Increasing Profits Through Good Customers

STAT 551 Final Project

Yuchi Hu

April 21, 2019

Contents

1. Introduction

The data for this project comes from RetentionDataRaw.xlsx, which we have already used extensively in the Midterm. The data is composed of credit card monthly billing statements from February 2010 to November 2010 (shorter time frames for closed accounts). The task now is to focus on the profitable or good customers. We want to give incentives to these customers in the form of, say, credit line increases or annual fee waivers to encourage them to use their credit cards more, thereby increasing our profit. Before we can do any modeling, we need to create the "model dataset" from the retention data. We will use the model dataset to build and evaluate two models: logistic regression and multivariate adaptive regression splines (MARS). We then compare these models' performances to that of the *Good Customer Score*. Lastly, we calculate profitability in the models' gains tables.

2. Creating the Model Dataset

The retention data originally has 97,465 rows and 26 columns. After removing the rows with missing ID's, we are left with 91,502 rows representing 9,997 customers. We should note that when we refer to the number of customers, we are really referring to the number of unique ID's since a customer could have multiple ID's if they have multiple accounts.

As the first step in creating the model dataset, we need to remove customers that we deem to be too risky. We also need to define "good" and "bad" customers.

2.1 Remove Risky Customers

We found out from the Midterm that some customers are too risky to give more money to, so we will simply remove them from the data. The customer is deemed too risky if their *Row Num* = 1 (month 1) satisfies any of the following conditions:

- *Days Deliq* > 0 (more than 0 days delinquent)
- non-blank *External Status* (a blank *External Status* corresponds to an open account)
- *Opening Balance* $>$ *Credit Limit*
- *Ending Balance* $>$ *Credit Limit*

We find that 4,167 of the 9,997 customers are too risky, so we are left with 5,830 customers after removing the risky customers. Overall, we are left with 55,895 rows of data.

2.2 Define Good and Bad Customers

Next, for the remaining customers, we create *Bad* (the outcome variable) to define whether they are good (*Bad*=0) or bad (*Bad*=1) customers. A customer is defined as bad if they satisfy any of the following conditions:

- *Days Deliq* ≥ 90 (90 or more days delinquent) in the final month
- *External Status* other than blank (open account) or "C" (closed account) in month 7 or later

We find that 610 customers satisfy the first criterion and 669 customers satisfy the second. (There are some customers who satisfy both criteria.) Customers who do not satisfy any of the two criteria are defined as good.

2.3 Keep Only *Row Num* = 1 for Each Customer

After removing the risky customers and defining *Bad* for the remaining customers, the next step in the creation of the model dataset is to keep only *Row Num* = 1 (month 1) for each customer. Interestingly, the customer with ID=1695646 has a duplicate row, which we remove. The final number of rows for the model dataset is 5,825, with one row per customer.

2.4 Input Variables

We can remove the following variables from the model dataset:

- *Row Num* – this is 1 for every customer
- *External Status* – this is blank (open account) for every customer
- *Days Deliq* and *Over limit Amount* – these are 0 for every customer
- *Good Customer Score*, *Behavior Score* and *Quarterly Fico Score* – these are specifically prohibited from the analysis
- *ClosureReason* – we will create a variable to deal with the 36 factor levels (see below)
- *Open Date*, *Last Statement Date*, *Cycle Date*, *Month End Date*, and *Last Payment Date* – these are dates

The remaining input variables are:

- *Months On Book*
- *Credit Limit*
- *Opening Balance*
- *Ending Balance*
- *Actual Min Pay Due*
- *Total Min Pay Due*
- *Net Payments During Cycle*
- *Net Purchases During Cycle*
- *Net Cash Advances During Cycle*
- *Net Premier Fees Billed During Cycle*
- *Net Behavior Fees Billed During Cycle*
- *Net Concessions Billed During Cycle*

In addition, we create the following input variables:

- *Opening Utilization = Opening Balance / Credit Limit*
- *Ending Utilization = Ending Balance / Credit Limit*
- *Utilization Difference = Ending Utilization - Opening Utilization*
- *Total Fees and Concessions = Net Premier Fees Billed During Cycle + Net Behavior Fees Billed During Cycle - Net Concessions Billed During Cycle*
- *Payment Type* = index of whether a customer paid above, below, or exactly their total minimum payment due or if they made no payment
- *Closure Reason Given* = index of whether or not a closure reason was given

3. Exploratory Data Analysis on the Model Dataset

Now that we have created the model dataset, we can perform exploratory data analysis on it. The model dataset is composed of 5,825 rows (number of customers) and 20 columns (18 inputs + outcome variable + ID).

3.1 Bar Charts and Histograms

First, let's look at the distributions of the levels of the categorical variables.

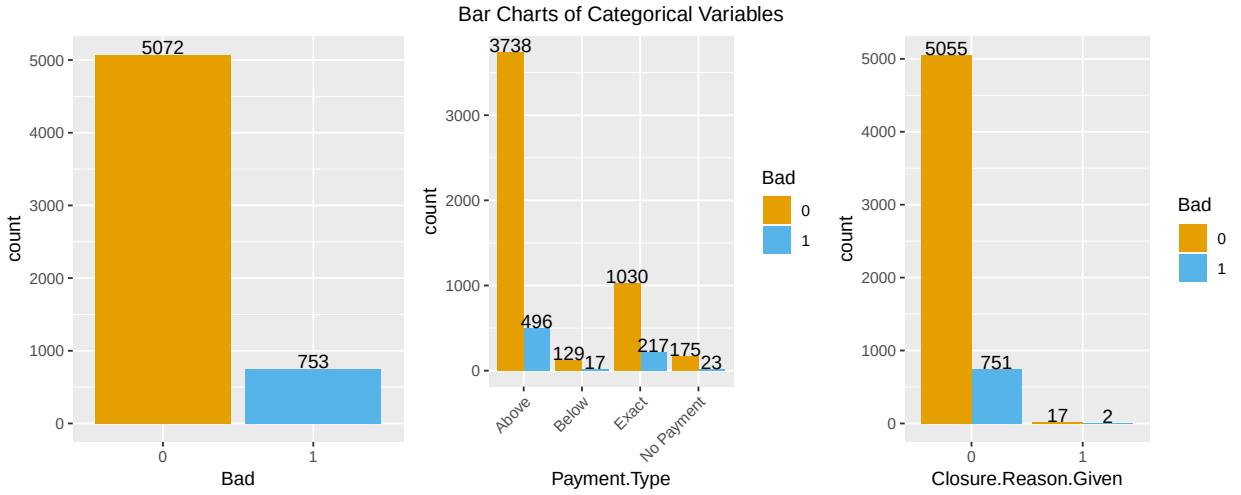

Figure 1: Bar charts of the categorical variables (good=orange, bad=blue).

