UNIVERSITY of VIRGINIA | DARDEN Business Publishing

UV7580
Rev. Nov. 14, 2018

Retention Modeling at Scholastic Travel Company (A) and (B)

Teaching Note

Synopsis

Scholastic Travel Company (STC) would like to use its customer profile data to predict who will purchase a travel package again in the following year—a data analytic task known as "classification." The cases present data on 2,389 customers, where the A case contains numerous profile data fields, and the B case presents the additional data on the net promoter scores (NPS). Both datasets are realistic in the extent of the "cleanness" of the data and the potential benefits and challenges in using various statistical and machine learning techniques for classification.

Objectives

These cases are an efficient vehicle for exposing students to a broad understanding of data science, machine learning, and predictive analytics, as applied to a very common problem of (binary/binomial) classification. It has been successfully used in an MBA and EMBA electives on data science, a specialized program on business analytics, and an Executive Education program on customer analytics.

The case is also effective for strengthening students' coding skills and teaching them the R programming language; see the Analyses and Pedagogy sections of this note for discussion of the author's approach. Python code is also available upon request.

This case can be supplemented with "Modeling Discrete Choice: Categorical Dependent Variables, Logistic Regression, and Maximum Likelihood Estimation" (UVA-QA-0779).[1]

The primary learning objective is to provide students with an introductory understanding of data science and machine-learning tools for classification, including the following:

- Understand the best practices in preparing data for classification (and for general predictions): the notions of training data versus testing data, representativeness of these samples, and the variables they contain.

[1] Anton Ovchinnikov, "Modeling Discrete Choice: Categorical Dependent Variables, Logistic Regression, and Maximum Likelihood Estimation," UVA-QA-0779 (Charlottesville, VA: Darden Business Publishing, 2011).

This teaching note was prepared by Anton Ovchinnikov, Distinguished Faculty Professor or Management Science and Operations Management and Scotiabank Scholar of Customer Analytics at the Smith School of Business, Queen's University, Canada.

- Understand the best practices in “cleaning” data: dealing with missing variables, rare categories, and balanced data versus unbalanced data.
- Understand the metrics of classification: confusion matrix, ROC curve, Area Under Curve (AUC) and Gini index, and Lift and Gains charts.
- Get an introduction and practice in using advanced data-science tools (R programming language and RStudio as its main tool) for practical classification problems using the following specific methods:
    - Logistic regression (`glm`), including variable selection using `stepAIC`;
    - CART, which stands for “classification and regression tree,” using `ctree` and `rpart` methods, including discussions and hands-on illustrations of overfitting;
    - CART-like extensions, in particular: `randomforest` and `xgboost`; and
    - Neural networks.

  R code for each of these methods is provided with this note’s supplementary files and can be distributed to students at the discretion of individual instructors. Python code is also available upon request.
- Interpret the outputs from each of these methods and understand their relative strengths and weaknesses.

Assignment Questions

The cases contain a rather unambiguous goal of building a model for predicting retention, and asking students specific questions may diminish the quest for taking the necessary steps. The student code, however, is effectively a step-by-step guide for how to do so—a roadmap/template, as is discussed in the Analysis and the Pedagogy sections that follow.

Analysis

I will start with explaining logistic regression; the supplementary file STC (A) Logistic.R (presented also in **Exhibit TN1**) contains the corresponding code. I will then explain the B case using logistic regression (see **Exhibit TN2** for code and STC (B) Logistic.R file). Finally, I will mention the CART methods (**Exhibit TN3**) and their “derivatives,” random forest and gradient boosting machines (**Exhibit TN4** and **TN5**), and neural networks (**Exhibit TN6**).

The analyses follow 12 main steps (not surprisingly, this is also the structure of the code). Per R, the text after the hashtag (#) is a commentary and is highlighted in green:

1. Make sure that the necessary packages and libraries are loaded. This is done using the “*pac*kage *man*ager” package, humorously called “pacman.”

```
if(“pacman” %in% rownames(installed.packages()) == FALSE)
{install.packages(“pacman”)} # Check if you have universal
installer package, install if not
```

```
pacman::p_load("caret","ROCR","lift","glmnet","MASS","e1071")    # Check, and if needed install the necessary packages
```

2. Load the data from the corresponding CSV data file, examine its structure, and fix the data types incorrectly identified[2] by R when importing from CSV.[3]

```
STCdata_A<-read.csv(file.choose()) # Load the data file to R

str(STCdata_A) # See if some data types were misclassified when importing data from CSV

# Fixing incorrectly classified data types:
STCdata_A$From.Grade <- as.factor(STCdata_A$From.Grade)
…
```

3. Data preprocessing: fix the missing values and combine the rate categories. For both, I provide the custom functions (see **Exhibit TN1**), `fixNAs(dataframe)` and `combinerarecategories(dataframe, mincount)`.

