Paper 881-2017

Fuzzy Matching and Predictive Models for Acquisition of New Customers

Krzysztof Dzieciolowski, Daniel Marinescu
John Molson School of Business, Concordia University

ABSTRACT

Acquisition of new customers is fundamental to the success of every business. Data science methods can greatly improve the effectiveness of acquiring prospective customers and can contribute to the profitability of business operations. In a consumer setting, a predictive model may be used to acquire a new donor for a charity or a new customer for a firm, for example by switching to another competitor. In the B2B domain, a predictive model may target business prospects as individual firms listed by an external database. A typical acquisition model can be defined using a binary response where the values categorize a firm's customers and non-customers, for which it is then necessary to identify which of the prospects are actually the customers of the firm. The methods of fuzzy logic, for example, based on the similarity distance between the strings, may help in matching customers' names and addresses with the overall universe of prospects. However, two errors, which affect model selection and performance, may occur: false positives – when the prospect is incorrectly classified as a firm's customer; and false negatives – when the prospect is incorrectly classified as a non-customer. In current practice of building models, these errors and their consequences are ignored. In paper, we assess how misclassification matching errors affect the performance of the predictive model for varied levels of confidence of matching algorithms. In the example considered, we discovered that the conservatively-matched model outperforms the liberally-matched model in their own-matched universes. We suggest a heuristic approach to verify models' performance based on different matching confidence levels. The analysis is illustrated with actual data and models, and accompanied by elements of SAS code used in the research.

1. INTRODUCTION

Models form and its performance are affected by matching algorithms of the underlying data. In paper we suggest how such an impact can be assessed. The matching algorithms with their relative strengths and weaknesses are described in the section 1. Our study design for constructing a predictive acquisition model is given in section 2. The approach of evaluating models performance in their own- and alternate-matched universes is described in section 3. There, we also discuss a pseudo paired-t statistic to compare lifts of models constructed at varying level of matching confidence. An illustrative example and review of the results. Finally, in section 4 we provide a pseudo-code used in the analysis.

2. MATCHING ALGORITHMS

SPEDIS, COMPGED AND N-GRAM

Several algorithms exist to compare a pair of strings, which attributes a value to the similarity or dissimilarity between them. Such matching algorithms include:

- **SPEDIS**: This function calculates the number of character transformations required to transform one character string into the other. Each type of transformation is assigned a specific 'cost'; e.g., character swap=50, character replacement=100. The incremental costs are summed and divided by the query length to generate a value, which is a measure of the dissimilarity between the two strings. Interpretively, the smaller this value, the larger the similarity between the two strings.
- **COMPGED**: In a similar way as SPEDIS, this function calculates a generalized edit distance between two strings. This is a generalization of the Lewenshtein's edit distance, which is the number of deletions, insertions, and replacements of single characters that are required to convert one string

into another. As with SPEDIS, this function also generates a measure of dissimilarity between two strings.

SPEDIS and COMPGED functions can often be coupled with SOUNDEX or UPCASE/COMPRESS to standardize the strings that are being compared (StringA and StringB)

```
output=SPEDIS(SOUNDEX(stringA),SOUNDEX(stringB)
output=SPEDIS(UPCASE(COMPRESS((stringA),,'kan')),UPCASE(COMPRESS((stringB),,'kan')))

output=COMPGED(SOUNDEX(stringA),SOUNDEX(stringB));
output=COMPGED(UPCASE(COMPRESS((stringA),,'kan')),UPCASE(COMPRESS((stringB),,'kan')));
```

- **N-GRAM :** An N-gram algorithm typically searches for pattern similarities between two strings. There are many combinations of characters that could serve as patterns, however the typical method is to compare adjacent character pairs between two strings. Both strings intended for comparison are decomposed to their 2-character substrings (excluding for spaces and duplicate characters). Decomposition of strings via N-gram algorithm:

BEN SHEPPARD	→	BE	EN	NS	SH	HE	EP	PA	AR	RD		
BENNIE SHEPERD	→	BE	EN	NI	IE	ES	SH	HE	EP	PE	ER	RD

DeShon & Vetmedica have outlined a basic method to formulate an N-gram algorithm. The code below has been inspired by the former and comments have been added to outline each significant step:

```
*Standardize the compared strings via UPCASE + COMPRESS;
stringA = UPCASE(COMPRESS((stringA),,'kan'));
stringB = UPCASE(COMPRESS((stringB),,'kan'));

*Set initial similarity count to 0;
count = 0;

*Execute a do loop which counts the number of adjacent character pairs from
string A which exist in string B (i.e A-to-B comparison);
do i = 1 to (LENGTH(stringA)-1);
   if FIND(stringB,SUBSTR(stringA,i,2)) then count + 1;
end;

*Execute a do loop which counts the number of adjacent character pairs from
string  B which exist in string A (i.e B-to-A comparison). Note that this
do loop continues to add to the similarity count from the prior do loop;
do i = 1 to (LENGTH(stringB)-1);
   if FIND(stringA,SUBSTR(stringB,i,2)) then count + 1;
end;

*The average count of the A-to-B and B-to-A comparisons is calculated by
dividing the count by two. This average count is normalized against the
total possible number of matches to generate an output which represents a
percentage score of similarity;
output = ((count/2) / (MAX(LENGTH(stringA)-1,LENGTH(stringB)-1)));
```

Sloan and Hoicowitz (2016) reviewed fuzzy-matching algorithms available in SAS and suggested using COMPGED along with character handling functions.

COMPARING ALGORITHM RESULTS

We can create three scenarios with distinct cases of comparisons; 1) Pair of similar strings with word order rearrangement, 2) Pair of similar strings with character differences, 3) Pair of similar strings with *both* characters differences and word-order rearrangement.

Scenario	String A	String B	SPEDIS with SOUNDEX	COMPGED with SOUNDEX	N-GRAM with UPCASE/COMPRESS
1	O'Connell, Francis B.	Francis B. O'Connell	68	690	0.93
2	Frank O'Konel	Francis B. O'Connell	10	200	0.4
3	O'Connell, Francis B.	Frank O'Konel	73	770	0.4

Table 2. Output results of matching algorithms

The results shows that the best (lowest) SPEDIS and COMPGED scores are found in the pair with character differences (scenario 2). Conversely, the N-gram algorithm showed the best (largest) score for the pair with word-order difference (scenario 1). The results of scenario 3 follow logically, as all algorithms show their relatively worst scores for the pairs with both character differences and word-order rearrangement.

RELATIVE STRENGTHS AND WEAKNESSES

- **Effect of word order:** SPEDIS and COMPGED are based on attributing a value to an editing distance (measure of dissimilarity), while N-gram algorithms are based on attributing a score to the number of matched character combinations (measure of similarity). As a result, SPEDIS and COMPGED are very vulnerable to word order, as reordering a given string may require multiple 'deletion' and 'insert' operations. Since these operations are associated with relatively high cost in terms of editing distance, similar strings with varying word order may results in artificially large SPEDIS or COMPGED values. As N-gram algorithms make substring comparisons which are independent of order, they are less affected by this.

- **Effect of character differences:** N-gram algorithms may be more vulnerable to character differences than SPEDIS and COMPGED. Generally, SOUNDEX can assist in reducing this effect by attenuating the impact of incorrect/missing vowels or differences among phonetically similar consonants.

- **Processing speed:** *DeShon et Vetmedica* have reported that the N-gram algorithm showed lower average run time as compared to SPEDIS and COMPGED. In turn, the time cost of SPEDIS and COMPGED is mitigated by certain advantages that they retain; when comparing matching algorithms which use edit distance, *Roesch* has reported that SPEDIS is generally more effective at matching longer strings, while COMPGED is more flexible (e.g; option available to set a cutoff value).

- **Normalized scoring:** N-gram algorithms provide a similarity value which can be weighted against the lengths of the compared strings. Therefore, unlike SPEDIS and COMPGED, N-gram algorithms have the advantages of providing an intuitive 'percent similarity' score.

