Paper 1769-2018

Reducing Customer Attrition with Machine Learning for Financial Institutions

Nate Derby, Stakana Analytics, Seattle, WA

ABSTRACT

As financial institutions market themselves to increase their market share against their competitors, they understandably focus on gaining new customers. However, they must also retain (and further engage) their existing customers. Otherwise, the new customers they gain can easily be offset by existing customers who leave. Happily, predictive analytics can be used to forecast which customers have the highest chance of leaving so that we can effectively target our retention marketing toward those customers. This can make it much easier and less expensive to keep (and further cultivate) existing customers than to enlist new ones.

A previous paper showed a simple but fairly comprehensive approach to forecasting customer retention. However, this approach fails in many situations with more complicated data. This paper shows a much more robust approach using elements of machine learning. We provide a detailed overview of this approach and compare it with the simpler (parametric) approach. We then look at the results of the two approaches.

With this technique, we can identify those customers who have the highest chance of leaving and the highest lifetime value. We then make suggestions to improve the model for better accuracy. Finally, we provide suggestions to extend this approach to cultivate existing customers and thus increase their lifetime value.

Code snippets will be shown for any version of SAS® but will require the SAS/STAT package. This approach can also be applied to many other organizations and industries.

The `%makeCharts` and `%makeROC` macros in this paper are available at `nderby.org/docs/charts.zip`.

INTRODUCTION: THE PROBLEM WITH CUSTOMER ATTRITION

Suppose we're the fictitious Fisher Bank of Ames, Iowa, which has 748,593 customers and $9.825B in assets as of March 30, 2018, with none of those customers having only a car loan.[1] Three months before, we had 736,343 customers and $9.681B in assets, so we grew their customer base by 1.66% and their assets by 1.49%. That might sound good on the surface, but it's really not. Our gain of 12,250 customers is a *net* gain. We actually gained 32,707 *new* customers but lost 20,457 *existing* customers. **Our customer attrition is dramatically affecting our net customer growth.**

- If we had retained just 20% of those departing customers, we would have gained a net 16,341 customers, for a 2.22% net customer growth.
- If we had retained 25% of those departing customers, we would have had gained a net 17,364 customers, for a 2.36% net customer growth.
- If we had retained 50% of those departing customers, we would have had gained a net 22,478 customers, for an amazing 3.05% net customer growth.

In fact, **retaining existing customers is usually easier and less expensive than gaining new customers**, since these customers already know and trust the bank, and the bank already knows so much about them. Keeping them might be as simple as making a friendly phone call. **The key is making full use of the data we have on our customers**. Since customer attrition can have such a huge effect on customer growth, why not focus some attention to it? Specifically, we can focus on those customers with the highest value and the highest risk of closing their accounts.

[1] Customers with just a car loan typically behave differently from the normal customers, so we're ignoring them to keep this analysis simple.

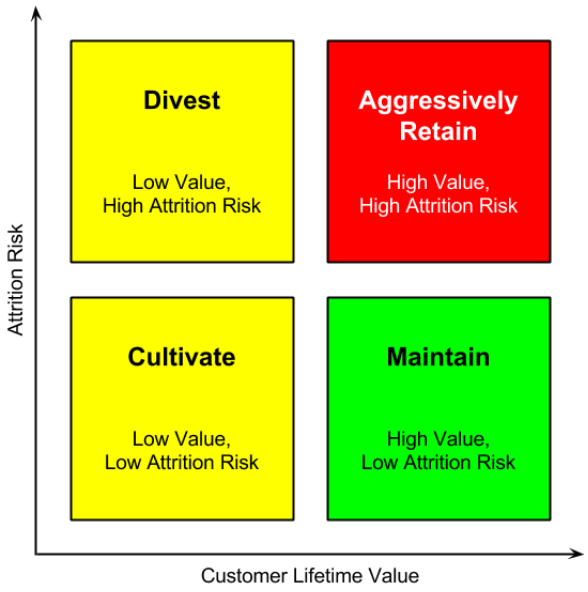


Figure 1: Customer segmentation into quadrants of lifetime value and attrition risk.

If we can identify which customers are most likely to leave, we can proactively focus our marketing efforts on retaining them. With this in mind, Figure 1 shows four attrition marketing quadrants that describe the segmentation of our customers. The horizontal axis shows the *customer lifetime value* (in terms of the aggregate balance of deposits and loans), explained in Fader (2012), which may be very different from the present value. For instance, a student typically has a low present value but will have a much higher lifetime value. The vertical axis shows the attrition risk. We can divide this region into four quadrants:

- *Maintain*: These are customers with a high value and low attrition risk. These are generally our best customers, and we should keep them satisfied without focusing much attention on them if possible.
- *Cultivate*: These are customers with a low value and low attrition risk. We're not in danger of losing them, but they aren't growing our bank. We should cultivate these customers to have a higher value.
- *Divest*: These are customers with a low value and high attrition risk. These are the customers that we wouldn't really mind losing, since they're low value and would require some effort to keep.
- *Aggressively Retain*: These are customers with a high value and high attrition risk, and **they are the ones we should pay the most attention to**. If we lose these customers, it will make a large difference to our asset size (either now or in the future).

So **how can we use our data to focus on these high-value, high-risk customers?**

Thomas (2010) presents a basic but very useful approach using logistic regression, and Karp (1998) provides details on the logistic regression procedure itself. Lu (2002) and Lu (2003) predict attrition and lifetime value using survival analysis, whereas Su et al. (2009) use statistical clustering. Both of these are more complex methods.

However, it turns out that the simpler logistic regression approach, as shown in Derby and Keintz (2016), gives suboptimal results. As with that paper, we'll follow the example of Thomas (2010), showing a basic but very useful approach that can produce quick results once the code infrastructure is in place. We'll just use it with a machine learning algorithm rather than logistic regression.

