Kaggle Santander Transaction Prediction

Belinda Trotta

This challenge requires us to predict whether customers will make a particular type of transaction, given 200 unlabelled continuous variables.

1 The magic

1.1 Feature independence

It was discovered in some excellent early kernels that the features seem to be independent given the target (see kernel and discusssion by Branden Murray [1], [2] and kernel by Chris Deotte [4]). So it appears likely that each feature column has been independently "scrambled" within each target class (0 or 1). This means there's no point looking for feature interactions among the columns.

1.2 Synthetic test data

The kernel by YaG320 [3] shows that some of the test data contains no unique feature values in the whole row, suggesting that it has been synthesised from the test of the data.

1.3 Repeated values

Repeated values seem to show a stronger signal, as shown in Figure 1.1. I wasn't able to figure out why this is, nor whether it's a real effect or an artifact of the way the data is prepared. But this is the key to the "magic" feature engineering. For each column, we add a feature counting the number of appearances of the value in the train and test sets. However, it's crucial to exclude the synthetic test data from this count. Again, I haven't really figured out why this works. I tried calculating the counts separately for train and test, and excluding the synthetic test data from the test counts, but although this gave good cross-validation results in training, it didn't seem to work on the test set. So that suggests that the combined train and test count leaks some information from test to train.

2 Modelling approach

We model each column separately then combine the predictions. Each column prediction is a blend of 2 lightgbm models and one modified logistic regression.

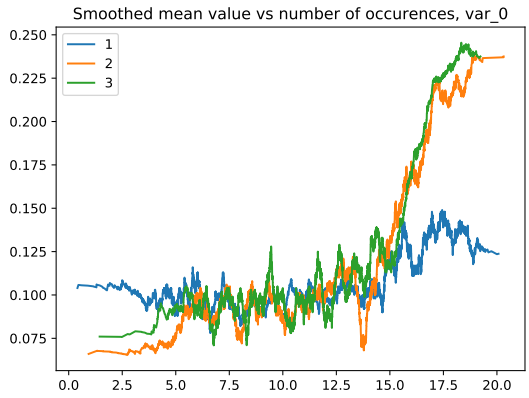

Figure 1: Code to generate the graphs is in `plot_graphs.py`.

2.1 Lightgbm 1

The model has 2 features, the column value and the frequency count. Maximum depth is 2. We choose the number of iterations separately for each column by cross-validation (since some columns have much more signal that others).

2.2 Lightgbm 2

The same as the first model, but instead of the raw frequency count, we used the normalised count, defined as the count divided by the average count of neighbouring points.

2.3 Modified logistic regression

We separate the data into 2 sets, frequency 1 and frequency > 1 and fit a separate model for each. In each model, we sort the training data by the value of the column, and divide it into (overlapping) blocks of 20000 samples with a step size of 500. For each block we fit a logistic regression with the single variable c, the column's value, and evaluate it at the midpoint of the column's c values. This gives us a prediction of the target for a subset of evenly spaced values of c. Then we linearly interpolate between these. The idea is similar to the Savitzky-Golay filter, but fitting a logistic function instead of a polynomial. Surprisingly, this gives results almost as good as the lightgbm models.

3 Combining the predictions from different columns

Chris Deotte's naive Bayes kernel [4] shows that the features are independent within each target class. The approach in the kernel is to multiply all individual column probabilities. However, Julian points out in the comments that this makes the additional

assumption that the individual columns are completely independent, not just conditionally independent given the target class.

Here I show how to optimally combine the column probabilities in a way that requires only the conditional independence. (I'm sure this is not original, but I couldn't find a thorough explanation elsewhere.)

By Bayes' theorem and the conditional independence assumption, we have

$$\begin{aligned}
P(y = 1 \mid x_0, \ldots, x_{199}) &= \frac{P(x_0, \ldots, x_{199} \mid y = 1)P(y = 1)}{P(x_0, \ldots, x_{199})} \\
&= \frac{P(x_0 \mid y = 1) \times \cdots \times P(x_{199} \mid t = 1)P(y = 1)}{P(x_0, \ldots, x_{199})} \\
&= \frac{\frac{P(y=1|x_0)P(x_0)}{P(y=1)} \times \cdots \times \frac{P(y=1|x_{199})P(x_{199})}{P(y=1)} P(y = 1)}{P(x_0, \ldots, x_{199})} \\
&= \frac{\prod_{i=0}^{199} P(y = 1 \mid x_i)P(x_i)}{P(y = 1)^{199} P(x_0, \ldots, x_{199})}
\end{aligned}$$

This expression isn't very useful for predicting $P(y = 1)$, since we'd have to estimate each $P(x_i)$, and the joint probability $P(x_0, \ldots, x_{199})$.

However, we will now see that if we instead predict the logit of $p = P(y = 1)$, that is, $\log(p/(1 - p))$, then we can avoid having to calculate these. By symmetry with the above calculation, we have,

$$P(y = 0 \mid x_0, \ldots, x_{199}) = \frac{\prod_{i=0}^{199} P(y = 0 \mid x_i)P(x_i)}{P(y = 0)^{199} P(x_0, \ldots, x_{199})}$$

Hence

$$\begin{aligned}
\frac{p}{1 - p} &= \left(\frac{\prod_{i=0}^{199} P(y = 1 \mid x_i)P(x_i)}{P(y = 1)^{199} P(x_0, \ldots, x_{199})} \right) \div \left(\frac{\prod_{i=0}^{199} P(y = 0 \mid x_i)P(x_i)}{P(y = 0)^{199} P(x_0, \ldots, x_{199})} \right) \\
&= \prod_{i=0}^{199} \left(\frac{P(y = 1 \mid x_i)}{P(y = 0 \mid x_i)} \right) \times \frac{P(y = 0)^{199}}{P(y = 1)^{199}}
\end{aligned}$$

Taking logs of both sides, we get

$$\log\left(\frac{p}{1 - p}\right) = \sum_{i=0}^{i=199} \log\left(\frac{P(y = 1 \mid x_i}{P(y = 0 \mid x_i)}\right) + 199P(y = 0) - 199P(y = 1)$$

Thus the logit of the target is the sum of the logits of the individual column predictions, plus a constant which is the same for each row. Since we are using the roc-auc metric, we can ignore the constant.

References

[1] `https://www.kaggle.com/brandenkmurray/randomly-shuffled-data-also-works?scriptVersionId=11467087`

[2] https://www.kaggle.com/c/santander-customer-transaction-prediction/discussion/83882

[3] https://www.kaggle.com/yag320/list-of-fake-samples-and-public-private-lb-split?scriptVersionId=11948999

[4] https://www.kaggle.com/cdeotte/modified-naive-bayes-santander-0-899?scriptVersionId=11969430