

Paper ST-160

Reject Inference Methodologies in Credit Risk Modeling

Derek Montrichard, Canadian Imperial Bank of Commerce, Toronto, Canada

ABSTRACT

In the banking industry, scorecards are commonly built to help gauge the level of risk associated with approving an applicant for their product. Like any statistical model, scorecards are built using data from the past to help infer what would happen in the future. However, for application scorecards, any applicant you declined cannot be used in your model, since the observation contains no outcome. Reject Inference is a technique used such that the declined population can now be included in modeling. This paper discusses the pros, cons, and pitfalls to avoid when using this technique.

INTRODUCTION

In the credit card industry, one of the original and more ubiquitous use of scorecard models is in application scoring and adjudication. When a prospect approaches the bank for a credit card (or any other type of loan), we need to ask ourselves if this individual is credit worthy, or if they are a credit risk and will default on their loan. Logistic regression models and scorecards are continuously being built and updated based upon the bank's customers' previous performance, and to identify key characteristics which could help aid in describing the future performance of new customers.

In any statistical model, the key assumption one makes is that the sample used to develop the model is generally indicative of the overall population. In the case of application scoring, this assumption does not hold true. By approving likely "good" customers and declining likely "bad" ones, the modeling dataset is inherently biased. We have known performance on the approved population only, which we absolutely know does not behave like the declined population (otherwise, why would we decline those people?). If the model is to be used only on the approved population to estimate further bad rates, the selection bias does not become a factor. However, since the model is generally meant to be applied on the entire through-the-door (TTD) population to make new decisions on whom to approve or decline, the bias becomes a very critical issue. Reject Inference methodologies are a way to account for and correct this sample bias.

KEY GOALS OF REJECT INFERENCE

In using only the approved / Known Good-Bad (KGB) population of applicants, a number of gaps occur in any statistical analysis due to the high sampling bias error. To begin with, any characteristic analyses will be skewed due to the "cherry picking" of potential good customers across the range of values. If a characteristic is truly descriptive of bad rates across the entire population, it follows that the approval rates by the same characteristic should be negatively correlated. For example, if an applicant has not been delinquent on any loans within the past 12 months, the segment's overall bad rate should be relatively small, and we would be approving a large proportion of people from this category. However, people with 4 or more bad loans in the past year are a serious credit risk. Any approvals in this category should have many other "good" characteristics to override this one derogatory characteristic (such as very large income, VIP customer at the bank, etc.). The overall population trend will then get lost by the selective sampling.