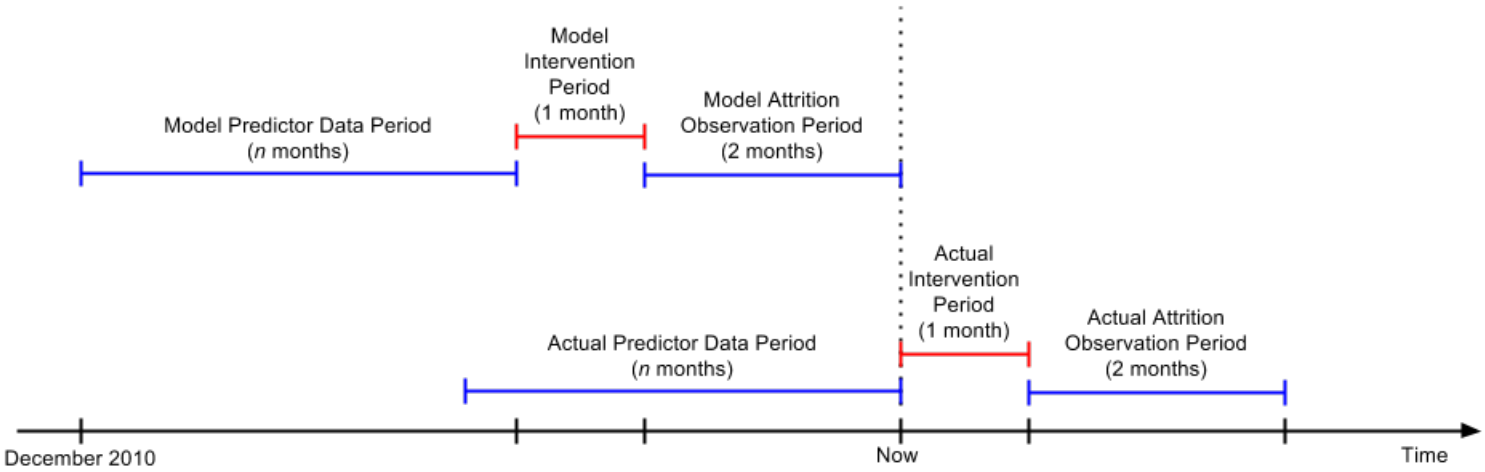


Figure 2: The scheme for duplicating our raw data into a modeling (upper) and a scoring (lower) data set.

DATA PREPARATION

We're building a *statistical model*, an equation that will tell us the probability that a customer will close his/her account 2-3 months later. Before we can do that, we'll need to prepare the data, which involves three steps applied to raw data from our core processor (customer demographic, account, and transactional data):

- *Duplicating the data* into two data sets that are almost the same: one for building the statistical model, and the other for using that statistical model to give us our forecasts.
- *Building our variables* for both of the above data sets.
- *Partitioning the data* (the first one above, for building the statistical model).

We'll explain the reason behind each step and how we're implementing it. We'll then show results from simulated data inspired by real data from Verity Credit Union of Seattle.

DUPLICATING THE DATA

A statistical model is an equation where the *input variables* are from the known data and the output variable is the unknown quantity we'd like to know. In this case, the input variables X_1, X_2, X_3, ... are attributes about the customer effective as of this point in time and the output variable is that customer's risk of leaving the bank 2-3 months later:[2]

$$\text{Probability of leaving 2-3 months later} = f(X_1, X_2, X_3, ...)$$

where f is some function we don't yet know. Once we have the function f, we'll use the input variables X_1, X_2, X_3 and get our probability of leaving. This may sound simple, but to figure out what that function is, we need to use mathematical algorithms on data at a point in time when we can see which customers actually left 2-3 months later. In other words,

- To *build* the statistical model, we need to use data as of three months ago, coupled with which customers left 2-3 months later (which we know).
- To *use* the statistical model, we need to use data as of now, which will tell us which customers are likely to leave 2-3 months later (which we don't know).

Since the statistical model requires input and output variables to be defined in the same way (whether we're building or using the statistical model), the time interval for the input variables must be the same length for both creating and using the statistical models. Therefore, from our raw data we'll create two data sets adjusted for the time intervals, as shown in Figure 2:

[2] 2-3 months later gives us a month to intervene and hopefully change the outcome so that the customer retains his/her account.

- The data set for *building* the statistical model will include input variables up to three months in the past, plus attrition data for the last two months (i.e., which customers closed their accounts).
- The data set for *using* the statistical model will include only input variables, for a time period moved forward by three months.

For consistency (i.e., some months have 31 days, some have 30, some have 28 or 29), we actually use groups of 4 weeks rather than 1 month, even when we call it a month in Figure 2.

We can efficiently code this in SAS by defining a macro:

```
%MACRO prepareData( dataSet );

  %LOCAL now1 now2 now ... attritionEndDate;

  PROC SQL NOPRINT;
    SELECT MAX( effectiveDate )
      INTO :now1
      FROM customer_accounts;
    SELECT MIN( tranPostDate ), MAX( tranPostDate )
      INTO :startDate, :now2
      FROM customer_transactions;
  QUIT;

  %LET now = %SYSFUNC( MIN( &now1, &now2 ) );

  %IF &dataSet = modeling %THEN %DO;

      %LET predictorStartDate = &startDate;
        %* starting at the earliest transaction date ;
      %LET predictorEndDate = %EVAL( &now - 84 );
        %* ending three months ago ;
      %LET attritionStartDate = %EVAL( &now - 56 + 1 );
        %* starting two months ago ;
      %LET attritionEndDate = &now;
        %* ending now ;

      %END;
    %ELSE %IF &dataSet = scoring %THEN %DO;

      %LET predictorStartDate = %EVAL( &startDate + 84 );
        % starting at the earliest transaction date plus three months ;
      %LET predictorEndDate = &now;
        % ending now ;

      %END;

      [SAS CODE FOR PULLING/PROCESSING THE DATA, USING THE MACRO VARIABLES ABOVE]

%MEND prepareData;
```

We can now create both data sets using the exact same process for each of them with the time periods shifted, as in Figure 2:

```
%prepareData( modeling )
%prepareData( scoring )
```

BUILDING OUR VARIABLES

For both of the data sets described above, we'll build variables that might be predictive of a customer closing his/her account. We don't care if these variables are *actually* predictive, as the statistical modeling process will figure that out. But the statistical modeling process is just a mathematical algorithm that doesn't understand human behavior. It needs to know ahead of time which variables to try out. So it's our job to give it those variables to try out, which of course we have to create.