3. STUDY DESIGN

OUR MATCHING APPROACH

Our practical matching approach is based on matching records between two datasets (S1 and S2) according to both their name fields and address fields. This is done through step-wise procedure, involving the following stages:

- First, records intended for comparison are matched in the small geographical areas. This reduces the Cartesian product of records to be matched to only groupings of records within realistic geographic proximity. Therefore, records which are very geographically distant are not compared, which significantly reduces the computational burden of conducting a full match of the two datasets. While this geographical restriction may somewhat reduce accuracy of matching, it is preferred, as it is assumed that the computational costs of full Cartesian products is time consuming when merging, for example, individuals, households or businesses.
- Next, records are compared according to numeric matches in their respective address fields. This step acts as a filter. If any numbers that are contained within the compared records result in a match, this matching couple proceeds to the next phase of matching. If no numeric match is found between any portion of the address fields, this matching couple does not pass to the next phase of matching.
- In the third step, both the names fields and address fields are decomposed into their individual words, and paired. Pairs of words from the name fields are matched, and similarly for the address fields. The matching process is done via a COMPGED procedure, with a threshold value 'c' previously chosen. Any pair with a COMPGED result below this threshold is considered a positive comparison. A ratio of positive hits is calculated separately for the name field comparison and the address field comparison (ratio defined as # positive comparisons / total # comparisons). If this ratio is above a threshold for either the name field or address field, the pair of records moves on to the next phase of matching. If not, the pair of records does not move on to the next phase of matching.
- By this point, a single record from the source dataset S1 may have been matched with multiple records from the target dataset S2. For each pair, the name and address fields are concatenated and compressed. We run a N-gram procedure on the concatenated fields for each of these pairs, and select the pair with the maximum score as the final positive match.

MATCHING ALGORITHM IN PREDICTIVE MODELLING

Predictive modeling has become a cornerstone of modern analytical methods used by the industry to gain future insights and better allocate value-generating efforts. The practical application of our matching algorithm for predictive modeling is done in the context of predicting a binary response variable (0 or 1) from a dataset that is results from a matching process between two different datasets. As previously shown, the matching process can be done with varying levels of confidence, by raising or lowering the COMPGED threshold value 'c' needed to qualify as a match. The different levels of matching confidence between the two datasets thus results in varying levels of confidence in the response variable (0 or 1) and different models altogether.

SAMPLING AND TARGET

Our study investigates the effect of varying matching confidence levels on performance of predictive models in the context of acquiring new customers. New customers can be identified by their addition to a company's internal database over a period of interest. Such events can be recognized as new matches between the company's internal data against an external dataset. Activations that are not matched to the external data are purged from the analysis and model training sample altogether. An external data contains a relatively exhaustive universe of all prospects as well as explanatory variables associated with these customers.

At time t_0, a matching procedure between these two datasets will generate both matched records and non-matched records. The non-matched records at t_0 become the baseline records for the study. At later time t_1, the matching procedure is repeated and a portion of the previously non-matched records will now register as matching. These newly-matched records will serve as the 1s (positive outcome, i.e. new customers), while the records which remained unmatched will serve as the 0s (negative outcome, not

new customers). This exercise can be repeated at multiple COMPGED thresholds ‘c’, which will result in various degrees of confidence in the outcomes.