From **Figure 1**, we see that there are 5,072 (87%) good customers and 753 (13%) bad customers. In terms of *Payment Type*, we see that most customers either paid above or exactly their total minimum payment due, which is good since this is an indication that they are good customers. In terms of *Closure Reason Given*, 19 customers have a reason listed for their account closure, which is odd since all of the customers left in the model dataset have a blank *External Status* or open account. This tells us that perhaps these 19 accounts should've been marked as closed (*External Status* of "C").

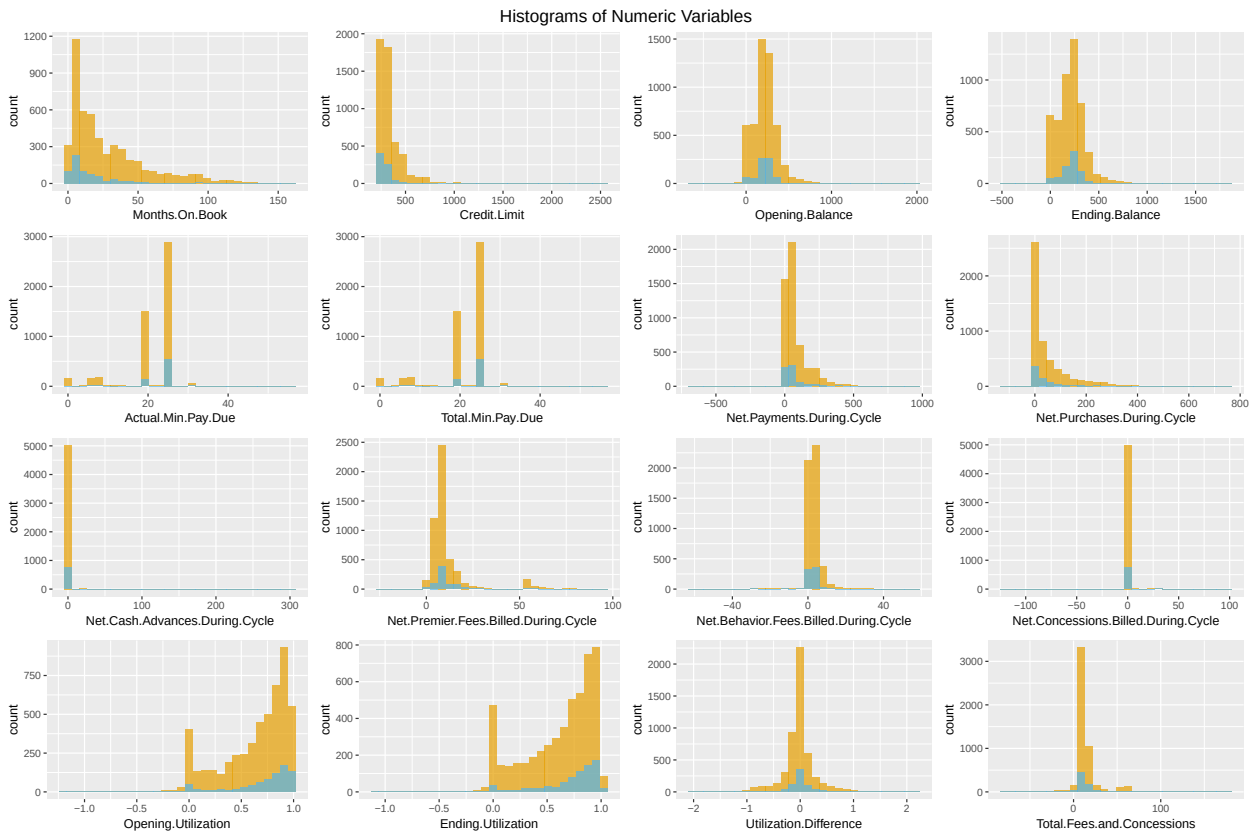

Figure 2: Histograms of the numeric variables grouped by Bad (good=orange, bad=blue).

Figure 2 shows the histograms of the numeric variables grouped by *Bad*. We see that good and bad customers have similar distributions for each numeric variable.

3.2 Correlation Matrix

Figure 3 shows the correlation matrix of the numeric variables in the model dataset. We see that many of the variables are correlated.

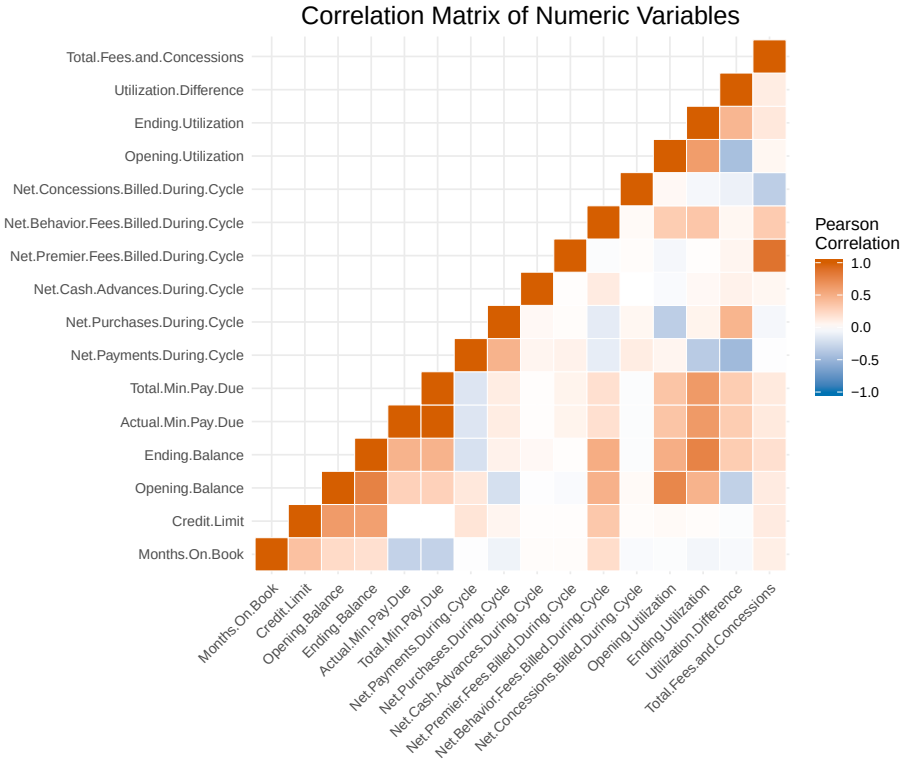

Figure 3: Correlation matrix of the numeric variables.