Here are some examples of variables we can try out:

- *Transaction Recency*: When was the last transaction (other than automatic transactions like interest)?
- *External Deposit Recency*: When was the last external deposit?
- *Recent Large Transaction*: Was there a recent large transaction? (Above the 90th or 95th percentile of all transactions over the most recent 2 or 4 weeks)
- *Small Number of Transactions*: Was there a recent small number of transactions? (Above the 90th percentile for the monthly number of transactions)
- *Large Number of Transactions*: Was there a recent large number of transactions? (Below the 10th percentile for the monthly number of transactions)
- *A Seasonality Component*: This is a dummy variable to determine if certain customers are more likely to close their accounts in the summer, or during the Christmas season.

Within SAS, we can code these variables into the `%prepareData` macro we previously defined so that we do the exact same process for both time intervals. As shown below, we have to be sure that we confine ourselves to certain transactions type codes (`tranTypCode`).

```
PROC SQL NOPRINT;
  CREATE TABLE predictorData1 AS
    SELECT
        id_customer,
        MAX( ( &predictorEndDate - tranPostDate )/7 ) AS tranRecency
          LABEL='Transaction Recency (Weeks)',
        MEAN( ABS( tranAmt ) ) AS meanTranAmt LABEL='Mean Transaction Amount',
        N( tranAmt ) AS nTrans LABEL='Number of Transactions',
        N( tranAmt )/ MAX( INTCK( 'month', tranPostDate, &now, 'c' ) )
          AS meanNTransPerMonth LABEL='Mean # Transactions per Month'
      FROM customer_transactions
      WHERE
        tranPostDate BETWEEN &predictorStartDate AND &predictorEndDate AND
        UPCASE( tranTypeCode ) IN ( 'CCC', 'CCD', ... 'WTHD' )
      GROUP BY id_customer;
  CREATE TABLE predictorData2 AS
    SELECT
        id_customer,
        MAX( ( &now - tranPostDate )/7 ) AS depRecency
          LABEL='External Deposit Recency (Weeks)'
      FROM customer_transactions
      WHERE
        tranPostDate BETWEEN &predictorStartDate AND &predictorEndDate AND
        UPCASE( tranTypeCode ) = 'XDEP'
      GROUP BY id_customer;
QUIT;
```

The percentiles can be computed with `PROC UNIVARIATE`, and all of these data sets can be joined by a simple `MERGE` statement within a `DATA` step.

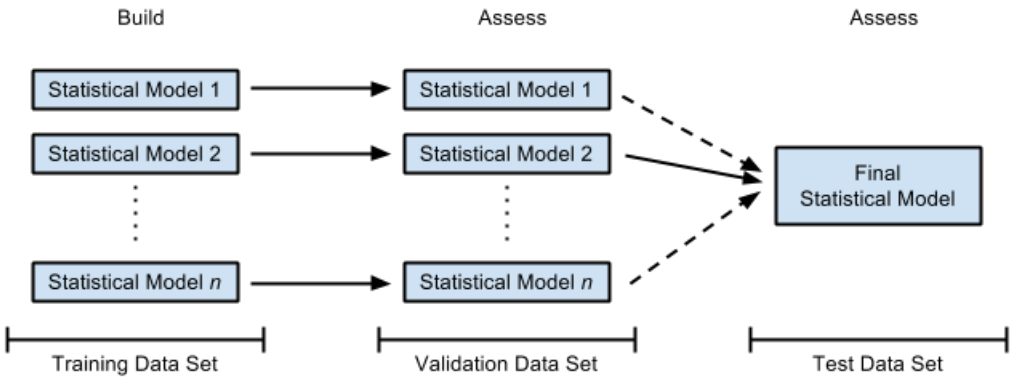


Figure 3: The three data partitions. Only one of the models makes it to the final model (in this case, model 2).

PARTITIONING THE DATA

For the modeling data set in Figure 2, we won't build *one* statistical model for our forecasts. Instead, we'll build several of them, choose the one that gives us the best results, then give an estimate of how accurate those results are. While this process may sound simple, we can't use the same data set for each of these steps, since that could give us biased results (which would be bad). To understand this, think of the data set we use when building a statistical model. This process involves mathematical algorithms that find the equation that best fits the data. If we used the same data set to assess how well that equation fit those data points, then by definition (since the algorithm was *designed* to get the best fit between the equation and the data points) we would get a really good fit! But the whole point of building a statistical model is to predict *data that we haven't seen yet*. **If we've never actually tested how well our statistical model predicts unknown data points, then we won't know how well it forecasts unknown data until we use it.** This could be a recipe for disaster.