   In short, to fix missing values, the function defines reactions (e.g., adding a new category "FIXED_NA" for a missing value of a categorical/factor variable), and then loops through all columns in the `dataframe`, reads their types (fixed, if necessary at step 2 above), and loops through all the values, applying the defined reaction to any missing data point. In addition, the function creates a surrogate dummy variable for each column containing at least one missing value (e.g., "Special.Pay_surrogate"), which takes a value of 1 whenever the original variable ("Special Pay") has a missing value, and 0 otherwise.

   To combine rare categories, the function again loops through all the columns in the `dataframe`, reads their types, and creates a table of counts for each level of the factor/categorical variables. All levels with counts less than the `mincount` are combined into "other."

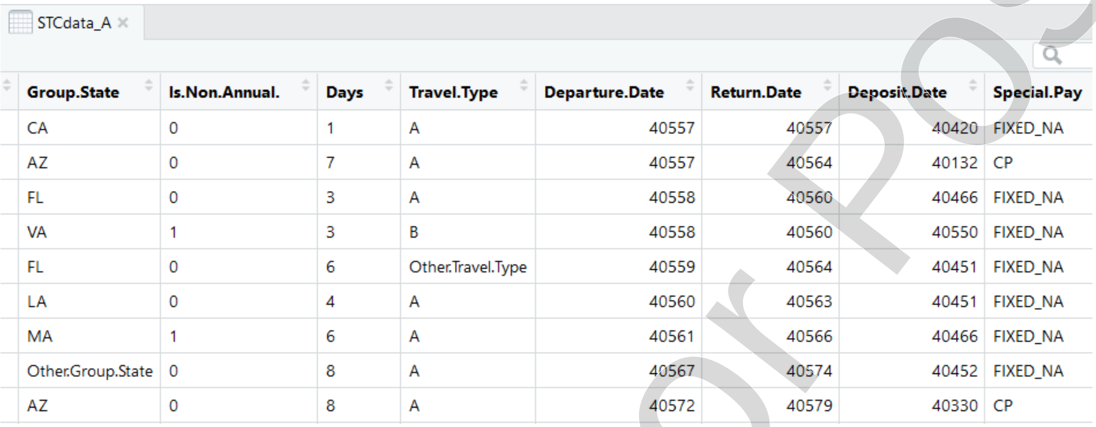
   Calling these functions for the dataframe `STCdata_A` defined in step 2 finalizes cleaning the data; "10" in the function argument below is the minimum count I used for the reasons discussed in step 4 below:

```
STCdata_A<-fixNAs(STCdata_A)

STCdata_A<-combinerarecategories(STCdata_A,10).
```

   Below is a screenshot of a data sample illustrating the result of the two functions:

[2] A note on regional settings: some of the data fields in the case are dates, and, especially when teaching this case to an international audience, one has to be mindful that the date formats may be different. For instance 05/07 in the United States is May 7, while in Germany it is July 5. For this reason the dates in the CSV data are coded as integers using the corrected Excel convention ("2" being January 1, 1990). On some student machines, these dates may not show correctly if opened in Excel; they may not be ready by R correctly after saving from Excel, and depending on the specific regional settings, certain R libraries on student machines may not be able to properly work with dates. For this reason the "base" code provided comments the dates out (i.e., treats them as integers), but the instructors and students are encouraged to experiment with including the dates data in their analyses.

[3] The case data come in an Excel file with three tabs: the title page, the data itself, and the data dictionary. Just save the data tab as a .csv file (comma separated values) and import/read that into R.