3.3 Binning

Using **BinProfet()** from the **Rprofet** package, we bin four variables: *Total Min Pay Due*, *Net Payments*, *Net Purchases*, and *Net Premier Fees*. We choose these variables since they appear to contain outliers, whose effect we want to minimize. These binned variables will replace their original counterparts as inputs for the models. The bar charts of the four binned variables are shown in **Figure 4**.

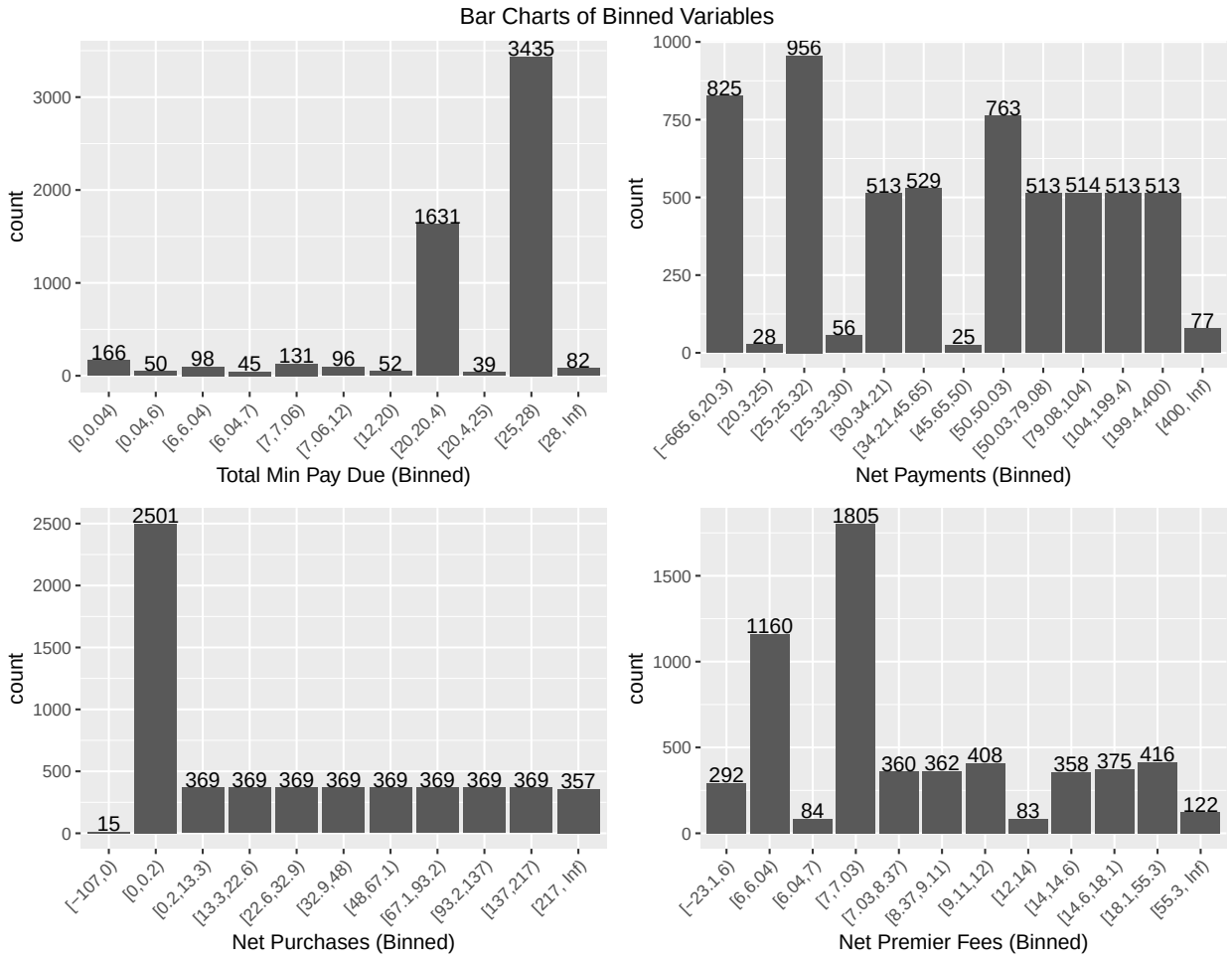

Figure 4: Bar charts of the binned Total Min Pay Due, Net Payments, Net Purchases, and Net Premier Fees.

4. The Models

Before building the models, we use **createDataPartition()** from the **caret** package to split the model dataset into training (70%; 4,079 observations) and validation sets (30%; 1,746 observations). This function partitions the data while maintaining the class ratios (stratified sampling).

