P2R1.0: Evaluation of VSM on Xeeva Customer data

Koushik Kumaraswamy

Wednesday, Mar 19, 2015

Introduction

In this report, we build on the previous analysis [P1R1.0] by evaluating VSM purely on Xeeva Customer data. The goal of this analysis is to observe the changes to VSM performance baselines as it operates on *less than rich* item data .

Design of Experiment

As before, this experiment was modeled after a traditional machine learning experiment by treating VSM as a model for classification . The details of this analysis experiment, designed per reproducible research standards, is outlined below:

On model formulation

The classification schema used in this analysis was Xeeva Schema with the following classification groups: Category and Sub-Category

Aside from the above difference, the model formulation is identical to the one employed in [P1R1.0].

Raw Data acquisition and cleansing

Raw Data for this analysis came from the input file that Xeeva Spend Analytics teams used for creating value charts for our largest customer. The data itself was provided in a .xlsb file with several fields. A copy of this file was made for this analysis and the irrelvant fields were cleansed out of the working file as noted in the table below.

```
##      Original.File.Columns Removed.in.PreProcessing.
## 1              REGION_NAME                       Yes
## 2             COUNTRY_NAME                       Yes
## 3             LOCATIONCODE                       Yes
## 4            LOCATION_NAME                       Yes
## 5                 DIVISION                       Yes
## 6              CREATEDDATE                       Yes
## 7             CREATED_YYYY                       Yes
## 8           CREATED_YYYYMM                       Yes
## 9         CREATED_YYYYMMDD                       Yes
## 10               TYPE_NAME                       Yes
## 11           CATEGORY_NAME                        No
## 12        SUBCATEGORY_NAME                        No
## 13             PRODUCT_SKU                       Yes
## 14            PRODUCT_NAME                        No
## 15             PRODUCT_MPN                        No
## 16                MFG_NAME                        No
```

```
## 17  ACTIVE_INACTIVE_STATUS                    Yes
## 18                   PRICE                    Yes
## 19         UNIT_OF_MEASURE                    Yes
## 20                CURRENCY                    Yes
## 21 BASELINE(INITIAL)_PRICE                    Yes
## 22          YTD-2014-SPEND                    Yes
## 23      UOM_YTD-2014-SPEND                    Yes
## 24            FILE_CREATED                    Yes
```

In analyzing the raw data file, 60 unique Category values were found and 497 unique category and subcategory combinations were found. These unique classifications were codified seperately in a 6 digit scheme with 3 digits each denoting the category and subcategory respectively. The codified representation of the category as well as category-subcategory combination was copied back into the data file. The textual descriptions of the category and sub-categories were then removed from the file. This file was saved as in CSV format and used for downstream analysis.

Data pre-processing

The modified input file was read into working memory as a dataframe object. The Raw Data fields viz. Item Name , Item MPN, and Manufacturer Name were merged into a large text field that represented a single item within the search space.This data was then segmented into groups of similar categories (based on selected classification granularity).

Training and Test set segmentation

The previous step yielded a collected of objects, each of which denoted a set of items belonging to the same group. These items were randomly distributed between a training set and a test set in the ratio of 80% to 20% . Groups that did not have a defined minimum number of records, were excluded from the downstream analysis. The intent of this step is to train the model on the *training set* records and to evaluate its performance on the *testing set*.

Training the model

For VSM, training involved the creation of set of artificial documents, each representing a group within the selected classification hierarchy. A Corpus comprising of these artificial documents was then created and normalized through various tranformations (viz. conversion to lower case, stripping white space and removing punctuation). Stemming was not performed as part of this analysis. A term document matrix was then created using the transformed corpus. This matrix and its associated dictionary represents the trained model for this analysis.

Testing the model

To get around computing power constraints of the analyst's computer, a test bed was created using a random sample of records from the testing data set. The constraint imposed on the sampling was to limit each item group's representation to at most 10 instances inside the test subset. Each item in the test bed was presented to the model for classification and the result recorded. As referenced in the formulation section, the output classification from the model corresponds to group that has the largest value from the dot product between query and group vectors. The performance of the model was quantified through a simple accuracy measure defined as [(# of correctly classified tests)/(# of total tests)].