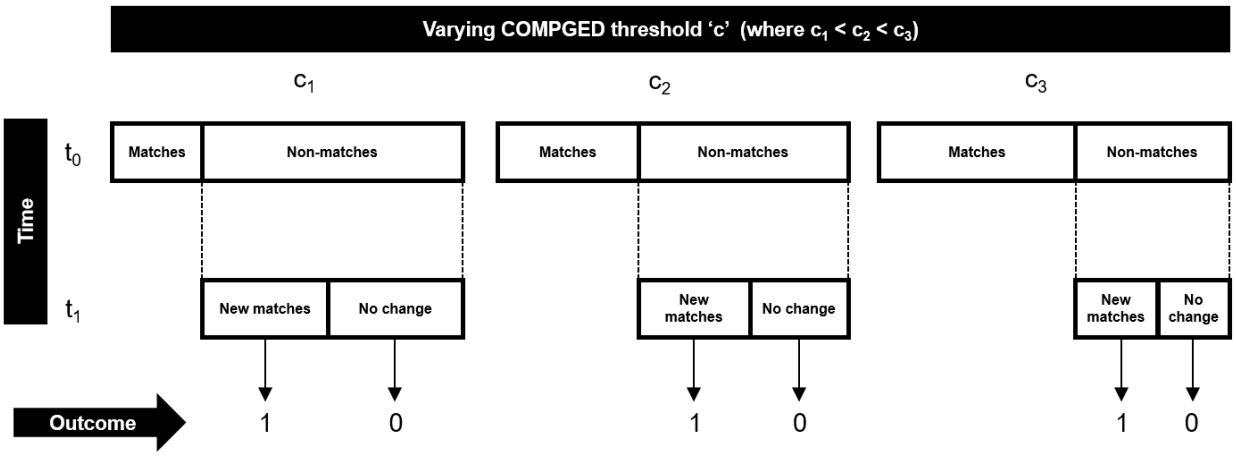


Table 3. Design of a training sample for an acquisition model.

Lower values of ‘c’ result in more conservative matches (higher fidelity matches); i.e. higher true positive rate of matches, but also lower true negative rate of matches. Higher values of ‘c’ results in more liberal matches (lower fidelity matches); i.e. lower true positive rate of matches, but also higher true negative rate of matches.

There are as many model training samples as confidence levels chosen to run the matching algorithm. For simplicity of the exposition, we compare only two matching confidence levels, c_1 and c_2, with $c_1<c_2$, leading to the respective Conservative and Liberal matches of the internal and external datasets at times t_0 and t_1. This design can be simplified to one match only at the time t_1 if the date of activation for a new customer is available in the internal database and, if we assume that the activation does not change the external profile of a new customer. Both conditions seem to be reasonable in practice. For example, a donor to a charity at the time t_1 is unlikely to have had changed, say, his or her occupation or address of residence in the period $[t_0, t_1]$ as a result of the donation.

4. PREDICTIVE MODELS IN FUZZY-MATCHING

As a result of matching internal and external data for two confidence levels, c1 and c2, with $c_1<c_2$, we obtain conservative and liberal-matched universes, respectively, as shown in Tables 4 and 5.

Conservative universe		Matched Activations		
Confidence level c1	Actual Activations	Yes	No	total
	Yes	TP1	FN1	AP1
	No	FP1	TN1	AN1
	total	PP1	PN1	N

Table 4. Conservative-matched universe at confidence level c1

Liberal universe		Matched Activations			
Confidence level c2	Actual Activations		Yes	No	total
		Yes	TP2	FN2	AP2
		No	FP2	TN2	AN2
		total	PP2	PN2	N

Table 5. Liberal-matched universe at confidence level c2

Given that $c_1<c_2$, we find that true positives and false positives of the conservative-matched universe are the subsets of their respective classifications in the liberal-matched universe, and consequently: TP1<TP2, FP1<FP2. Similarly, TN1>TN2 and FN1>FN2. Of course, the number of actual positives (AP=TP+FN) and actual negatives (AN=FP1+TN1), although unknown, remains the same, hence AN1=AN2 and AP1=AP2. The total of the modeling universe, N, also remains the same. The structure of the two universes can be viewed in Table 6 where it is clear that predicted events in the conservative-matched universe are the subset of the corresponding liberal-matched universe. The nested nature of the events and nonevents helps explain the models' relative performance.

Universe		Conservative		Liberal	
Actual		Yes	No	Yes	No
Matched	Yes	TP1	FP1	TP2	FP2
	No	FN1	TN1	FN2	TN1

Table 6. Conservative and liberal-matched universes

EFFECT OF MATCHING ON MODEL PERFORMANCE

The actual labeling of observations in the training sample is done through matching. Selecting a confidence level determines, in a binary response model, events and non-events. The actual events, such as new activations, remain unknown and are subject to a successful match. We would like to develop an accurate and robust predictive model that is not unduly biased by the selection of the confidence level. In order to evaluate the effect of matching confidence levels on model performance, we suggest the following methodology:

- Select a range of confidence levels $c_1<c_2< \ldots <c_k$, where lower c_i indicates more restrictive, conservative, matching; and conduct respective matches.
- Construct models for each match.
- Calculate measures of performance, e.g. lift to evaluate the models in own and alternate-matched universes.