Number of Major Derogs at Bureau

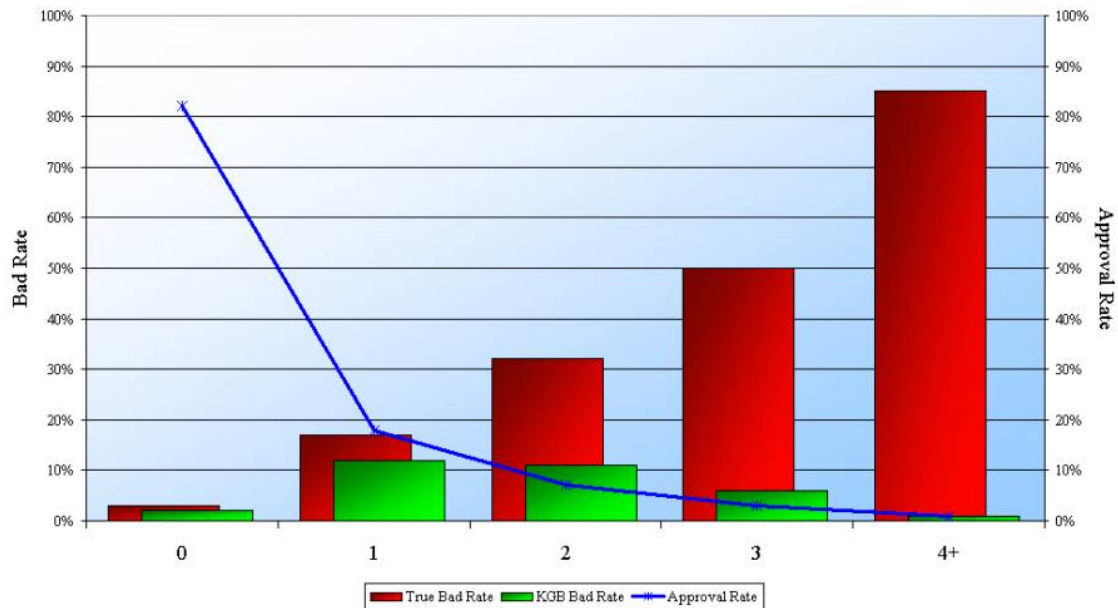


Chart 1: Bad Rates by characteristic for known population and true population

If there is a better understanding of the key characteristics across both the approved and declined region, then better models can be built to be used over the entire TTD population. This potential improvement is hoped to be seen on two key fronts in application scoring:

- 1) Segregating the truly good accounts from the bad accounts (ROC, GINI statistics)
- 2) Accuracy with respect to predicted bad rate to final observed bad rate

This second issue is quite key for decision-making. In using the application score, cutoffs can be set by either trying to match a certain approval rate or by a particular risk threshold. If the estimates on the new bad rates are underestimated by a significant factor, much higher losses will occur than expected, which would be disastrous to any business.

Bad Rates by Model Rank

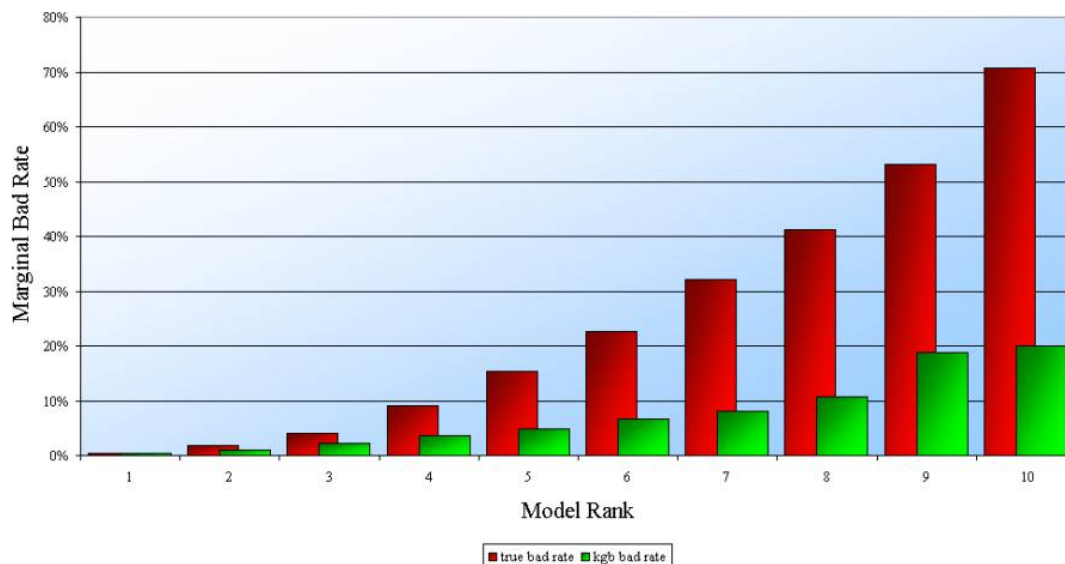


Chart 2: Bad Rates by model rank for known population and true population

One final gap is in the comparison between two models. To make a judgment as to which model would be best to use as the “Champion” scorecard, most naïve comparisons simply would look at the performance over the approved population, and assume that if one model performs better on the KGB group, it will also be better over the entire TTD population. This is not always the case. To simulate this effect, a sample of data was created, and a model was built on the entire dataset. Based upon this model, the lowest 50% of the population were “declined” and their performance masked. A new model was built solely upon the “approved” group. When the ROC curves were compared based upon the approved data, the KGB model appears to outperform the original model. However, when the “declines” were unmasked, the KGB model is now worse.