Group.State	Is.Non.Annual.	Days	Travel.Type	Departure.Date	Return.Date	Deposit.Date	Special.Pay
CA	0	1	A	40557	40557	40420	FIXED_NA
AZ	0	7	A	40557	40564	40132	CP
FL	0	3	A	40558	40560	40466	FIXED_NA
VA	1	3	B	40558	40560	40550	FIXED_NA
FL	0	6	Other.Travel.Type	40559	40564	40451	FIXED_NA
LA	0	4	A	40560	40563	40451	FIXED_NA
MA	1	6	A	40561	40566	40466	FIXED_NA
Other.Group.State	0	8	A	40567	40574	40452	FIXED_NA
AZ	0	8	A	40572	40579	40330	CP

4. Split the data into testing and training. This is a very important step, both conceptually and technically. Conceptually, because the goal of predictive modeling is not to build a model that fits well into the data it trains on, but rather one that would best predict the new data—a testing dataset is in this sense the best representation of what the "new data" may look like. Technically, to facilitate comparison between the A and B cases, as well as between different models, we need to maintain the same IDs in the corresponding sets at all times and on all student machines. The suggested code accomplishes this through two "tricks": a random seed ensures that the random-number generator is initialized identically in each run; and the inTrain vector is created once and can then be applied anytime the data needs to be split. By default, the code sets 500 data points in the testing dataset, and the remainder 1,889 into the training dataset.

```
set.seed(77850) # set a random number generation seed to ensure
that the split is the same every time
inTrain <- createDataPartition(y = STCdata_A$Retained.in.2012.,
                               p = 1888/2389, list = FALSE)
training <- STCdata_A[ inTrain,]
testing <- STCdata_A[ -inTrain,]
```

5. Run the "base-case" model. One may be tempted to simply include all the variables, for example, by using `glm(Retained.in.2012.~ ., data=training, family=“binomial” (link = “logit”))`, but in my experience this would make the resultant model too large for the variable selection that follows, and running it "live" in class would just take too much time. So instead, I manually preselected some reasonable variables (based on my experience with the case), making the base-case model as follows:

```
glm(formula = Retained.in.2012. ~ Special.Pay + To.Grade +
Group.State + Is.Non.Annual. + Tuition + FRP.Active +
FRP.Cancelled + FRP.Take.up.percent. + Cancelled.Pax +
Total.Discount.Pax + Initial.System.Date + Poverty.Code +
CRM.Segment + School.Type + Parent.Meeting.Flag +
MDR.Low.Grade + MDR.High.Grade + Total.School.Enrollment +
EZ.Pay.Take.Up.Rate + School.Sponsor + SPR.New.Existing + FPP +
FirstMeeting + LastMeeting + DifferenceTraveltoFirstMeeting +
DepartureMonth + MajorProgramCode + SingleGradeTripFlag +
FPP.to.School.enrollment + FPP.to.PAX + SchoolSizeIndicator,
family = binomial(link = "logit"), data = training)
```

Note that the training dataset is used to fit the model. **Table TN1a** presents the results of the estimation. The model is overfit, meaning it has too many insignificant variables, which naturally leads to the next step.

6. Variable selection: I use the stepAIC method, with "both" for direction and "on" for trace Note that this step will take several minutes, but as it progresses it could be insightful to point out to the students the process of adding and moving variables (labeled by "+" and "−" in the output), gradual shrinking of the model, and reduction in (improvement of) AIC:

```
model_logistic_stepwiseAIC<-stepAIC(model_logistic,direction =
c("both"),trace = 1) # AIC stepwise
summary(model_logistic_stepwiseAIC)
```

The resultant model is the following:

```
glm(formula = Retained.in.2012. ~ Special.Pay + Is.Non.Annual. +
FRP.Active + CRM.Segment + School.Type + MDR.High.Grade +
SPR.New.Existing + DepartureMonth + MajorProgramCode +
SingleGradeTripFlag + SchoolSizeIndicator,
family = binomial(link = "logit"), data = training)
```

Table TN1b presents the results of the estimation: contrast, visually, how much smaller the model is and, proportionally, how many more variables are significant:

Table TN1a

```
Coefficients:                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -3.514e+00  3.404e+00  -1.032 0.301947
Special.PayFIXED_NA            -7.457e-01  5.550e-01  -1.344 0.179078
Special.PayFR                  -7.522e-01  5.788e-01  -1.300 0.193750
Special.PaySA                   2.471e-01  6.724e-01   0.368 0.713235
To.Grade11                     -1.110e+00  1.202e+00  -0.924 0.355508
To.Grade12                     -1.790e-01  8.407e-01  -0.213 0.831356
To.GradeOther.To.Grade          1.040e+01  3.247e+02   0.032 0.974444
To.Grade4                      -2.475e-01  9.612e-01  -0.258 0.796771
To.Grade5                       6.363e-01  9.454e-01   0.673 0.500963
To.Grade6                       5.668e-01  9.061e-01   0.626 0.531578
To.Grade7                       8.442e-01  9.018e-01   0.936 0.349250
To.Grade8                       2.918e-01  7.968e-01   0.366 0.714198
To.Grade9                       5.755e-01  8.522e-01   0.675 0.499498
To.GradeFIXED_NA                7.531e-01  8.319e-01   0.905 0.365314
Group.StateAL                   6.146e-01  8.223e-01   0.747 0.454795
Group.StateAR                  -1.260e-01  1.001e+00  -0.126 0.899772
Group.StateAZ                   3.175e-01  5.806e-01   0.547 0.584500
Group.StateCA                   5.453e-01  3.987e-01   1.368 0.171436
Group.StateCO                   4.142e-01  4.991e-01   0.830 0.406582
Group.StateCT                   5.391e-01  1.154e+00   0.467 0.640409
Group.StateFL                   5.397e-01  5.865e-01   0.920 0.357463
Group.StateGA                   5.063e-01  7.156e-01   0.707 0.479264
Group.StateIA                   6.836e-03  7.387e-01   0.009 0.992616
Group.StateID                  -1.557e+00  1.226e+00  -1.270 0.203976
Group.StateIL                   7.711e-01  5.093e-01   1.514 0.130023
Group.StateIN                  -4.345e-01  6.013e-01  -0.723 0.469918
Group.StateKS                   1.087e-01  7.093e-01   0.153 0.878235
Group.StateKY                  -1.826e-01  8.670e-01  -0.211 0.833170
Group.StateLA                   3.520e-01  6.460e-01   0.545 0.585890
Group.StateMA                   4.291e-01  7.416e-01   0.579 0.562864
Group.StateMD                  -1.986e-01  8.655e-01  -0.229 0.818523
Group.StateMI                   7.627e-01  6.437e-01   1.185 0.236075
Group.StateMN                   5.116e-01  6.483e-01   0.789 0.430068
Group.StateMO                   1.270e+00  6.422e-01   1.978 0.047890 *
Group.StateNC                  -9.546e-01  8.603e-01  -1.110 0.267163
Group.StateNE                   8.039e-01  6.677e-01   1.204 0.228581
Group.StateNM                   7.651e-01  7.771e-01   0.985 0.324848
Group.StateNV                   1.812e+00  8.558e-01   2.117 0.034235 *
Group.StateNY                   4.069e-01  7.932e-01   0.513 0.608018
Group.StateOH                  -4.166e-01  5.988e-01  -0.696 0.486629
Group.StateOK                   2.214e-01  7.351e-01   0.301 0.763223
Group.StateOR                   4.832e-01  5.897e-01   0.819 0.412565
Group.StateSC                  -1.726e+00  1.024e+00  -1.686 0.091888 .
Group.StateSD                   1.831e+00  9.421e-01   1.944 0.051925 .
Group.StateTN                   5.110e-01  6.762e-01   0.756 0.449813
Group.StateTX                   7.396e-01  4.091e-01   1.808 0.070625 .
Group.StateVA                   4.844e-01  8.858e-01   0.547 0.584434
Group.StateWA                   1.483e-01  4.605e-01   0.322 0.747500
Group.StateWI                   8.661e-01  6.640e-01   1.304 0.192069
Is.Non.Annual.1                -2.793e+00  2.309e-01 -12.094  < 2e-16 ***
Tuition                        -2.778e-04  2.069e-04  -1.343 0.179430
FRP.Active                      4.404e-02  1.334e-02   3.301 0.000962 ***
```