Such an approach could be expensive to implement so we suggest a modification in which the number of confidence levels is small, say, k=2, and the single model is constructed to remove its functional form confounding the effect of matches on a model's performance.

AN ILLUSTRATIVE EXAMPLE

We illustrate the model evaluation methodology in a fuzzy matching setting with an example of two datasets, internal and external, matched at two confidence levels; conservative $c_1<c_2$ and liberal $c_1<c_2$, where $c_1<c_2$. Two models were fit to each of the conservative- and liberal-matched universes. The performance of the models has been evaluated in their own- and alternate-matched universes, as shown in Table 7. The own-matched universe is defined as the same universe in which the model was constructed (estimated), for example, when conservative-matched universe based model is used to produce lift in the same universe, that is for the same target. The alternate-matched universe is defined as the universe that is different than the one in which the model was constructed (estimated), for example, when the conservative-matched universe model and its ranking is evaluated on the liberal-matched universe and its liberal target. Naturally, the universes and their binary target for conservative and liberal levels of matching are different.

	Model	
Universe	Conservative	Liberal
Conservative	A	B
Liberal	C	D

Table 7. Schematic analysis of models comparison in multiple universes.

We analyze model performance in the following universe settings:

1) Conservative and Liberal models in their own-matched universes, A vs. D
2) Conservative and Liberal models in alternate-matched universes, B vs. C
3) Conservative model in its own- and alternate-matched universes A vs. C
4) Liberal model in its own- and alternate-matched universes B vs. D

The effect of matching confidence levels can be shown in a difference of model performance as measured, for example, by lift. All lifts shown below are cumulative.

We introduce pseudo paired-t statistic, t*, to measure the difference between two lift curves. Let W denote the difference between the lifts in k ranks (k=20 for semi-deciles). Then, t* can be defined as:

$$t^* = \frac{\bar{W}}{S_W/\sqrt{k}}$$

Consequently, to measure the difference between to lifts, we use log-worth value L* = -log(p), where p is a p-value of 2-sided paired t-test.

The conservative model (A) outperforms the liberal model (W) in their own-matched universes, as shown in Figure 1 (A vs. D), possibly due to fewer False-Positives in the conservative-matched universe. The pseudo L*=3.39 indicating the noteworthy difference between the two curves.

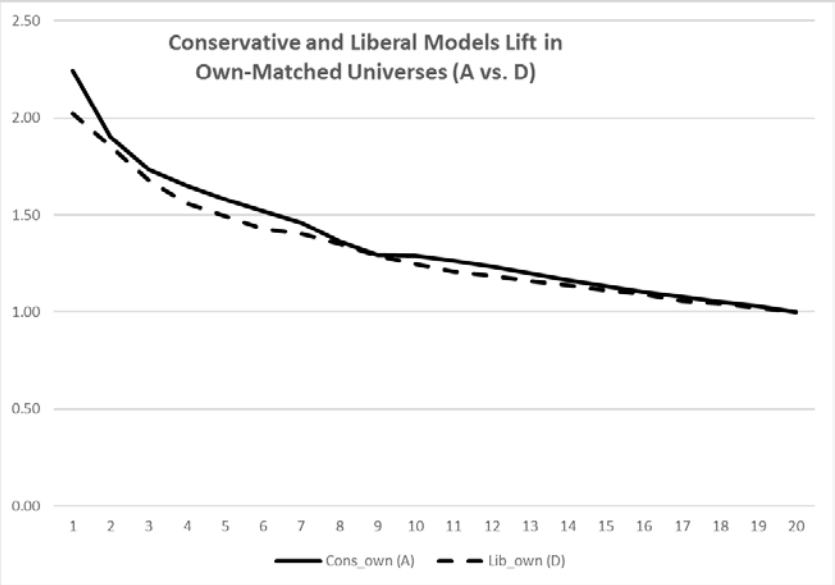


Figure 1. Conservative model lift (A) outperforms liberal-matched model in own-matched universes.