4.1 Logistic Regression

We build a logistic regression model with *Bad* as the outcome and the 18 variables mentioned in **Section 2.4** as the inputs (remember that the four binned variables replace their original counterparts). We perform stepwise variable selection using **StepAIC** from the **MASS** package. The summary of the final logistic regression model is below:

```
##
## Call:
## glm(formula = Bad ~ Months.On.Book + Credit.Limit + Opening.Balance +
##     Net.Concessions.Billed.During.Cycle + Opening.Utilization +
```

```
##     Ending.Utilization + Payment.Type + binned.total.min.pay.due,
##     family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0705  -0.5858  -0.4716  -0.2969   3.6067
##
## Coefficients:
##                                    Estimate Std. Error z value
## (Intercept)                        0.140023   0.755254   0.185
## Months.On.Book                    -0.012095   0.002359  -5.127
## Credit.Limit                      -0.010072   0.002308  -4.364
## Opening.Balance                    0.007881   0.002799   2.816
## Net.Concessions.Billed.During.Cycle  0.016006   0.008885   1.801
## Opening.Utilization               -2.144879   0.826150  -2.596
## Ending.Utilization                 1.236938   0.224886   5.500
## Payment.TypeBelow                  0.296425   0.349570   0.848
## Payment.TypeExact                  0.237203   0.111395   2.129
## Payment.TypeNo Payment            -0.592155   0.290655  -2.037
## binned.total.min.pay.due[0.04,6)   0.530616   0.592388   0.896
## binned.total.min.pay.due[6,6.04)   0.454507   0.494759   0.919
## binned.total.min.pay.due[6.04,7)  -0.945968   1.076110  -0.879
## binned.total.min.pay.due[7,7.06)  -0.106841   0.482375  -0.221
## binned.total.min.pay.due[7.06,12)  0.114345   0.520605   0.220
## binned.total.min.pay.due[12,20)    0.848855   0.522757   1.624
## binned.total.min.pay.due[20,20.4)  0.584470   0.338922   1.724
## binned.total.min.pay.due[20.4,25)  0.704389   0.608181   1.158
## binned.total.min.pay.due[25,28)    0.100260   0.331626   0.302
## binned.total.min.pay.due[28, Inf)  0.642282   0.474519   1.354
##                                       Pr(>|z|)
## (Intercept)                            0.85292
## Months.On.Book                    0.0000002950 ***
## Credit.Limit                      0.0000127435 ***
## Opening.Balance                        0.00487 **
## Net.Concessions.Billed.During.Cycle    0.07165 .
## Opening.Utilization                    0.00943 **
## Ending.Utilization                0.0000000379 ***
## Payment.TypeBelow                      0.39645
## Payment.TypeExact                      0.03322 *
## Payment.TypeNo Payment                 0.04162 *
## binned.total.min.pay.due[0.04,6)       0.37040
## binned.total.min.pay.due[6,6.04)       0.35828
## binned.total.min.pay.due[6.04,7)       0.37937
## binned.total.min.pay.due[7,7.06)       0.82471
## binned.total.min.pay.due[7.06,12)      0.82615
## binned.total.min.pay.due[12,20)        0.10442
## binned.total.min.pay.due[20,20.4)      0.08462 .
## binned.total.min.pay.due[20.4,25)      0.24679
## binned.total.min.pay.due[25,28)        0.76240
## binned.total.min.pay.due[28, Inf)      0.17588
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 3143.5  on 4078  degrees of freedom
## Residual deviance: 2933.4  on 4059  degrees of freedom
## AIC: 2973.4
##
## Number of Fisher Scoring iterations: 6
```

We see that there are 8 inputs in the final model compared to the 18 in the original model. The AIC of the final model is 2973.4.

Next, we move on to model evaluation (KS, ROC, gains table, and lift).

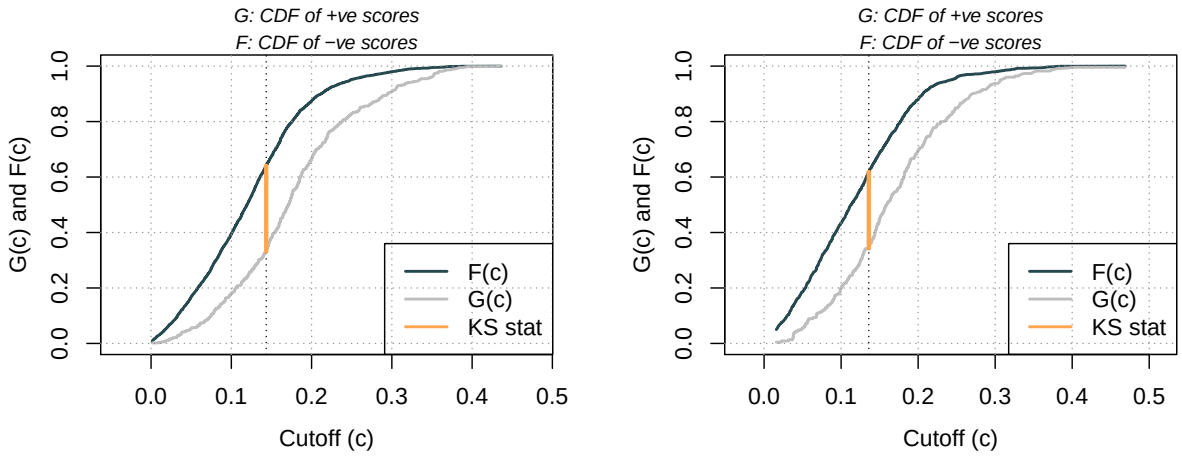

Figure 5: KS plots of the training (left) and validation sets (right) for logistic regression.

Figure 5 shows the KS plots of the training and validation sets for logistic regression.

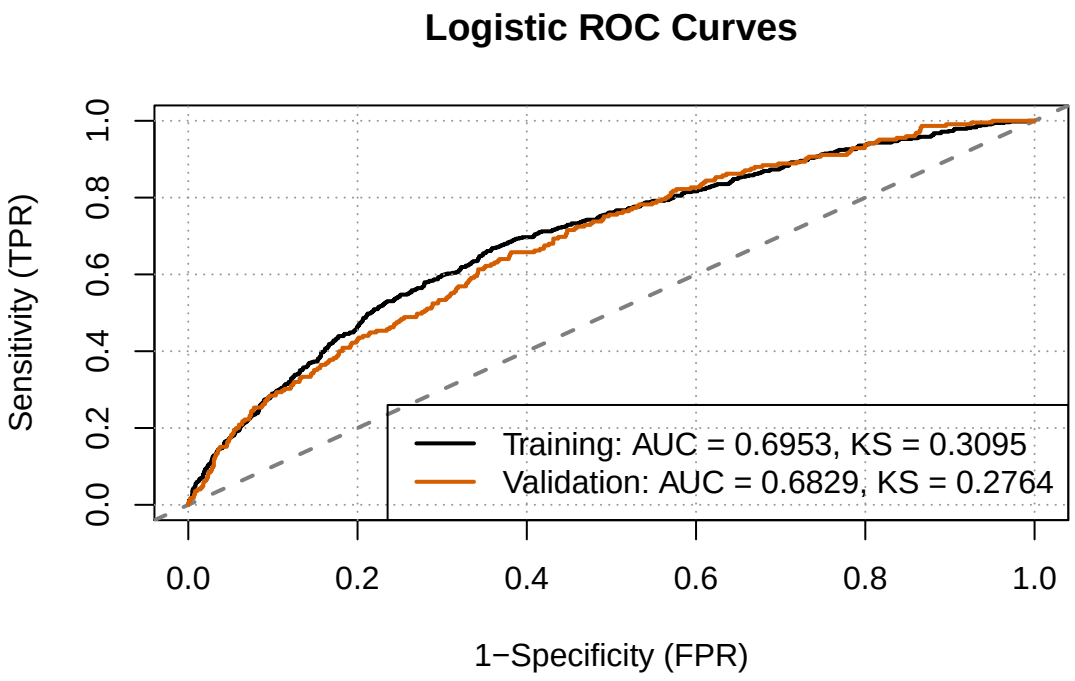

Figure 6: ROC curves of the training and validation sets for logistic regression.