Table TN1b

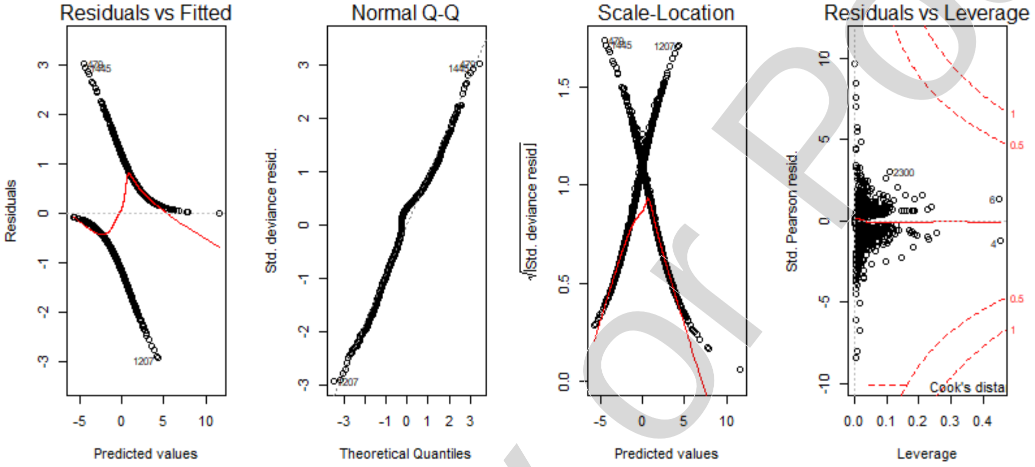
```
Coefficients:                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                            0.966757   1.357815   0.712  0.47647
Special.PayFIXED_NA                   -0.727395   0.527963  -1.378  0.16828
Special.PayFR                         -0.692510   0.553081  -1.252  0.21054
Special.PaySA                          0.289181   0.615820   0.470  0.63865
Is.Non.Annual.1                       -2.724595   0.213751 -12.747  < 2e-16 ***
FRP.Active                             0.041079   0.006239   6.584 4.59e-11 ***
CRM.Segment10                          0.706590   0.381933   1.850  0.06431 .
CRM.Segment11                          1.571575   0.953249   1.649  0.09922 .
CRM.Segment2                          -0.098996   0.566779  -0.175  0.86134
CRM.Segment3                           0.873555   1.029371   0.849  0.39609
CRM.Segment4                           2.322681   0.758219   3.063  0.00219 **
CRM.Segment5                           0.649724   0.388786   1.671  0.09469 .
CRM.Segment6                           1.743522   0.728542   2.393  0.01670 *
CRM.Segment7                           0.488423   0.737468   0.662  0.50778
CRM.Segment8                           0.168147   0.590332   0.285  0.77577
CRM.SegmentOther.CRM.Segment          -0.793365   0.911851  -0.870  0.38427
School.TypeCHD                        -0.340545   0.363523  -0.937  0.34887
School.TypePrivate non-Christian       0.618729   0.403725   1.533  0.12539
School.TypePUBLIC                     -0.361060   0.285121  -1.266  0.20539
MDR.High.Grade12                      -0.941716   0.952547  -0.989  0.32284
MDR.High.Grade5                       -1.911913   1.105530  -1.729  0.08374 .
MDR.High.Grade6                       -2.373763   1.078090  -2.202  0.02768 *
MDR.High.Grade7                       -0.180702   1.134323  -0.159  0.87343
MDR.High.Grade8                        0.017753   0.939492   0.019  0.98492
MDR.High.Grade9                        0.045237   1.028373   0.044  0.96491
MDR.High.GradeFIXED_NA                 1.217031   1.251268   0.973  0.33073
SPR.New.ExistingNEW                   -1.347067   0.138204  -9.747  < 2e-16 ***
DepartureMonthFebruary                 1.973805   0.843932   2.339  0.01934 *
DepartureMonthOther.DepartureMonth     1.876605   1.352372   1.388  0.16525
DepartureMonthJune                     0.058567   0.171797   0.341  0.73317
DepartureMonthMarch                    0.541059   0.211936   2.553  0.01068 *
DepartureMonthMay                      0.524614   0.203196   2.582  0.00983 **
MajorProgramCodeH                     -0.912232   0.365160  -2.498  0.01248 *
MajorProgramCodeI                     -0.898368   1.189758  -0.755  0.45020
MajorProgramCodeS                     -1.739228   0.659565  -2.637  0.00837 **
SingleGradeTripFlag1                   0.966808   0.135396   7.141 9.29e-13 ***
SchoolSizeIndicatorL                   0.891637   0.584198   1.526  0.12695
SchoolSizeIndicatorM-L                 0.352310   0.578643   0.609  0.54262
SchoolSizeIndicatorS                  -0.177974   0.578815  -0.307  0.75848
SchoolSizeIndicatorS-M                 0.446037   0.579697   0.769  0.44164
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2531.2  on 1888  degrees of freedom
Residual deviance: 1598.1  on 1848  degrees of freedom
AIC: 1680.1

Number of Fisher Scoring iterations: 13
```

```
FRP.Cancelled                     -4.374e-02  3.795e-02  -1.153 0.249069
FRP.Take.up.percent.              -4.933e-02  4.370e-01  -0.113 0.910124
Cancelled.Pax                      1.832e-02  2.947e-02   0.622 0.534160
Total.Discount.Pax                 1.106e-01  7.415e-02   1.492 0.135690
Initial.System.Date                1.306e-05  3.767e-05   0.347 0.728898
Poverty.CodeA                      1.398e+00  1.467e+00   0.952 0.340849
Poverty.CodeB                      1.198e+00  1.454e+00   0.824 0.409959
Poverty.CodeC                      1.071e+00  1.452e+00   0.737 0.460832
Poverty.CodeD                      1.694e+00  1.539e+00   1.101 0.270945
Poverty.CodeE                      1.367e+00  1.663e+00   0.822 0.410916
Poverty.CodeFIXED_NA               1.851e+00  1.340e+00   1.381 0.167178