ROC Curves of Models on KGB Population

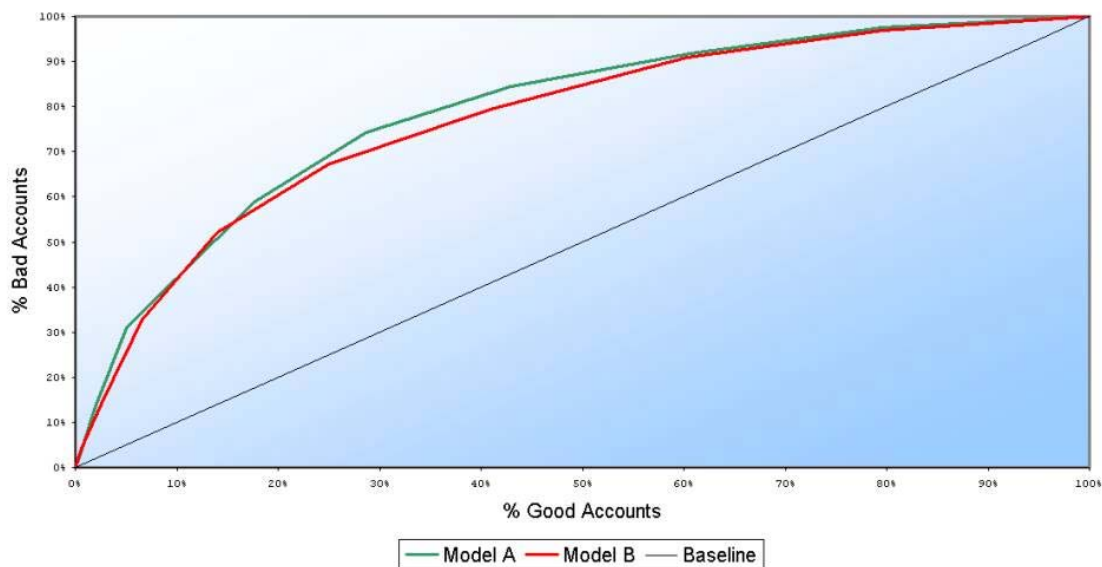


Chart 3: ROC curve on Known Good Bad Population (Model A appears better)

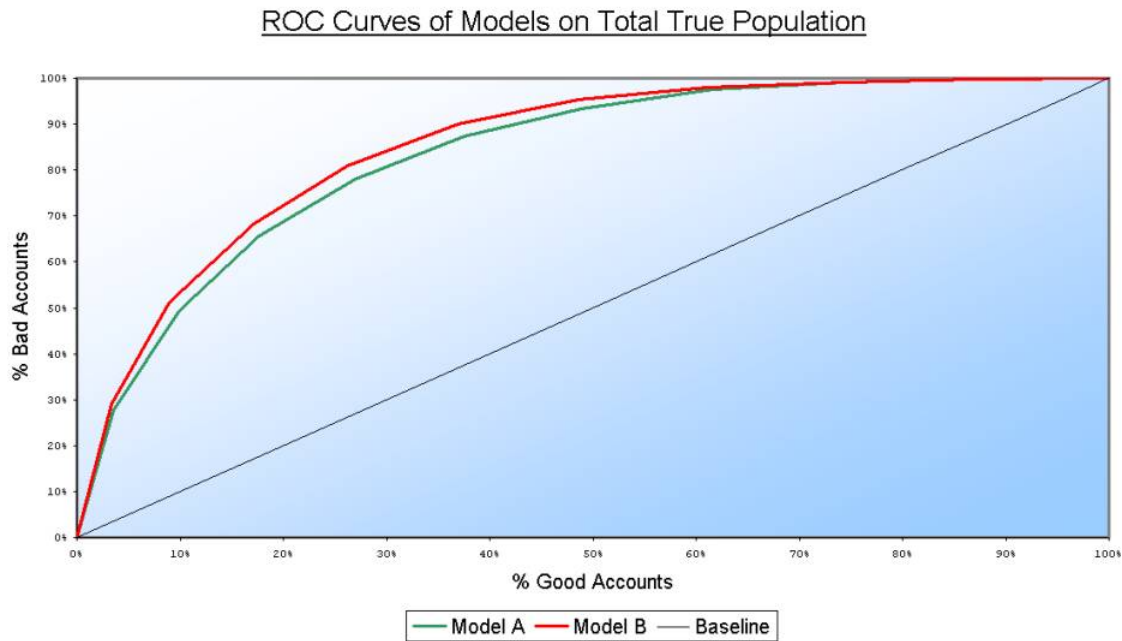


Chart 4: ROC curve on Total Population (Model B appears better)

By gaining insight to the properties of the declined population, as opposed to simply excluding these records from all modeling, it is hoped that the impact of these issues will be minimized.