There's a much better way to do this. Instead of using the same data set for making the model and then testing it out, we can randomly partition the data points into three distinct sets:

- *The training data set* (60% of the data) is used to build each of the statistical models that we're trying out.
- *The validation data set* (20% of the data) is used to determine how well each of these statistical models actually forecasts attrition. That is, using each of the statistical models we built with the training set, we'll forecast the customers in the validation set who closed their accounts and check their accuracy. **The statistical model that has the best accuracy will be our final statistical model.**
- *The test data set* (20% of the data) is used to determine how well the final model (i.e., the winning model from the validation set) actually forecasts attrition. That is, we'll use the final model to forecast the customers in the test set who closed their accounts and check their accuracy. We do this to double check that everything is OK. If the accuracy is much different than it was for the validation set, it's a sign that something is wrong and we should investigate this further. Otherwise, our final model is all good!

This is illustrated in Figure 3. In SAS, we can do this with the following code at the end of our `%prepareData` macro, using a random uniform distribution with the `RAND` function:

```
DATA trainingData validationData testData;
  SET inputData;
  CALL STREAMINIT( 29 );
  randUni = RAND( 'uniform' );
  IF randUni < .6 THEN OUTPUT trainingData;
    ELSE IF randUni < .8 THEN OUTPUT validationData;
    ELSE OUTPUT testData;
RUN;
```

For our data set of 69,534 customers at the end of March 2015, we get 41,875 customers in our training set (60.22%), 13,807 customers in our validation set (19.86%), and 13,852 customers in our test set (19.92%).

BUILDING THE STATISTICAL MODELS

Building the statistical models is considerably more complex than using logistic regression. Here, we'll use the bootstrap aggregating (bagging) macro `%bagging` as explained in Liu (2012) with different sets of explanatory variables. However, for the purposes of this paper, we could substitute any machine learning technique here.

One problem with the original `%bagging` macro is that it uses the same data set for fitting a model and for scoring. As mentioned in the preceding section, we want these data sets to be separate. However, we can easily modify this algorithm to accompany this by adding a `scoreData` parameter in the macro definition. We can also add an `id` parameter for identifying the individuals:

```
%MACRO bagging( data=, y=, numx=, catx=, ntrees=50, id=, scoreData= );

  ...

%MEND bagging;
```

In the code where the `_tmp1` data set is defined with the `&data` data set, simply define an additional `_temp1` data set with the `&scoreData` data set instead of `&data`, and set the `_id_` variable in both of them:

```
DATA _tmp1 ( KEEP = &y &numx &catx _id_ );
  SET &data;
  _id_ = &id;
RUN;

DATA _temp1 ( KEEP = &numx &catx _id_ );
  SET &scoreData;
  _id_ = &id;
RUN;
```

Later in the code, we score them separately:

```
*** score the scoring data by each tree output file ***;
DATA _score&i ( KEEP = p_&y.1 p_&y.0 _id_ );
  SET _temp1;
  %INCLUDE in_tree;
RUN;

*** score the original data by each tree output file ***;
DATA _score&i.b ( KEEP = p_&y.1 p_&y.0 _id_ );
  SET _tmp1;
  %INCLUDE in_tree;
RUN;
```

As in the original macro, we then calculate the Kolmogorov-Smirnov statistic from the `_score&i.b` data set. Continuing on in this way, we end up with two output data sets: One for the scored data, one for the model data:

```
DATA &outData;
  MERGE &scoreData _temp3;
    BY &id;
RUN;

DATA &outData._m;
  MERGE &data( in=ina ) _tmp3;
    BY &id;
  IF ina;
RUN;
```

ASSESSING THE STATISTICAL MODELS

Using the edited `%bagging` macro as described above will give us a scored validation or test data set (as shown in Figure 3) which will be used in our assessments as described in the next few pages. We call the macro like this, for example:

```
%bagging( data=trainingData, y=attrition, numx=depRecency lom nProducts calls,
  catx=ageTier, ntrees=50, id=id_member, scoreData=validationData );
```

A few comments about this:

- It's *very* important to keep track of our training and validation data sets.
- We set the `attrition`[3] reference level to 1 so that our statistical model predict those customers who are leaving, not those who are staying.
- We've listed external deposit recency (`depRecency`), age tier (`ageTier`), length of relationship (`lom`), number of products (`nProducts`), and number of customer service calls (`calls`) as our explanatory variables for this particular model.

If we want to apply this model to our test data set, we just change the `scoreData` parameter:

```
%bagging( data=trainingData, y=attrition, numx=depRecency lom nProducts calls,
  catx=ageTier, ntrees=50, id=id_member, scoreData=testData );
```

Finally, when we're done and want to make forecasts of the entire data set, we change the `scoreData` parameter once again:

```
%bagging( data=trainingData, y=attrition, numx=depRecency lom nProducts calls,
  catx=ageTier, ntrees=50, id=id_member, scoreData=inputData );
```

To compare different models with the validation set, we use *gain charts*, *lift charts*, and *K-S charts* as described in the next section (as originally described in Derby (2013)).

[3] The variable `attrition` is an indicator variable equal to 1 if the customer closed his/her account 2-3 months in the future and 0 otherwise. The outcome of our model gives the probability that this variable is equal to 1.

