but the flip-side of some of these rules can give useful results that can be missed.

Advice:

When filtering to create a manageable number or rules from a data set, it's not uncommon for the highest volume products to produce the most rules, as they are present in a higher number of the

transactions. As we prune our rule generation to produce handfuls of meaningful rules instead of thousands, support and confidence are generally increased in the input parameters.

One useful trick is to look at the LHS items for the high support/confidence rules, then isolate the most common LHS items, and use as RHS items. In other words, we can force new rules to be generated using these items as a RHS result. The outcome from this is that although there is less overall support for these rules (since LHS items are generally less common), they can demonstrate very high lifts and confidence values. They also show a new unique linkage with the most common items appearing on the LHS of these new rules. Although they have overall less support, the high lift can indicate a very strong relationship that can yield business insights.