DIFFERENT METHODS OF REJECT INFERENCE

There are three different subclasses of Reject Inference methodologies commonly in use today. In describing these different methods, pseudo-code will be presented with the following terms:

Datasets:

- full : SAS® dataset including all observations, including both approvals and declined population
- approve : SAS dataset containing approvals only
- decline : SAS dataset containing declines only

Variables:

- approve_decline : SAS variable to identify if the record was approved ("A") or declined ("D")
- good_bad : SAS variable to identify if the outcome was good ("good") or bad ("bad"), and missing for all declined records
- x1, x2, x3, ..., xn : SAS variables fully available for modeling purposes
- i1, i2, i3, ..., im : SAS independent variables which cannot be used for modeling purposes, but available for measurement purposes (e.g. Age, gender, region, national unemployment rate at month of application, prime interest rate, etc.)
- model_1, model_2, ..., model_p : SAS variables containing preexisting models (Credit Bureau Score, original application score, etc.)

MANUAL ESTIMATION

Otherwise known as "Expert Rules", this method is when the business analyst or modeler uses preexisting knowledge to manually simulate performance on the declined population. For example, when building other models, the analyst sees that when x1 is greater than 5, over 50% of that group were labeled as "bad" after 12 months. Therefore, when building the application scorecard, the analyst will identify this segment from the declined population and randomly assign 50% of the people as "good" and the remaining as "bad". For the remaining declines, the analyst assumes an overall 30% bad rate.

```

data model;
set full;
  if (approve_decline = "D") and (x1 > 5) then do;
    if ranuni(196) < 0.5 then new_good_bad = "bad";
    else new_good_bad = "good";
  end;
  else if (approve_decline = "D") then do;
    if ranuni(196) < 0.3 then new_good_bad = "bad";
    else new_good_bad = "good";
  end;
  else new_good_bad = good_bad;
run;
proc logistic data=model;
  model new_good_bad = x1 x2 ... xn/stepwise;
run;

```

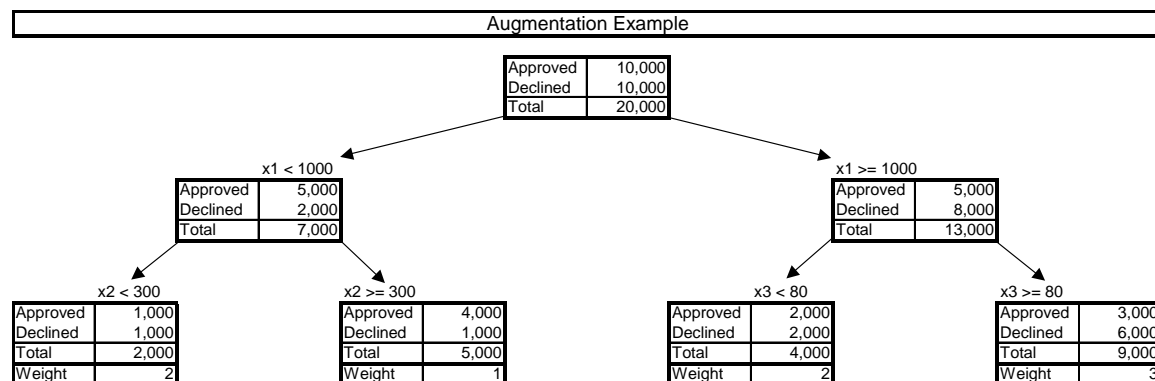
While simple in nature, the manual estimation method is unvettable. In the previous example, the analyst simply assigned the values of 50% and 30% by “instinct”. If another modeler decides that the rates should be 60% and 50%, completely different results will occur. Without consensus across all groups on the assumptions, the model can never be replicated.