On the other hand, the liberal model (B) outperforms the conservative model (C) in the alternate-matched universes as shown below in Figure 2 (B vs. C). It seems that the liberal model adapts better to an alternate-matched universe, probably due to the larger number of events that the model is based on. As a result, the log-worth value here L*=5.99 is larger than previously in Figure 1. The lifts in the alternate-matched universes (Figure 2) are generally lower than in the own-matched ones, Figure 1 for both models.

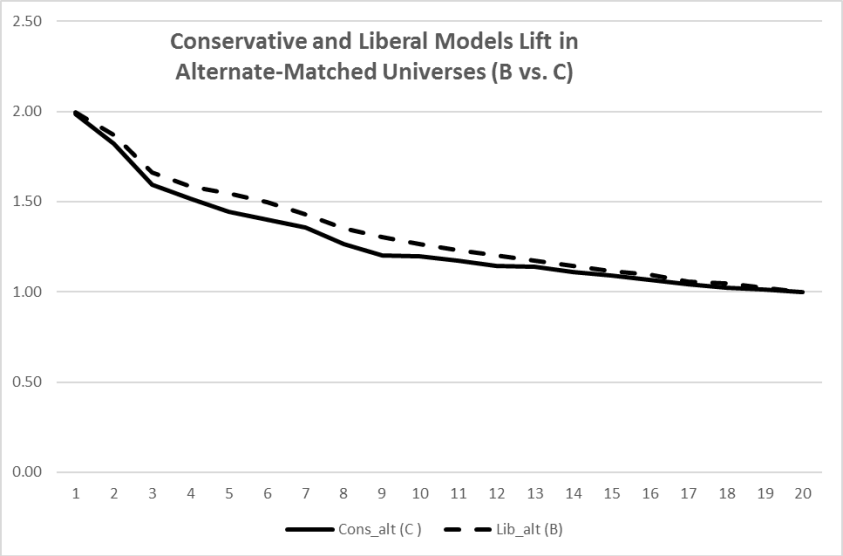


Figure 2. Conservative and liberal models lift in the alternate-matched universes.

The conservative model lift (A) in its own conservative-matched universe of 2.24 outperforms its lift in the alternate, liberal-matched, universe of 1.99 as demonstrated below in Figure 3 (A vs. C). The log-worth value L*=5.67.

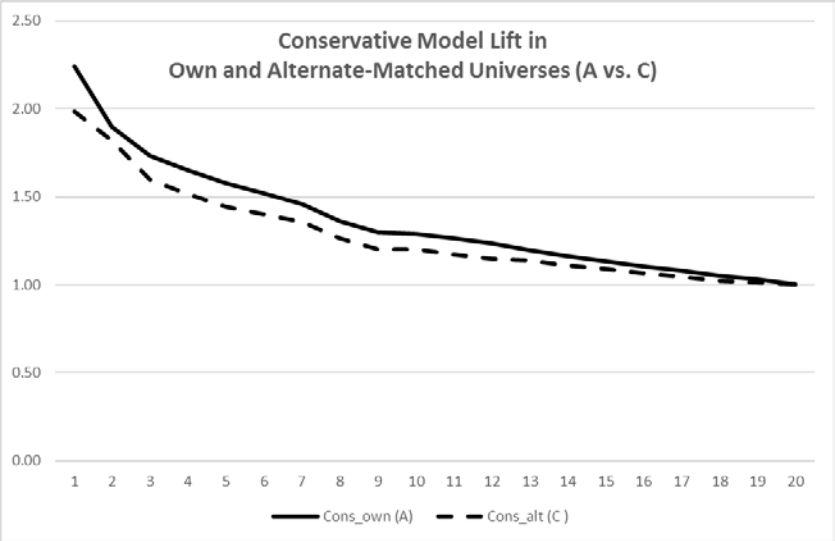


Figure 3. Conservative model lift in own- and alternate-matched universes.