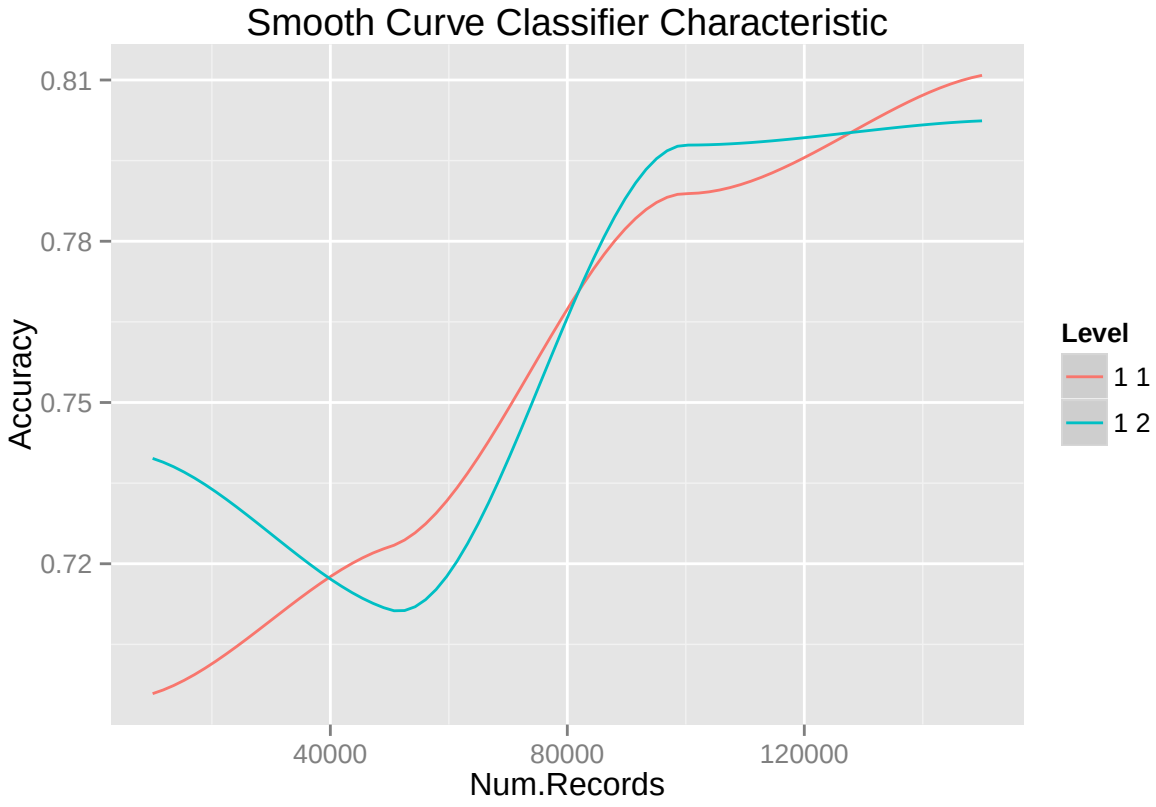
Run structure

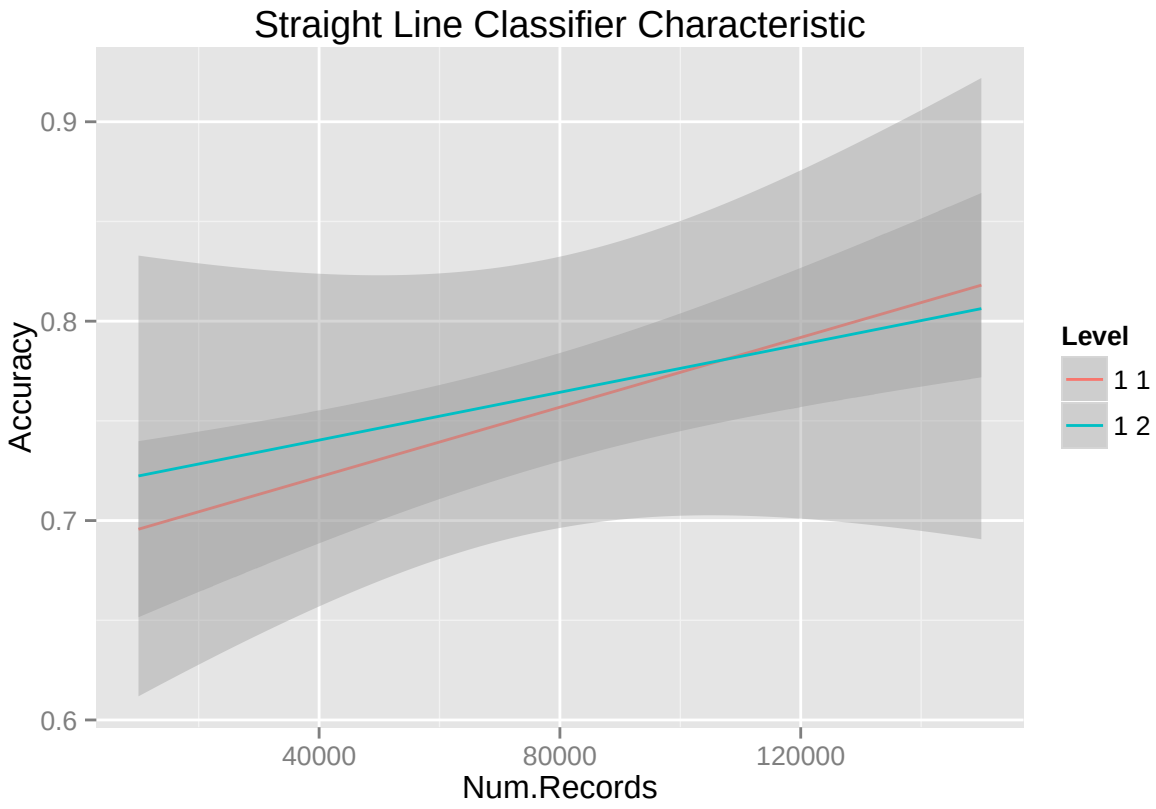
The training and testing of the VSM model was performed in sequence for each classification level (i.e. category and Sub-category) with increasing record counts. As the number of target groups in the data increased (specifically with larger record counts), the minimum records threshold was correspondingly increased to maximize the model's chance to *learn from the data*.

Run Results and analysis

The result of the various runs are tabulated below:

```
##   Level Num.Records Num.Groups Min.rows.per.group Accuracy Time.of.run
## 1   1 1       10000         27                  20   0.6958      0.4483
## 2   1 1       50000         44                  20   0.7232      3.6363
## 3   1 1      100000         45                  20   0.7889      5.9303
## 4   1 1      150000         48                  20   0.8109      9.9220
## 5   1 2       10000         65                  20   0.7396      0.9174
## 6   1 2       50000        178                  20   0.7115     15.7229
## 7   1 2      100000        238                  20   0.7979     32.6290
## 8   1 2      150000        260                  20   0.8024     55.3086
```


Smooth Curve Classifier Characteristic
0.81
0.78
0.75
0.72
Accuracy
40000
80000
120000
Num.Records
Level
1 1
1 2


Straight Line Classifier Characteristic
0.9
0.8
0.7
0.6
Accuracy
40000
80000
120000
Num.Records
Level
1 1
1 2

From the graph above, it can be observed that the higher level classifier uniformly performs better with increase in size of training data sets. The more granular classifer follows the same pattern but with lot more volatility of direction.

Conclusion and next steps

VSM trained on Xeeva Client data seems to improve its performance progressively with increase in training set size. This finding is in marked contrast to [P1R1.0], where VSM performance deteriorated with increase in training set size.

On its own, VSM performance seems to perform better than the 70% acceptability threshold if training data sets are greater than or equal to 50000. Subsequent analyses will need to quantify performance when VSM is employed on a combination of rich and real-life data.