AUGMENTATION

Augmentation is the family of methods that use the relationship between the number of approved accounts in a cluster to the number of declines. Based upon this ratio, the approved accounts will then be reweighted such that the sum of the weights will equal the total number of applications. For example, when investigating clusters, the analyst sees that a particular cluster has 1,000 approves and 1,000 declines. In the final model, each of these 1,000 approved accounts will be modeled as if they counted as two observations each, so that the cluster has a total weight of 2,000.

Typical augmentation methods involve two modeling steps. The first step is to create an “approve/decline” model on the entire dataset, either through logistic regression, or decision trees, or neural networks. Based upon this model, each record will have a probability to be approved associated with it. The final weight is simply $1/p(\text{approve})$. Two examples are given below: The first uses a decision tree strategy to classify observations and assign weights; the second uses a logistic regression approach.

Decision Tree Example



Logistic Regression Example

```

Proc logistic data=full;
  Model approve_decline (event="A") = x1 x2 ... xn/stepwise;
  Output out=augment p=pred;
Run;
data model;
  set augment;
  where (approve_decline = "A");
  aug_weight = 1/pred;

```

```

run;
proc logistic data=model;
  model good_bad = x1 x2 ... xn/stepwise;
  weight aug_weight;
run;

```

Augmentation does have two major drawbacks. The first deals with how augmentation handles “cherry picked” populations. In the raw data, low-side overrides may be rare and due to other circumstances (VIP customers). By using only the raw data, the total contribution of these accounts will be quite minimal. By augmenting these observations to now represent tens to thousands of records, the “cherry picking” bias is grossly inflated.

The second drawback is due to the entire nature of modeling on the Approve/Decline decision. When building any type of model, the statistician is attempting to build a mathematical model that will describe the target variable as accurately as possible. For the Approve/Decline decision, a model **is** known to perfectly describe the data. Simply by building a decision tree that replicates the Approve/Decline strategy, approval rates in each leaf should be either 100% or 0%. Augmentation is impossible on the decline region, since you no longer have any approved accounts to augment with (weight = $1/p(\text{approve}) = 1/0 = \text{DIV0!}$).

EXTRAPOLATION

Extrapolation comes in a number of forms, and may be the most intuitive in nature to a statistician. The idea is fairly simple: Build a good/bad model based on the KGB population. Based upon this initial model, estimated probabilities to be bad can be assigned to the declined population. These estimated bad rates can now be used to simulate outcomes on the declines, either by Monte Carlo Parceling methods (random number generators) or by splitting the one decline into fractions of goods and bads (fuzzy augmentation).

Parceling Example

```

Proc logistic data=approve;
  Model good_bad (event="bad") = x1 x2 ... xn/stepwise;
  Score data=decline out=decline_scored;
Run;
data model;
  set approve (in=a) decline_scored (in=b);
  if b then do;
    myrand = ranuni(196);
    if myrand < p_bad then new_good_bad = "bad";
    else new_good_bad = "good";
  end;
  else new_good_bad = good_bad;
run;
proc logistic data=model;
  model new_good_bad = x1 x2 ... xn/stepwise;
run;

```

Fuzzy Augmentation Example

```

Proc logistic data=approve;
  Model good_bad (event="bad") = x1 x2 ... xn/stepwise;
  Score data=decline out=decline_scored;
Run;
Data decline_scored_2;
  Set decline_scored;
  new_good_bad = "bad";
  fuzzy_weight = p_bad;
  output;
  new_good_bad = "good";
  fuzzy_weight = p_good;
  output;
run;

```

```

data model;
  set approve (in=a) decline_scored_2 (in=b);
  if a then do;
    new_good_bad = good_bad;
    fuzzy_weight = 1;
  end;
run;
proc logistic data=model;
  model new_good_bad = x1 x2 ... xn/stepwise;
  weight fuzzy_weight;
run;

```

Note that in the fuzzy augmentation example, each decline is now used twice in the final model. The declined record is broken into two pieces as a partial good and a partial bad. Therefore, if the estimated bad rate from the KGB model gives a probability to go bad of 30% on a declined individual, we take 30% of the person as bad, and 70% of the person as good. This method is also more stable and repeatable than the parceling example due to the deterministic nature of the augmentation. In parceling, each record is randomly placed into a good or bad category. By changing the random seed in the number generator, the data will change for new runs of code.