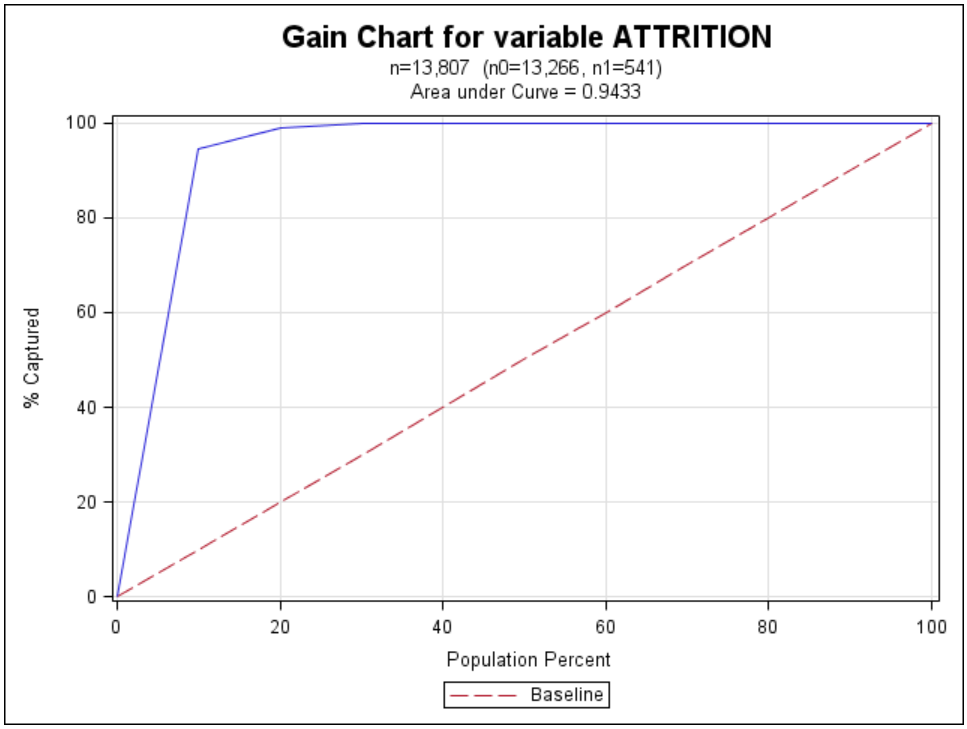


Figure 4: A gain chart.

GAIN CHARTS

A *gain chart* illustrates the effectiveness of a classification model calculated as the ratio between the results obtained with and without the model. Suppose we ordered our cases by the scores (in our case, the attrition risk).

- If we take the top 10% of our model results, what percentage of actual positive values would we get?

In our example,

- If we take the top 10% of our model results, what percentage of actual customers who left would we get?

If we then do this for 20%, 30%, etc., and then graph them, we get a gain chart as in Figure 4.[4]

For a baseline comparison, let's now order our cases (i.e., customers who closed their accounts) at random. On average, if we take the top 10% of our results, we should expect to capture about 10% of our actual positive values. If we do this for all deciles, we get the straight baseline in Figure 4. If our model is any good, it should certainly be expected to do better than that! As such, the chart for our model (the solid line) should be above the dotted line.

How do we use this chart to assess our model? In general, the better our model, the steeper the solid line in our gain chart. this is commonly reflected in two statistical measurements:

- *The area under the curve*: The better the model, the closer the area under the curve is to 1. In our case (Figure 4), we have 94.33% (which is unusually high mainly because we simulated the data).
- *The 40% measure*: What percentage of our actual targets are captured by our top 40% of predicted values? In our case (Figure 4), we have 100% (which again is unusually high in this case).

In practice, these measures don't mean very much unless we compare them to measures from other models. Indeed, some phenomena are easier to predict than others.

[4] We could do this for *any* percentile, but it's typically just done for deciles.

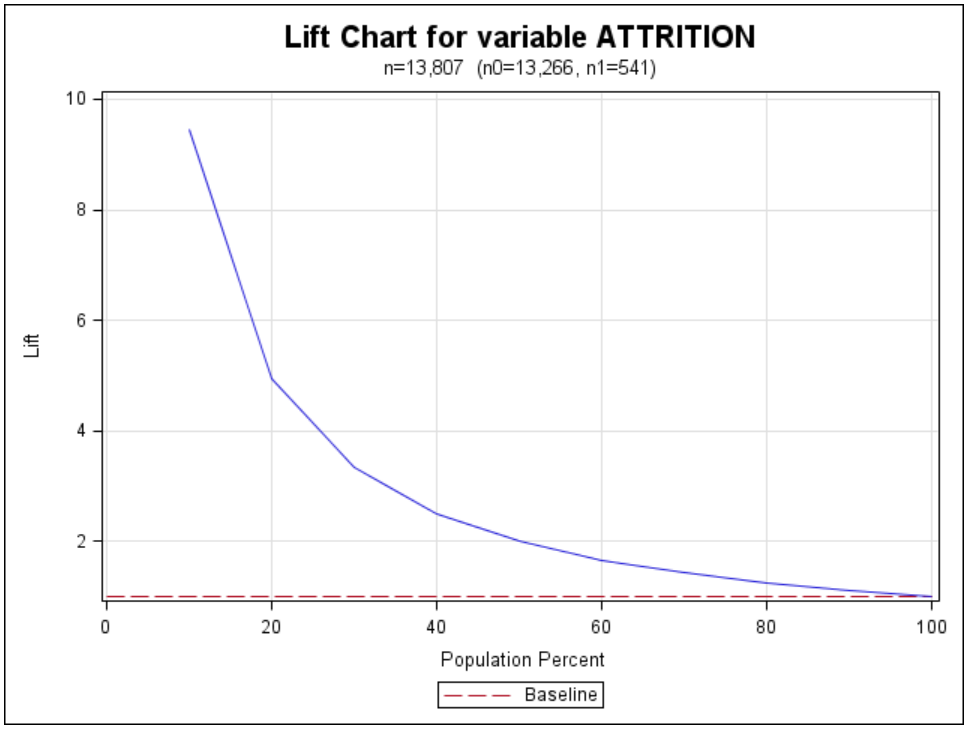


Figure 5: A lift chart.