Figure 6 shows the ROC curves of the training and validation sets for logistic regression. The training and validation AUC's are 0.695 and 0.683, respectively. The training and validation KS statistics are 0.310 and 0.276, respectively.

Table 1: Logistic Gains Table (Validation)

Bucket	Obs	CObs	Depth	Resp	CResp	RespRate	CRespRate	CCapRate	Lift	CLift
1	175	175	0.1	57	57	0.33	0.33	0.25	2.53	2.53
2	174	349	0.2	29	86	0.17	0.25	0.38	1.29	1.91
3	175	524	0.3	26	112	0.15	0.21	0.50	1.15	1.66
4	174	698	0.4	30	142	0.17	0.20	0.63	1.34	1.58
5	175	873	0.5	21	163	0.12	0.19	0.72	0.93	1.45
6	175	1048	0.6	18	181	0.10	0.17	0.80	0.80	1.34
7	174	1222	0.7	17	198	0.10	0.16	0.88	0.76	1.26
8	175	1397	0.8	9	207	0.05	0.15	0.92	0.40	1.15
9	174	1571	0.9	15	222	0.09	0.14	0.99	0.67	1.10
10	175	1746	1.0	3	225	0.02	0.13	1.00	0.13	1.00

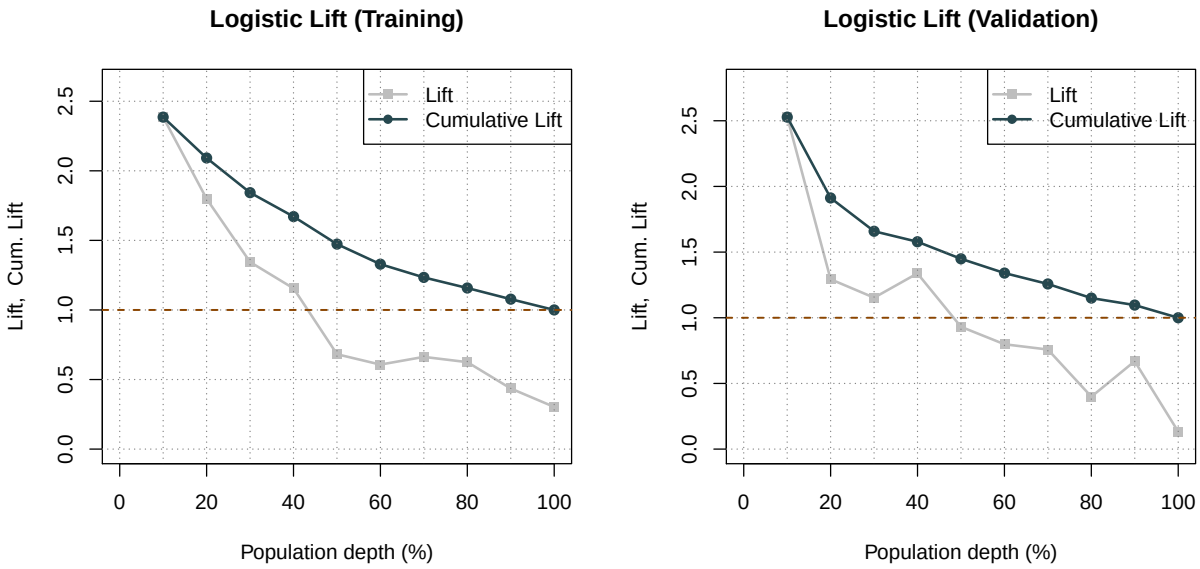

Figure 7: Lift and cumulative lift of training (left) and validation sets (right) for logistic regression.

Table 1 shows the gains table of the validation set, and **Figure 7** shows the lift and cumulative lift of the training and validation sets for logistic regression.

4.2 MARS

We build a MARS model with *Bad* as the outcome and the 18 variables mentioned in **Section 2.4** as the inputs (again, remember that the four binned variables replace their original counterparts). We set the degree of interaction to 2. The summary of the MARS model is below:

```
## Call: earth(formula=formula, data=train, glm=list(family=binomial),
##             degree=2)
##
## GLM coefficients
##                                                                          1
## (Intercept)                                                     -3.2938096
## h(400-Credit.Limit)                                             -0.0056487
## h(11-Months.On.Book) * Payment.TypeNoPayment                    -0.1678832
## h(Months.On.Book-11) * Closure.Reason.Given1                     0.1364700
## h(Ending.Utilization-0.55576) * Payment.TypeExact                0.7071500
## h(Months.On.Book-6) * h(400-Credit.Limit)                        0.0010488
## h(15-Months.On.Book) * h(400-Credit.Limit)                       0.0013528
## h(Months.On.Book-15) * h(400-Credit.Limit)                      -0.0010963
## h(11-Months.On.Book) * h(0.81902-Opening.Utilization)            0.1757581
## h(11-Months.On.Book) * h(Ending.Utilization-0.983345)            8.9088555
## h(11-Months.On.Book) * h(0.983345-Ending.Utilization)           -0.1036415
## h(400-Credit.Limit) * h(Opening.Balance-97.32)                   0.0000265
## h(400-Credit.Limit) * h(97.32-Opening.Balance)                   0.0000425
## h(400-Credit.Limit) * h(Net.Behavior.Fees.Billed.During.Cycle-0.93)  0.0002087
## h(400-Credit.Limit) * h(Net.Concessions.Billed.During.Cycle-0)   0.0002121
## h(Net.Behavior.Fees.Billed.During.Cycle-2.88) * h(0.55576-Ending.Utilization)  0.5668949
```

```
## h(Ending.Utilization-0.55576) * h(0.84875-Utilization.Difference)              2.6085585
##
## GLM (family binomial, link logit):
##  nulldev   df       dev   df   devratio     AIC iters converged
##   3143.5 4078   2842.84 4062     0.0956    2877     5         1
##
## Earth selected 17 of 33 terms, and 11 of 59 predictors
## Termination condition: RSq changed by less than 0.001 at 33 terms
## Importance: Months.On.Book, Credit.Limit, Ending.Utilization, ...
## Number of terms at each degree of interaction: 1 1 15
## Earth GCV 0.1053808    RSS 421.2505    GRSq 0.06530197    RSq 0.08354844
```

We see that MARS automatically selects variables and the knots of the hinge functions, and it selected 17 of 33 terms and 11 of 59 predictors. The AIC of the MARS model is 2877, which is lower than that of the logistic regression model (2973.4). That is, MARS performed slightly better than the logistic.