At face value, augmentation appears to me the most mathematically sound and reliable method for Reject Inference. However, augmentation used in a naïve fashion can also lead to naïve results. In fact, the results can be better than just modeling on the KGB population alone. Reject Inferencing simply reinforces the perceived strength of the model in an artificial manner. In any model build stage, the vector B is chosen to minimize the error of Y-XB. On the KGB phase, we have:

$$\hat{Y} = X_{KGB} \hat{B}_{KGB}$$

On the extrapolation stage, we now want to find a suitable B to minimize the error terms on

$$\text{minimize} : Y - X_{ALL} \hat{B}_{ALL}$$

We can decompose this to two parts: one for the approved population, and one for the declined:

$$\text{minimize} : Y_{approve} - X_{approve} \hat{B}_{ALL} \text{ and } Y_{decline} - X_{decline} \hat{B}_{ALL}$$

By definition, \hat{B}_{KGB} minimizes the error terms for the approved population from the KGB model. On the declined population, $Y_{decline} \equiv X_{decline} \hat{B}_{KGB}$. Using \hat{B}_{KGB} for \hat{B}_{ALL} will minimize error terms. All reject inference has done is assumed that absolutely no error will occur on the declined population, when in reality some variation will be seen. In fact, since we are extrapolating well outside our known region of X-space, we expect the uncertainty to increase in the declined region, and not decrease.

This result can also lead to disaster when comparing two models. When comparing your existing champion application score to a new reject inference model, the new model will always appear to outperform the existing model. The comparison using declined applicants will need an estimate of the number of bads, which assumed that the new model is “correct”. Since we have already made the assumption that the new model is “more correct” than the old model, any comparison of new vs. old scorecards using the new scorecard Reject Inference weights is meaningless and self-fulfilling.

OPTIONS FOR BETTER REJECT INFERENCE

Reject Inference can be a standard Catch-22 situation. The more bias we have in the approved population compared to the declined population, the more we need reliable and dependable reject inference estimates; however, the more bias there is in the populations, the less reliable reject inference can be. Similarly, the best accuracy of Reject Inference models comes when we can fully describe the declined population based upon information from the

approved population; however, if we can describe the declined group from the approved population, there is no real need for reject inference. How does one go about in escaping this vicious cycle?

The first solution is to simply ask if accurate Reject Inference is necessary for the project. One case may be when going forward, plans are to decrease the approval rate of the population in a significant fashion. While estimations on the declined population may be weak, these applicants will still most likely continue to be declined. The strength of the scorecard will be more focused on the current approved niche, which will then get pruned back in future strategies.

Another situation would be when the current strategy and decision making appears to be random in nature. If the current scorecards appear to show little segmentation of goods from bads. In this case, it may be assumed that the approved population is close enough to a random sample, bias is minimized, and we can go forward to modeling as normal.

Using variables within the approve/decline strategy itself becomes important. By forcing these variables into the model, regardless of significance and relationship between the estimated Betas and bad rate, may help control for selection bias. If different strategies are applied to certain sub-populations (such as Branch applications vs. Internet applications), separate inference models could be built on these sub-groups. After building these sub-models, the data can be recombined into one full dataset.

One key for reject inference models is to not be limited by using only the set of variables that can enter the final scorecard. In the reject inference stage, use as much information as possible, including:

- Models, both previous and current. This approach is also referred to as Maximum Likelihood RI. If a current scorecard is in place and used in decision-making, we must assume then that it works and can predict accurately. Instead of disregarding this previous model, and re-inventing the wheel, we start with these models as baseline independent variables.
- "Illegal" variables. Variables such as age, gender, and location may not be permissible within a credit application scorecard, but they still may contain a wealth of information. Using these for initial estimates may be permissible, as long as they are removed from the final scorecard.
- Full set of X scorecard variables.