How did we make the chart in Figure 4? SAS Institute (2011) gives us a macro that makes it relatively easy to create a gain chart. We created our own macro based on that one but creating some more details, as shown in Figure 4:

```
%makeCharts( INDATA=validationForecasts, RESPONSE=attrition, P=p_1, EVENT=1,
  GROUPS=10, PLOT=gain, PATH=&outroot, FILENAME=Gain Chart );
```

The parameters are the following:

- `INDATA`: The input data set. (Optional: default is `_last_`, the last created data set)
- `RESPONSE`: The response variable.
- `P`: The probability/score variable.
- `EVENT`: Is an event defined when the `RESPONSE` variable is 0 or 1? (Optional: default is `1`)
- `GROUPS`: Do we want to break the data down in groups of ten (at 10% intervals) or twenty (at 5% intervals)? (Optional: default is 20)
- `PLOT`: What graph do we want to plot? Three options (all described in this paper): `gain`, `lift` or `ks`.
- `PATH`: The path of the resulting graph (as a PNG file).
- `FILENAME`: The name of the resulting graph (as a PNG file). (Optional: default is `output`)

LIFT CHART

A *lift chart* simply looks at the ratio of the gain chart results with our model and with the baseline – i.e., the ratio of the solid line to the dotted line. This is shown in Figure 5. Using our same macro from before, we have the following syntax in SAS:

```
%makeCharts( INDATA=validationForecasts, RESPONSE=attrition, P=p_1, EVENT=1,
  GROUPS=10, PLOT=lift, PATH=&outroot, FILENAME=Lift Chart );
```

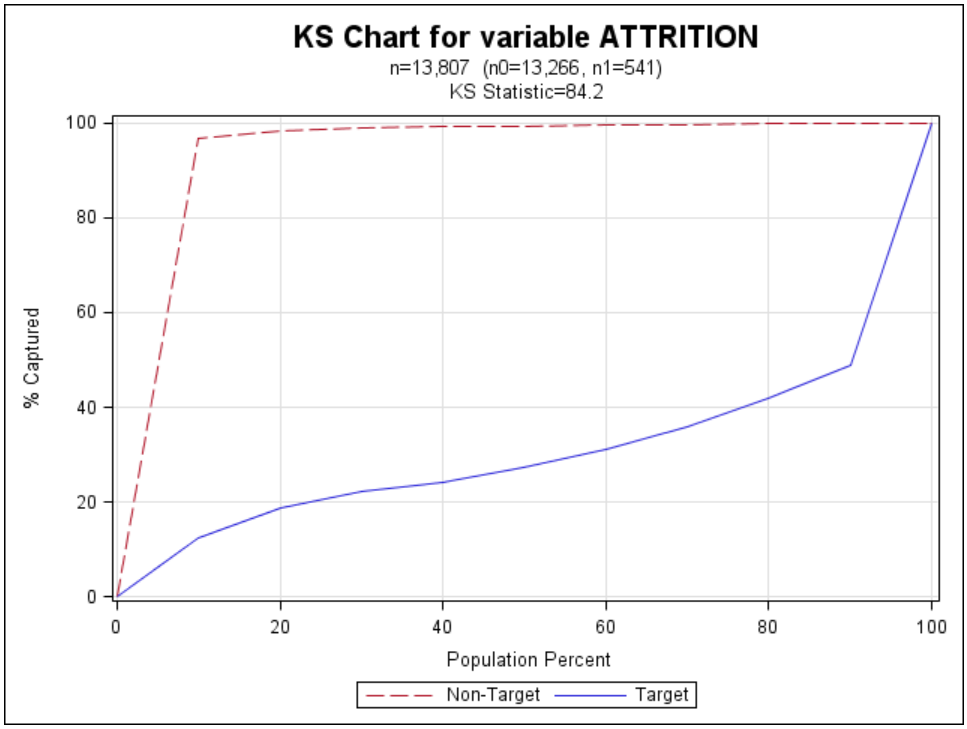


Figure 6: A K-S chart.

K-S CHART

A *K-S Chart* or *Kolmogorov-Smirnov Chart* measures the model performance in a slightly different way:

- If we look at cases with a target probability below 10%, what percentage of actual targets and non-targets would we get?

For our specific situation,

- If we look at customers with an attrition risk below 10%, what percentage of actual customers who left and customers who stayed would we get?

We then look at this for every decile, giving us the graph in Figure 6. We have two lines: For the target (attrition customers) and for the non-target (non-attrition customers). Depending on how we defined our target (e.g., whether we defined attrition as 1 or 0), the target line will be either above or below the non-target line. What's important is their maximal distance away: If we can find a probability cutoff point[5] that maximizes the difference between the targets and non-targets, we should use that one for optimal results. This maximal distance is called the *K-S statistic* or *Kolmogorov-Smirnov statistic*, which is 84.20 for our data (as before, unusually high because we simulated our data). The higher the K-S statistic, the better the model.

To code this in SAS (and generate Figure 6), we use our same macro from before:

```
%makeCharts( INDATA=validationForecasts, RESPONSE=attrition, P=p_1, EVENT=1,
  GROUPS=10, PLOT=ks, PATH=&outroot, FILENAME=KS Chart );
```

[5] i.e., if the probability cutoff point is x, then we predict that a customer will close his/her account if his/her attrition risk is above x and predict that he/she won't close his/her account otherwise. The best cutoff point isn't always 50%!

OVERALL

Overall, after fitting a large number of different models,we pick the model that gives us the best results (highest area under the curve for the gain chart, or the highest K-S statistic). If there's disagreement among the results (i.e., one model has the largest area under the curve for the gain chart but another has the highest K-S statistic), we can make an educated judgment call.

RESULTS

NO PREDICTIVE MODEL

Compared to the regression approach in Derby and Keintz (2016), one important difference with machine learning techniques like this is that *there is no predictive model*. More accurately, there is just a set of forecasts for different ranges of values of the input variables. There is little reason to write this down as some kind of formula, since it would be quite complicated and there may not be much of a mathematical relationship between the different ranges of the input variables. Indeed, this is a strength of the machine learning approach: The algorithms focus exclusively on maximizing the accuracy of the forecasts. The cost is that there is no model that lends itself to interpretation.