Next, we move on to model evaluation (KS, ROC, gains table, and lift).

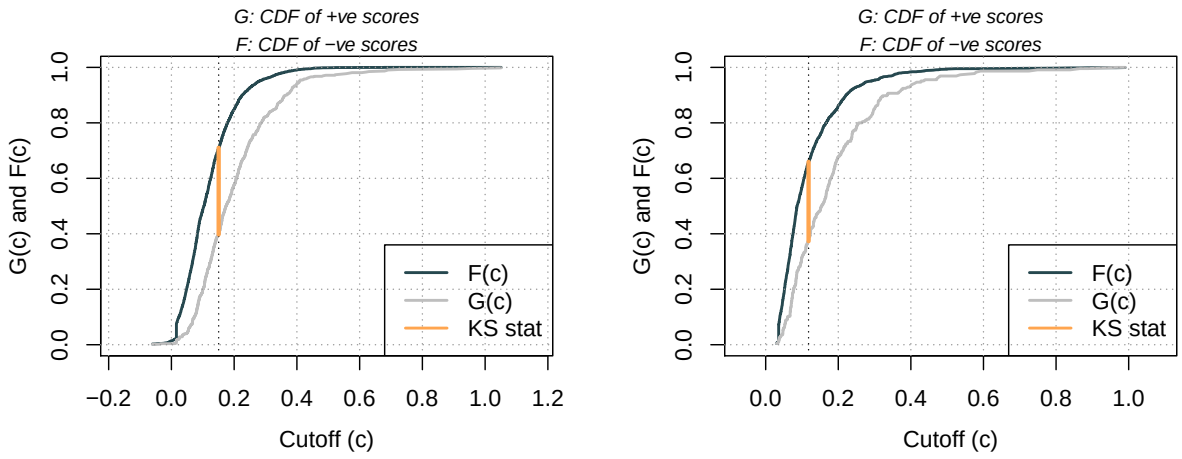

Figure 8: KS plots of the training (left) and validation sets (right) for MARS.

Figure 8 shows the KS plots of the training and validation sets for MARS.

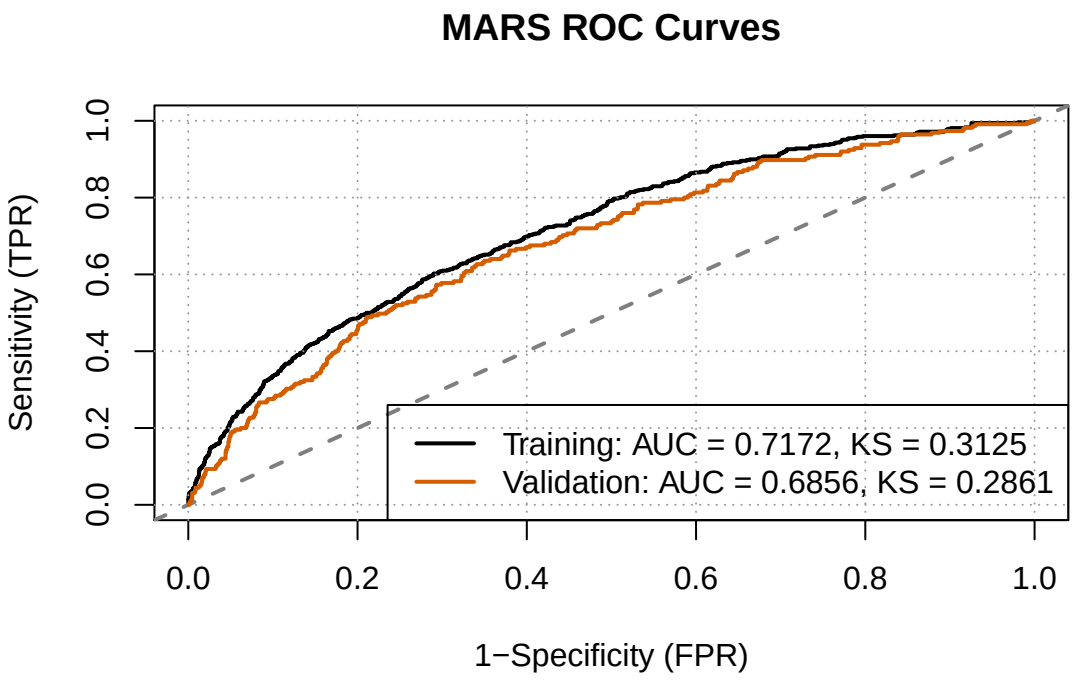

Figure 9: ROC curves of the training and validation sets for MARS.

Figure 9 shows the ROC curves of the training and validation sets for MARS. The training and validation AUC's are 0.717 and 0.686, respectively. The training and validation KS statistics are 0.313 and 0.286, respectively.

Table 2: MARS Gains Table (Validation)

Bucket	Obs	CObs	Depth	Resp	CResp	RespRate	CRespRate	CCapRate	Lift	CLift
1	175	175	0.1	54	54	0.31	0.31	0.24	2.39	2.39
2	174	349	0.2	35	89	0.20	0.26	0.40	1.56	1.98
3	175	524	0.3	30	119	0.17	0.23	0.53	1.33	1.76
4	174	698	0.4	25	144	0.14	0.21	0.64	1.11	1.60
5	175	873	0.5	18	162	0.10	0.19	0.72	0.80	1.44
6	175	1048	0.6	17	179	0.10	0.17	0.80	0.75	1.33
7	174	1222	0.7	20	199	0.11	0.16	0.88	0.89	1.26
8	175	1397	0.8	9	208	0.05	0.15	0.92	0.40	1.16
9	174	1571	0.9	11	219	0.06	0.14	0.97	0.49	1.08
10	175	1746	1.0	6	225	0.03	0.13	1.00	0.27	1.00