Fuzzy / MIX Augmentation Example

```
Proc logistic data=approve;
  Model good_bad (event="bad") = model_1 model_2 ... model_p i1 i2 ... im x1 x2 ... xn
  /stepwise;
  Score data=decline out=decline_scored;
Run;
Data decline_scored_2;
  Set decline_scored;
  new_good_bad = "bad";
  fuzzy_weight = p_bad;
  output;
  new_good_bad = "good";
  fuzzy_weight = p_good;
  output;
run;
data model;
  set approve (in=a) decline_scored_2 (in=b);
  if a then do;
    new_good_bad = good_bad;
    fuzzy_weight = 1;
  end;
run;
proc logistic data=model;
  model new_good_bad = x1 x2 ... xn/stepwise;
  weight fuzzy_weight;
run;
```


By using the set of MIX variables instead of the set of X alone, a wider base for estimation is created for more stable results. Plus, by using the entire set of MIX variables to estimate \hat{B}_{KGB} , it no longer becomes the case that

$$\hat{B}_{All} \equiv \hat{B}_{KGB} .$$

If possible, the most accurate way to perform Reject Inference is to not only expanding your variable base, but your known good-bad base. This can be done by performing a designed experiment, where for one month, a very small proportion of applicants, regardless of adjudication score, are approved. By randomly approving a small number of individuals, augmentation methods can be applied to adjust the proportional weights. Once the augmentation method is used to create initial estimates, these estimates can then be applied to the remaining declined population, and extrapolation methods can be used for a final three-tied model. From a statistical approach, this may be the most robust method for application modeling, but from a business standpoint, it may be the most costly due to the higher number of bad credits being issued. This method must have sign-off from all parties in the organization, and must be carefully executed.

Finally, in making comparisons between models, boosting methods can be used to get better estimates of overall model superiority. Instead of using the inferred bad rates from the extrapolation stage, a second and independent model based entirely on champion scores is also built. On the declines, the inferred bad rate on the declines would simply be the average $p(\text{bad})$ from the two separate models. A third or fourth model can also be built using variables that were not part of the final new scorecard, thus giving an independent (yet weaker) view of the data. Using these averages on an out of time validation dataset tend to give a more robust description of how well (or worse) the new scorecard performs against the current scorecard.

CONCLUSION

Inference techniques are essential for any application model development, especially when the decision to approve or decline is not random in nature. Modeling and analyses on the known good/bad region alone is not sufficient. Some estimation must be applied to the declines to allow the bank to set cut-offs, estimate approval volume, estimate delinquency rates post implementation, swap in/skip out analyses, etc. Without tremendous care and caution, Reject Inference techniques may lead modelers to have unrealistic expectations and overconfidence of their models, and may ultimately lead to deciding upon the wrong model for the population at hand. With care and caution, more robust estimates can be made on how well the models perform on both the known population and on the entire through the door population.

REFERENCES

- Gongyue Chen & Thomas B. Astebro. A Maximum Likelihood Approach for Reject Inference in Credit Scoring
- David Hand & W.E. Henley. Can Reject Inference Ever Work?
- Dennis Ash, Steve Meester. Best Practices in Reject Inferencing
- G Gary Chen & Thomas Astebro. The Economic Value of Reject Inference in Credit Scoring
- Geert Verstraeten, Dirk Van den Poel. The Impact of Sample Bias on Consumer Credit Scoring Performance and Profitability
- Gerald Fahner. Reject Inference
- Jonathan Crook, John Banasik. Does Reject Inference Really Improve the Performance of Application Scoring Models?
- Naeem Siddiqi. Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring

ACKNOWLEDGMENTS

I would like to thank Dina Duhon for her proofreading, commentary, and general assistance in creating this paper, and Gerald Fahner and Naeem Siddiqi for their wonderful conversations about Reject Inferencing in modeling. I would also like to thank CIBC for supporting me in this research. Finally, I would like to thank Joanna Potratz for her ongoing support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Derek Montrichard
CIBC
750 Lawrence Ave West

Toronto, Ontario M6A1B8
Work Phone: 416.780.3680
E-mail: Derek.montrichard@cibc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.