CRM.Segment10                      7.465e-01  4.161e-01   1.794 0.072812 .
CRM.Segment11                      1.247e+00  9.722e-01   1.282 0.199726
CRM.Segment2                       1.540e-01  6.401e-01   0.241 0.809872
CRM.Segment3                       7.152e-01  1.131e+00   0.632 0.527319
CRM.Segment4                       2.759e+00  8.182e-01   3.372 0.000745 ***
CRM.Segment5                       8.364e-01  4.451e-01   1.879 0.060207 .
CRM.Segment6                       1.954e+00  8.407e-01   2.324 0.020126 *
CRM.Segment7                       5.103e-01  8.062e-01   0.633 0.526774
CRM.Segment8                       5.356e-01  8.570e-01   0.625 0.532007
CRM.SegmentOther.CRM.Segment      -7.277e-01  9.961e-01  -0.731 0.465050
School.TypeCHD                    -3.988e-01  4.202e-01  -0.949 0.342496
School.TypePrivate non-Christian   4.890e-01  4.453e-01   1.098 0.272187
School.TypePUBLIC                 -1.060e-01  6.225e-01  -0.170 0.864787
Parent.Meeting.Flag1               2.627e+01  2.090e+02   0.126 0.899949
MDR.Low.Grade3                     1.150e+00  1.321e+00   0.870 0.384060
MDR.Low.Grade4                     6.825e-01  1.219e+00   0.560 0.575710
MDR.Low.Grade5                     1.331e+00  1.039e+00   1.281 0.200151
MDR.Low.Grade6                     9.280e-01  9.780e-01   0.949 0.342695
MDR.Low.Grade7                     9.498e-01  9.808e-01   0.968 0.332877
MDR.Low.Grade8                     4.530e-02  1.271e+00   0.036 0.971579
MDR.Low.Grade9                     7.413e-01  1.094e+00   0.678 0.497966
MDR.Low.GradeFIXED_NA              1.692e+00  1.737e+00   0.974 0.329910
MDR.Low.GradeK                     6.720e-01  9.803e-01   0.686 0.493021
MDR.Low.GradePK                    3.932e-01  9.878e-01   0.398 0.690571
MDR.High.Grade12                  -9.973e-01  1.154e+00  -0.864 0.387515
MDR.High.Grade5                   -2.372e+00  1.308e+00  -1.814 0.069707 .
MDR.High.Grade6                   -2.877e+00  1.271e+00  -2.263 0.023628 *
MDR.High.Grade7                   -6.916e-01  1.348e+00  -0.513 0.607943
MDR.High.Grade8                   -2.814e-01  1.138e+00  -0.247 0.804746
MDR.High.Grade9                    7.766e-02  1.239e+00   0.063 0.950016
MDR.High.GradeFIXED_NA                    NA         NA      NA       NA
Total.School.Enrollment            3.080e-04  3.392e-04   0.908 0.363825
EZ.Pay.Take.Up.Rate               -4.654e-02  4.435e-01  -0.105 0.916415
School.Sponsor1                    4.143e-01  3.286e-01   1.261 0.207352
SPR.New.ExistingNEW               -1.403e+00  1.567e-01  -8.955  < 2e-16 ***
FPP                               -1.200e-02  8.789e-03  -1.365 0.172292
FirstMeeting                       4.747e-04  5.072e-03   0.094 0.925442
LastMeeting                       -1.117e-03  1.914e-03  -0.584 0.559470
DifferenceTraveltoFirstMeeting    -1.951e-04  4.921e-03  -0.040 0.968377
DepartureMonthFebruary             1.940e+00  9.440e-01   2.055 0.039918 *
DepartureMonthOther.DepartureMonth 2.589e+00  1.523e+00   1.700 0.089173 .
DepartureMonthJune                 4.264e-02  3.089e-01   0.138 0.890200
DepartureMonthMarch                5.135e-01  2.588e-01   1.984 0.047240 *
DepartureMonthMay                  4.747e-01  2.703e-01   1.756 0.079090 .
MajorProgramCodeH                 -8.467e-01  4.186e-01  -2.023 0.043103 *
MajorProgramCodeI                 -1.465e+00  1.354e+00  -1.082 0.279358
MajorProgramCodeS                 -1.595e+00  7.295e-01  -2.187 0.028755 *
SingleGradeTripFlag1               1.029e+00  1.610e-01   6.395 1.61e-10 ***
FPP.to.School.enrollment          -1.787e-01  1.140e+00  -0.157 0.875511
FPP.to.PAX                         1.241e+00  1.835e+00   0.676 0.498967
SchoolSizeIndicatorL               7.005e-01  7.237e-01   0.968 0.333030
SchoolSizeIndicatorM-L             2.948e-01  6.620e-01   0.445 0.656085
SchoolSizeIndicatorS              -2.670e-02  6.374e-01  -0.042 0.966585
SchoolSizeIndicatorS-M             4.942e-01  6.416e-01   0.770 0.441133 ---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

7. Check the diagnostic plots. Note that for the four plots to appear side-by-side, we first partition the output window into a table with one row and four columns, then plot and partition them back into a single window.