The liberal model in its own-matched universe lift is only marginally higher than in the conservative-matched universe, as shown below in Figure 4 (B vs. D) with the log-worth value of L*=1.95.

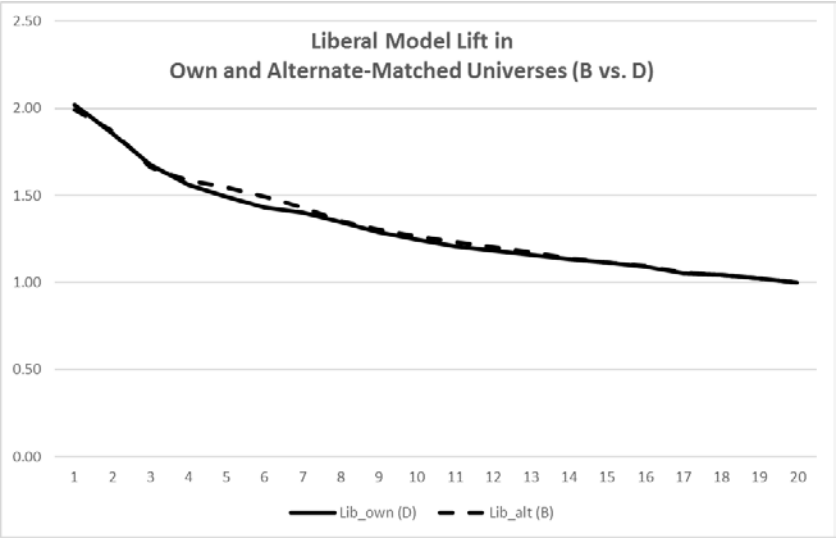


Figure 4. Liberal model lift in its own- and alternate-matched universes.

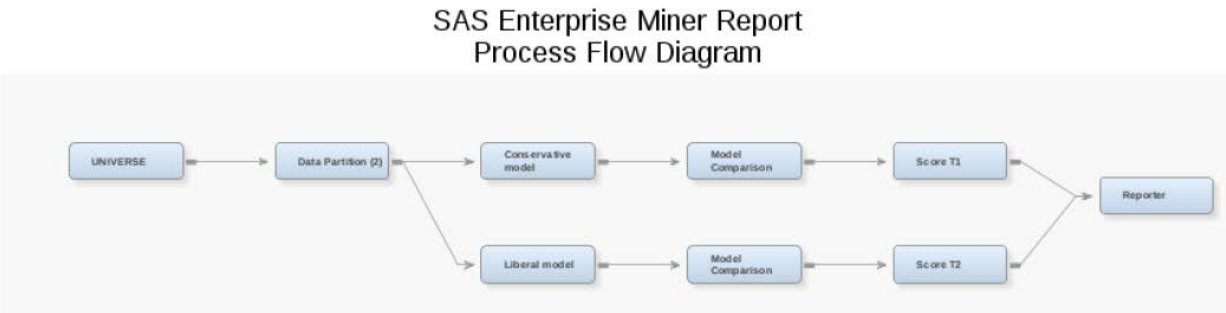
5. PSEUDO-CODE FOR MODEL COMPARISON

SAS pseudo-code that was used to construct model evaluation in its own- and alternate-matched universes consists of four basic steps:

- Step 1. Construct conservative and liberal-matched universes for various levels of COMPGED and/or n-gram confidence levels, and merge them by the common KEY to obtain the UNIVERSE and respective conservative and liberal Target variables: Target_C and Target_L.
- Step 2. Fit the model Targets in the universe and fit the respective models using SAS EM.
- Step 3. Rank the UNIVERSE with the conservative and liberal models to get Score_C and Score_L datasets.
- Step 4. Merge Score_C and Score_L using the common KEY.

```
*Step 1. Merge conservative and liberal matches;
data UNIVERSE;
      MERGE MATCH_CONS(in=C)
            MATCH_LIB(in=L);
      BY KEY;
      IF C then Target_C=Target; else
      IF L then Target_L=Target;
RUN;

*Step 2. Fit the models to Target_C and Target_L and score the
UNIVERSE to produce Score_C and Score_L datasets with the respective
scores: P_Target_C1 and P_Target_L1, as shown in Display 1.
```