FORECASTS

Our final predictive scores assigns an attrition risk to each customer. We can then order these customers by that risk so that the customers with the highest attrition risk are on the top. Furthermore, we can add the customer's aggregate balance, plus all the explanatory variables from the predictive model. This will give us a measure of customer value[6], plus perhaps give us indications why each customer is at risk.

These results are shown on Figure 7. Here, all aggregate balances more than $10,000 are shown in red, highlighting our more valuable customers that we should pay attention to. Since we don't have a predictive model that we can interpret, we don't have indications of which explanatory variables have the strongest effect on that score. Still, we list these variables to give us some indications of what might be the case:

- *Age Tier*: Some age tiers are susceptible to closing an account due to life changes, such as after college graduation or upon retirement.
- *Length of Relationship (Months)*: Customers with smaller relationship lengths may have a higher attrition risk.
- *External Deposit Recency (Weeks)*: Longer external deposit recencies may have a higher attrition risk.
- *Number of Products*: Customers with a fewer number of products can have a higher attrition risk since they can close an account more easily.
- *Customer Service Calls (past month)*: Customers with more calls might have a higher attrition risk.

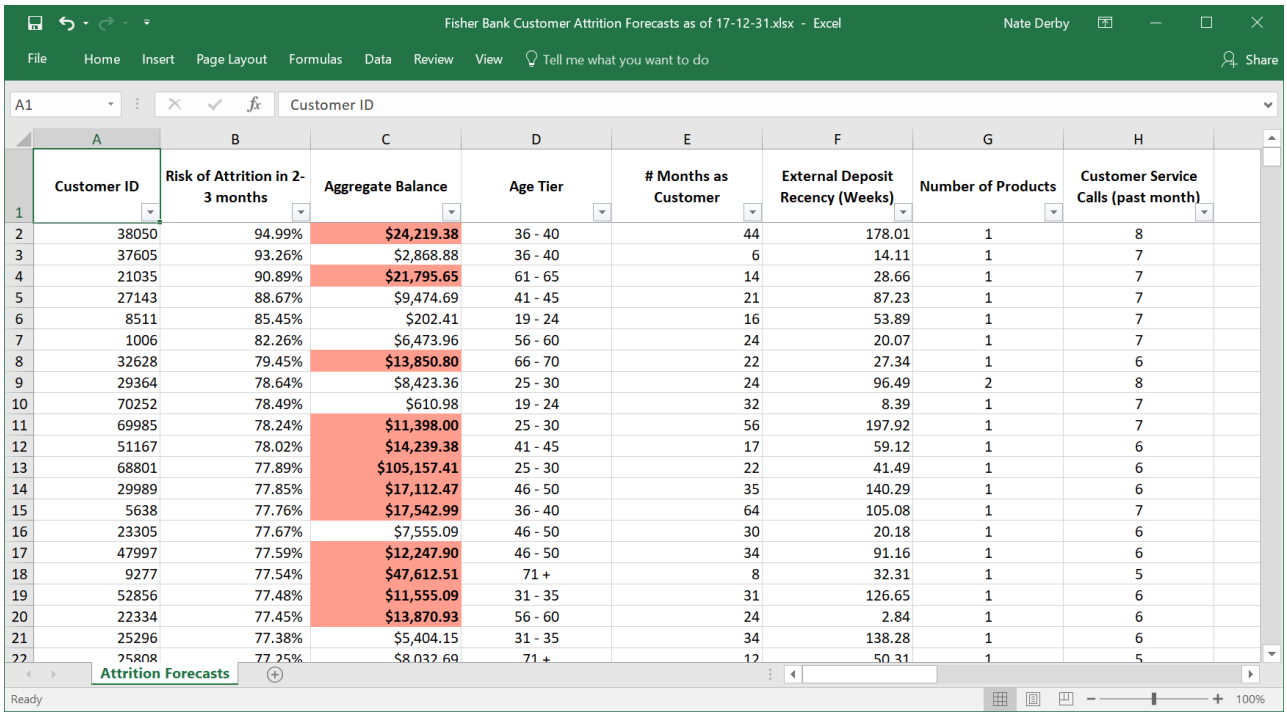
Overall, **the forecasts in Figure 7 give us a very convenient way to quickly focus on our high-risk, high-value customers mentioned in Figure 1**.

CAVEATS

These results are based on statistical approximations and standard assumptions from the data and not at all guaranteed. If some external unforeseen circumstance happens (such as, say, a competitor running a major marketing campaign), these results may not hold.

The forecasts listed in Figure 7 are only half of our retention strategy. **Nothing will happen unless these forecasts are paired with an effective intervention strategy** to change the decision of our customers at high risk of attrition. Finding the optimal strategy for this purpose is one of our goals for the future.

[6] This isn't as good as the customer lifetime value shown in Figure 1, but it's something we can quickly calculate. We can incorporate a more robust estimate of a customer lifetime value later on.



Fisher Bank Customer Attrition Forecasts as of 17-12-31.xlsx - Excel

Nate Derby

File Home Insert Page Layout Formulas Data Review View Tell me what you want to do Share