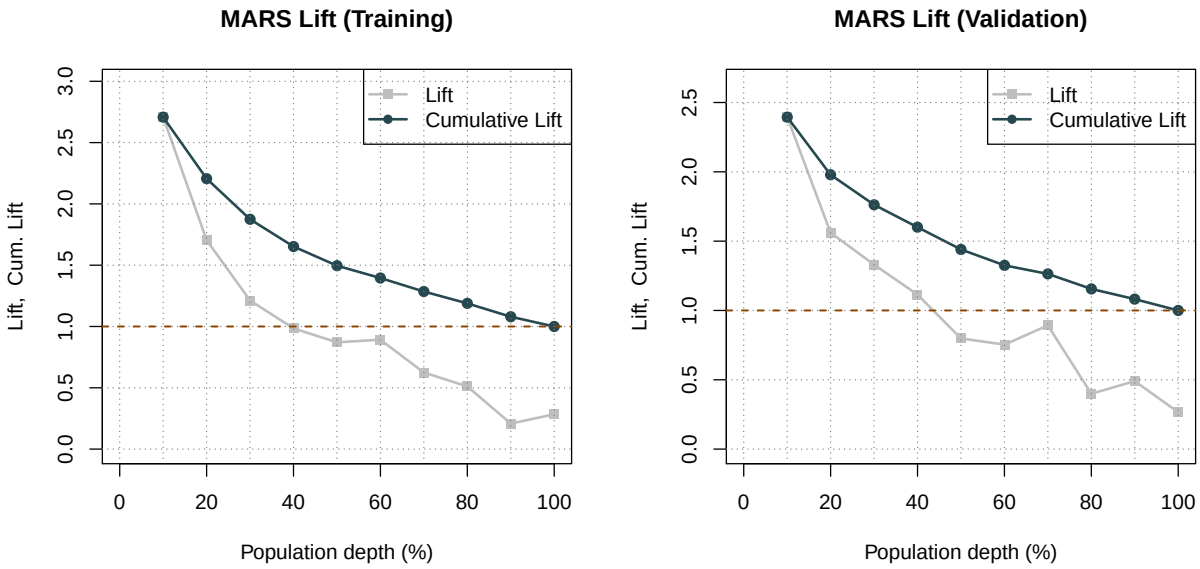

Figure 10: Lift and cumulative lift of the training (left) and validation sets (right) for MARS.

Table 2 shows the gains table of the validation set, and **Figure 10** shows the lift and cumulative lift of the training and validation sets for MARS.

4.3 Good Customer Score

Now, we compare our model to the *Good Customer Score*. This score can be considered as a model and is simply a probability or log-odds transformed into points. Since rank order is retained, we can calculate or plot the ROC/AUC, KS, gains table, and lift for *Good Customer Score*.

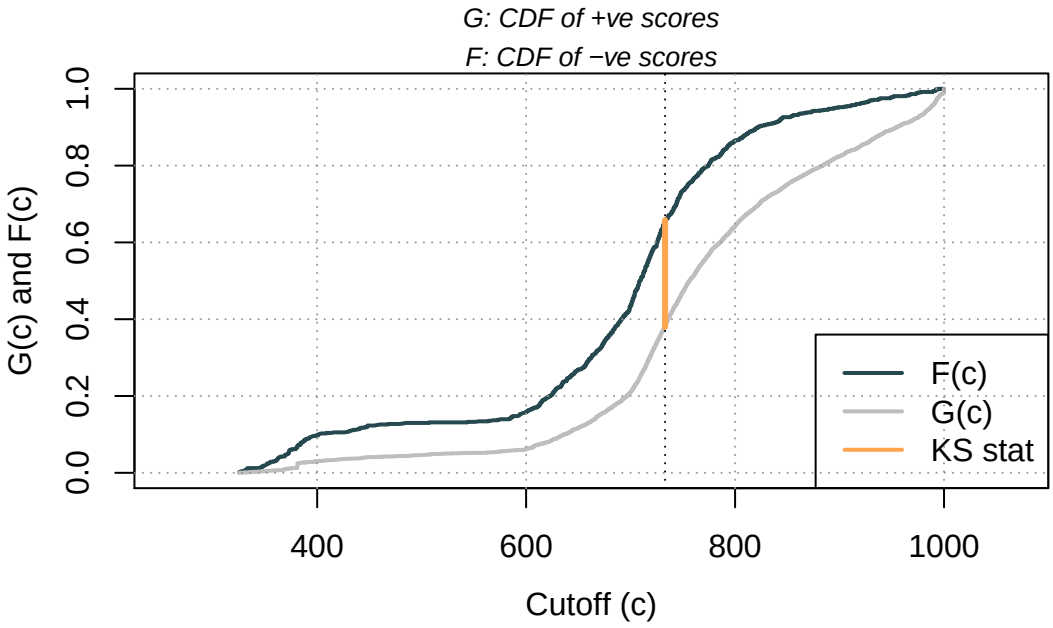

Figure 11: KS plot for Good Customer Score.