Display 1. SAS EM flow diagram for modelling the conservative and liberal-matched universes.

```
*Step 3. Rank the UNIVERSE using conservative and liberal model
scores;

PROC RANK DATA = Score_C
      DESCENDING
      GROUPS=20
      TIES=MEAN
      OUT=SCORE_C_Rank;
      VAR P_Target_C1;
RANKS rank_P_Target_C1;
```

```
RUN;

PROC RANK DATA = Score_L
      DESCENDING
      GROUPS=20
      TIES=MEAN
      OUT=SCORE_L_Rank;
      VAR P_Target_L1;
RANKS rank_P_Target_L1;
RUN;

*Step 4. Merge Score_C and Score_L using the common KEY.

DATA SCORE_all;
      MERGE SCORE_C_Rank
            SCORE_L_Rank;
      BY KEY;
RUN;

PROC FREQ data= SCORE_all;
      TABLE rank_P_Target_C1 * rank_P_Target_L1 * Target_C * Target_L /
      OUT = new outpct list;
RUN;
```

6. CONCLUSIONS

Summarizing the above analysis, we note that the two models perform in a consistent manner with higher lift in their own-matched universe than in the alternate one. In the considered sample, there has been a moderate effect of matching confidence level on model's performance. As a result, the conservative model seems to perform somewhat better than the liberal model in their own-matched universes. This would suggest that the conservative confidence level c_1 should be used for data matching.

The analysis of the effect of matching on inference and models performance is preliminary and reporting on the work in-progress. The key issue is that we don't know what is the actual state of the relationship between matched universes. It is a question open to leveraging methods of statistical latent variables analysis. There is a need for better understanding of matching and model-based inference given their wide application in business and other areas.

It would be helpful for business applications to be able to compare the distributions of models' scores in own and alternate-matched universes using, for example, Kolmogorov-Smirnov statistic, as this may help identify in more details how matching affects the model selection and its performance.

SAS Enterprise Miner can potentially be used not only for model development but also for comparing models' performance in different universes. This could provide diagnostic measures and a common framework for better understanding of how matching affect model performance.

Model performance evaluation, can be extended to the subsets of the both universes to gain insight into nature of the effect of confidence level on model performance. In case of the two universes considered a non-overlapping subset of interest can be constructed by removing a miss-matched subset (AN2+AP2)-(AN1+AP1). While we evaluated a common model scored in own- and alternate-matched universes, it is possible to expand the approach to select and evaluate a common model to eliminate its effect on the conclusions. Other measures such as ROC, can also be used to evaluate model performance in relation to the matching confidence level. Given the importance and widespread practice of data matching, it

would be prudent to evaluate the effect of matching methods on revenue and profitability of business initiatives.

7. ACKNOWLEDGEMENTS

Authors would like to thank Concordia University Part-Time Faculty Association for the financial support of this research.

8. REFERENCES

Roesch, Amanda. “Matching Data Using Sounds-Like Operators and SAS® Compare Functions.” Educational Testing Service, Princeton, NJ. Available at: http://support.sas.com/resources/papers/proceedings12/122-2012.pdf

DeShon, Joe. “Creating a Q-gram algorithm to determine the similarity of two character strings.” Boehringer-Ingelheim Vetmedica SAS. Available at: http://support.sas.com/resources/papers/proceedings16/2080-2016.pdf

Sloan, Stephen et Dan Hoicowitz. 2016.“Fuzzy Matching: Where Is It Appropriate and How Is It Done? SAS® Can Help.” Accenture Federal Services, SAS Global Forum 2016, Paper 7760-2016

SAS. 2016. “COMPGED Function”: Accessed December 20, 2016. http://support.sas.com/documentation/cdl/en/lefunctionsref/69762/HTML/default/viewer.htm#p1r4l9jwgatggtn1ko81fyjys4s7.htm

SAS. 2016. “SPEDIS Function”: Accessed December 20, 2016. http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245949.htm