A1 Customer ID

	A	B	C	D	E	F	G	H
1	Customer ID	Risk of Attrition in 2-3 months	Aggregate Balance	Age Tier	# Months as Customer	External Deposit Recency (Weeks)	Number of Products	Customer Service Calls (past month)
2	38050	94.99%	$24,219.38	36 - 40	44	178.01	1	8
3	37605	93.26%	$2,868.88	36 - 40	6	14.11	1	7
4	21035	90.89%	$21,795.65	61 - 65	14	28.66	1	7
5	27143	88.67%	$9,474.69	41 - 45	21	87.23	1	7
6	8511	85.45%	$202.41	19 - 24	16	53.89	1	7
7	1006	82.26%	$6,473.96	56 - 60	24	20.07	1	7
8	32628	79.45%	$13,850.80	66 - 70	22	27.34	1	6
9	29364	78.64%	$8,423.36	25 - 30	24	96.49	2	8
10	70252	78.49%	$610.98	19 - 24	32	8.39	1	7
11	69985	78.24%	$11,398.00	25 - 30	56	197.92	1	7
12	51167	78.02%	$14,239.38	41 - 45	17	59.12	1	6
13	68801	77.89%	$105,157.41	25 - 30	22	41.49	1	6
14	29989	77.85%	$17,112.47	46 - 50	35	140.29	1	6
15	5638	77.76%	$17,542.99	36 - 40	64	105.08	1	7
16	23305	77.67%	$7,555.09	46 - 50	30	20.18	1	6
17	47997	77.59%	$12,247.90	46 - 50	34	91.16	1	6
18	9277	77.54%	$47,612.51	71 +	8	32.31	1	5
19	52856	77.48%	$11,555.09	31 - 35	31	126.65	1	6
20	22334	77.45%	$13,870.93	56 - 60	24	2.84	1	6
21	25296	77.38%	$5,404.15	31 - 35	34	138.28	1	6
22	25808	77.25%	$8,032.69	71 +	12	50.31	1	5

Attrition Forecasts

Ready 100%

Figure 7: Customer attrition forecasts from our final model.

CONCLUSIONS AND FURTHER SUGGESTIONS

Predictive analytics can be a very powerful tool for customer retention. A relatively simple procedure was presented in an early paper, but that doesn't work as effectively as the more complex machine learning techniques mentioned in this paper. Using this technique, we can effectively estimate which customers have a high attrition risk so that we can focus on them. However, this needs to be paired with an effective intervention strategy to be effective! Predicting alone won't solve anything.

However, **the techniques in this paper just scratch the surface of how we can reduce customer attrition with machine learning predictive techniques!** As machine learning is becoming much more widespread, more tools are becoming available. Correa et al. (2013) introduce *gradient boosting* to the decision tree approach we used here for estimating credit risk, and and Edwards et al. (2017) introduce an algorithm that extends that approach.

This approach can also be used to predict other aspects of customer behavior, such as when a customer will buy a car or a home (and thus need a car loan or a mortgage), as mentioned in Derby (2017). As such, **we can also use these techniques to further cultivate our customers** by marketing additional products to them at just the right time.

REFERENCES

Correa, A., González, A. and Amézquita, D. (2013), Using the boosting technique to improve the predictive power of a credit risk model, *Proceedings of the 2013 SAS Global Forum*, paper 093-2013.
http://support.sas.com/resources/papers/proceedings13/093-2013.pdf

Derby, N. (2013), Managing and monitoring statistical models, *Proceedings of the 2013 SAS Global Forum*, paper 190-2013.
http://support.sas.com/resources/papers/proceedings13/190-2013.pdf

Derby, N. (2017), Maximizing cross-sell opportunities with predictive analytics for financial institutions, *Proceedings of the 2017 SAS Global Forum*, paper 941-2017.
http://support.sas.com/resources/papers/proceedings17/0941-2017.pdf

Derby, N. and Keintz, M. (2016), Reducing credit union member attrition with predictive analytics, *Proceedings of the 2016 SAS Global Forum*.
http://support.sas.com/resources/papers/proceedings16/11882-2016.pdf

Edwards, P., Duhon, D. and Shergill, S. (2017), Real AdaBoost: Boosting for credit scorecards and similarity to woe logistic regression, *Proceedings of the 2017 SAS Global Forum*, paper 1323-2017.
http://support.sas.com/resources/papers/proceedings17/1323-2017.pdf

Fader, P. (2012), *Customer Centricity: Focus on the Right Customers for Strategic Advantage*, Wharton Digital Press, Philadelphia, PA.

Karp, A. (1998), Using logistic regression to predict customer retention, *Proceedings of the Eleventh Northeast SAS Users Group Conference*.
http://www.lexjansen.com/nesug/nesug98/solu/p095.pdf

Liu, W. (2012), A SAS macro for bootstrap aggregating (bagging).
https://statcompute.wordpress.com/2012/07/14/
a-sas-macro-for-bootstrap-aggregating-bagging

Lu, J. (2002), Predicting customer churn in the telecommunications industry – an application of survival analysis modeling using SAS, *Proceedings of the Twenty-Seventh SAS Users Group International Conference*, paper 114-27.
http://www2.sas.com/proceedings/sugi27/p114-27.pdf

Lu, J. (2003), Modeling customer lifetime value using survival analysis – an application in the telecommunications industry, *Proceedings of the Twenty-Eighth SAS Users Group International Conference*, paper 120-28.
http://www2.sas.com/proceedings/sugi28/120-28.pdf

SAS Institute (2011), Gains and lift plots for binary-response models.
http://support.sas.com/kb/41/683.html

Su, J., Cooper, K., Robinson, T. and Jordan, B. (2009), Customer retention predictive modeling in the healthcare insurance industry, *Proceedings of the Seventeenth SouthEast SAS Users Group Conference*, paper AD-007.
http://analytics.ncsu.edu/sesug/2009/AD007.Su.pdf

Thomas, W. (2010), Improving retention by predicting both who and why, *Proceedings of the Twenty-Third Northeast SAS Users Group Conference*.
http://www.lexjansen.com/nesug/nesug10/sa/sa08.pdf

ACKNOWLEDGEMENTS

We thank our good friends at many of the financial institutions we've worked with for their help and cooperation!

CONTACT INFORMATION

Comments and questions are valued and encouraged, as we depend on partnering banks to do our research. As such, please don't hesitate to contact me:

Nate Derby
Stakana Analytics
815 First Ave., Suite 287
Seattle, WA 98104-1404

nderby@stakana.com
http://stakana.com
http://nderby.org