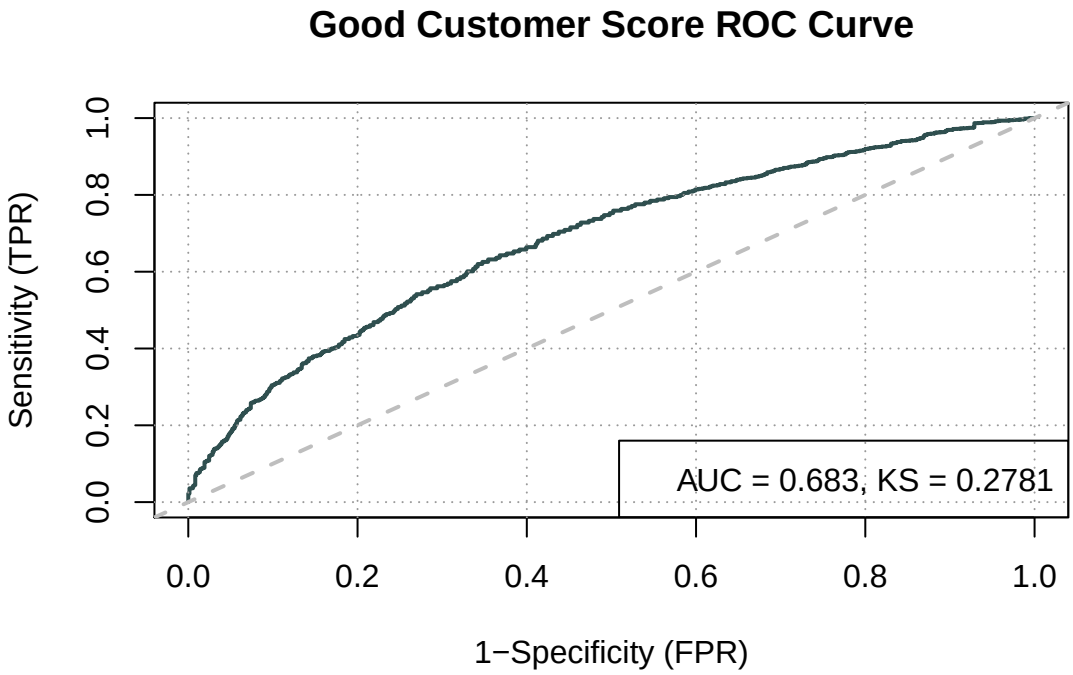

Figure 12: ROC curve for Good Customer Score.

Table 3: Good Customer Score Gains Table

Bucket	Obs	CObs	Depth	Resp	CResp	RespRate	CRespRate	CCapRate	Lift	CLift
1	554	554	0.1	536	536	0.97	0.97	0.11	1.11	1.11
2	554	1108	0.2	527	1063	0.95	0.96	0.22	1.10	1.11
3	554	1662	0.3	512	1575	0.92	0.95	0.33	1.06	1.09
4	554	2216	0.4	499	2074	0.90	0.94	0.43	1.04	1.08
5	554	2770	0.5	501	2575	0.90	0.93	0.54	1.04	1.07
6	554	3324	0.6	483	3058	0.87	0.92	0.64	1.00	1.06
7	554	3878	0.7	470	3528	0.85	0.91	0.73	0.98	1.05
8	554	4432	0.8	445	3973	0.80	0.90	0.83	0.93	1.03
9	554	4986	0.9	430	4403	0.78	0.88	0.92	0.89	1.02
10	554	5540	1.0	406	4809	0.73	0.87	1.00	0.84	1.00

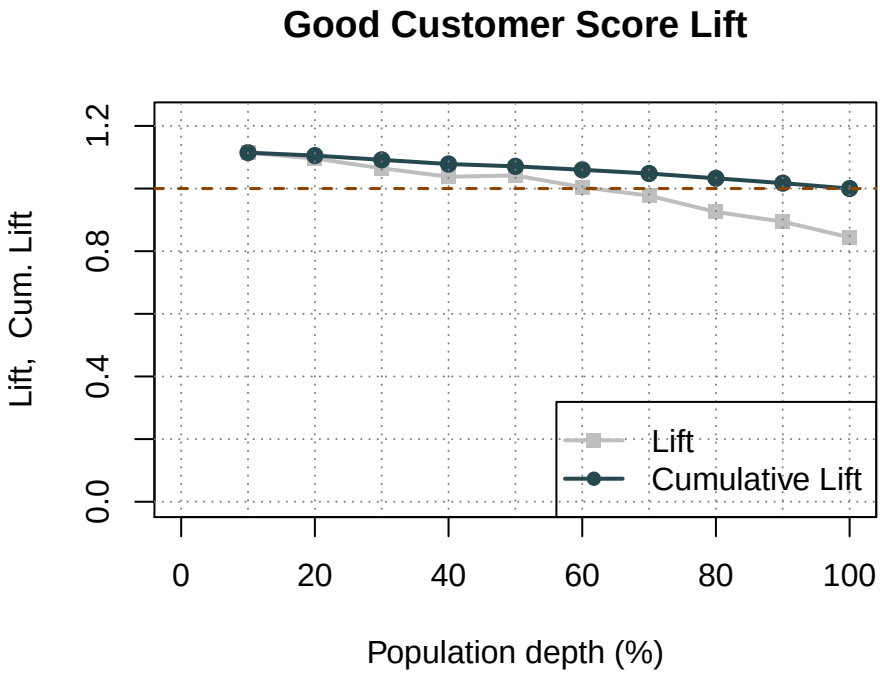

Figure 13: Lift and cumulative lift for Good Customer Score.

Table 3 shows the gains table for *Good Customer Score*. **Figures 11-13** shows the KS plot, ROC curve, and lift and cumulative lift for *Good Customer Score*. The AUC is 0.683, and the KS statistic is 0.278. A caveat here is that the **ROCit** package sorts the *Good Customer Score* by descending order, which means the good customers tend be in the top buckets. In contrast, for the logistic and MARS models, the predicted probabilities of being bad is sorted by descending order, which means the bad customers tend to be in the top buckets. **See the powerpoint for the corrected Good Customer Score gains table and lift.**

4.4 Model Comparison

Table 4 compares the AUC and KS among the logistic, MARS, and *Good Customer Score* models. We see that all three models performed similarly; in fact, MARS was able to slightly outperform *Good Customer Score*.

Table 4: Model Comparison

	AUC	KS
Logistic (Validation)	0.6829	0.2764
MARS (Validation)	0.6856	0.2861
Good Customer Score	0.6830	0.2781

5. Are We Making Money?

Finally, we use all rows of each customer in the model dataset to calculate *Net Profit*. This is a profit for a good customer and a loss for a bad customer. It is calculated as:

- For good customers ($Bad = 0$), sum up *Net Payments During Cycle* for all rows of each customer
- For bad customers ($Bad = 1$), take the *Ending Balance* from the last row of each customer. If the *Ending Balance* is 0 in the last row, take the *Ending Balance* from the second to last row.

We find that for good customers, we earn a total profit of \$3,125,741 and a profit per customer of \$616.27. For bad customers, we suffer a total loss of \$300,766.9 and a loss per customer of \$399.96. This gives us a total net profit of \$2,824,974. With this information, we can calculate the profitability in the logistic and MARS gains tables. **See the powerpoint for enhanced versions of the logistic and MARS gains tables.** We see that we are indeed making money.

6. Conclusion

The logistic and MARS models both provide decent separation of good and bad customers. Both models perform on par with the *Good Customer Score*, with the MARS model having the slight edge. We also see from the calculated profits in the logistic and MARS gains tables that we are indeed making money. A recommeded course of action based on these gains tables is to give incentives in the form of credit line increases or annual fee waivers to the good customers in the bottom buckets. This willl encourage them to use their credit cards more, thereby increasing our profit.