```
par(mfrow=c(1,4))
plot(model_logistic_stepwiseAIC) # Error plots: similar in nature
to lm plots.
par(mfrow=c(1,1))
```

The resultant plots are below. Their interpretation is like the error plots for linear regression. In fact, the Q-Q plot and the Residuals vs. Leverage are identical. On the Q-Q, we expect to see that the standardized residuals form a 45-degree line (which they closely resemble); this shows that there are no major outliers or systematic deviations. On the Residuals vs. Leverage plot, all the data points are within the bell-shaped curves; this shows that there are no overly influential data points. All of these are sufficient to conclude that we have a “good” model that can be used for predictions.



8. Predict probabilities (a vector in the case of logistic regression) and classification. Note that the `testing` dataset is used (i.e., the data that the model has not seen yet).

```
logistic_probabilities<-predict(model_logistic_stepwiseAIC,
newdata=testing,type=“response”) # Predict probabilities
logistic_classification<-rep(“1”,500)
logistic_classification[logistic_probabilities<0.6073]=“0”
# Predict classification using 0.6073 threshold. Why 0.6073?
That’s the average probability of being retained in the data. An
alternative code: logistic_classification <-
as.integer(logistic_probabilities >
mean(testing$Retained.in.2012. == “1”))
```

This step requires a major decision in the analyses of classification: the cutoff or “threshold,” which is the probability of being retained above which we will label a customer as being classified as “retained.” Many packages will use the default 50% cutoff—this is driven by the underlying principle that the classification data must be balanced (i.e., contain exactly half of the data points belonging to Class 0, and half belonging to Class 1). In our case, the data is slightly unbalanced: 1,451 data points (60.73%) are in Class 1 and 938 are in Class 0. For very unbalanced data, I recommend first balancing it (over- or under-sample), but in this case the benefits of balancing are unclear, hence one can implement the average probability of being retained as a cutoff. I emphasize, though, that this is ultimately an analyst’s decision.

9. Obtain confusion matrix and its statistics:

```
logistic_classification<-as.factor(logistic_classification)
confusionMatrix(logistic_classification,testing$Retained.in.2012.
,positive = “1”)
```

On the technical side, `positive = “1”` above is important, as by default *R* assigns the level found in the first row of a categorical/factor variable as default (“0”). In our case, however, the customer in the first row was retained, and so one must redefine the meaning of the “positive” (otherwise the calculations of the metrics in the confusion matrix below will be reversed).

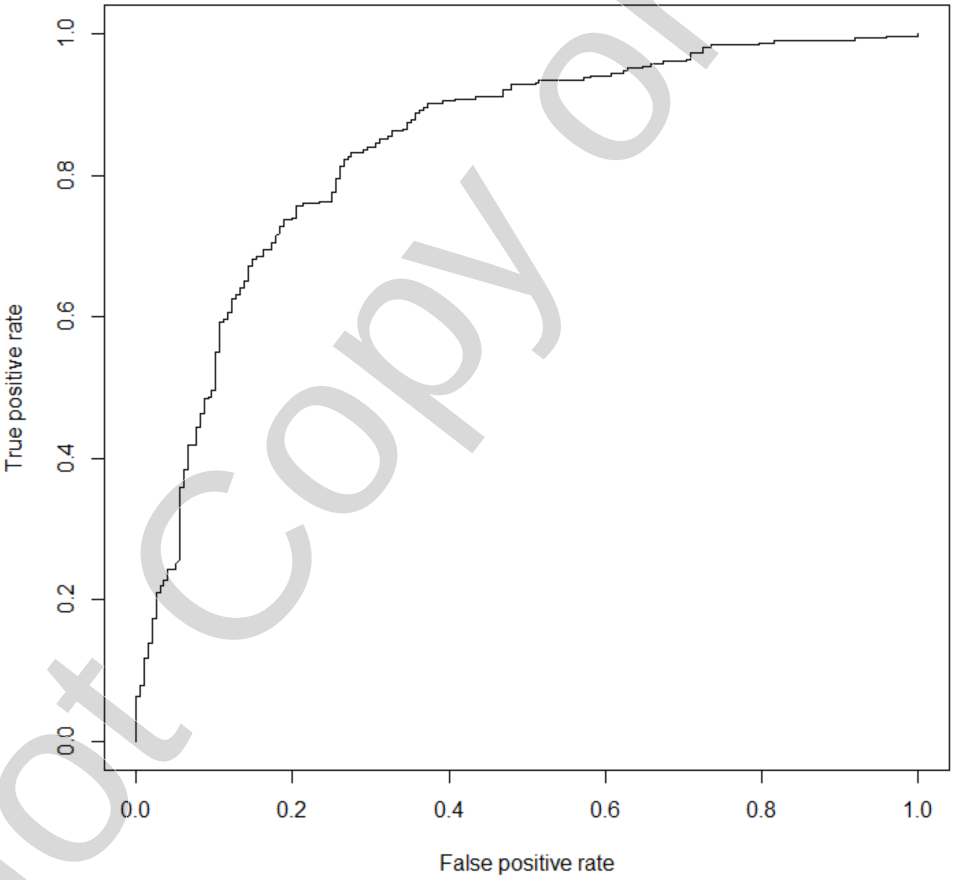
We next discuss the resulting output, shown below. The confusion matrix is at the very top—to read it, proceed as follows. The 500 testing data points contained 151 + 45 = 196 customers who were not retained (i.e., actually belong to Class 0, which we above defined as "Negative"). The model correctly predicted that 151 of them would not be retained (True Negative), and made 45 mistakes (False Positive). The resultant Specificity, or True Negative rate, is 151 / 196 = 0.7704 Similarly for positives, the model made 233 correct predictions out of (233 + 71), a 76.64% Sensitivity. Overall, the model made 45 + 71 = 116 mistakes out of 500, an accuracy of 76.8%, and so on.

```
            Reference
  Prediction   0   1
           0 151  71
           1  45 233

                Accuracy : 0.768
                  95% CI : (0.7285, 0.8043)
     No Information Rate : 0.608
     P-Value [Acc > NIR] : 2.293e-14

                   Kappa : 0.5245
  Mcnemar’s Test P-Value : 0.02028

             Sensitivity : 0.7664
             Specificity : 0.7704
          Pos Pred Value : 0.8381
          Neg Pred Value : 0.6802
              Prevalence : 0.6080
          Detection Rate : 0.4660
    Detection Prevalence : 0.5560
       Balanced Accuracy : 0.7684

        ‘Positive’ Class : 1
```

At this point, I make a remark that our case model has balanced accuracy, meaning that it makes positive and negative mistakes with an approximately equal rate (sensitivity is approximately equal to specificity). This may or may not be desirable, depending on the application, and can be easily manipulated by setting the classification cutoff/threshold differently. This naturally brings the discussion to the next step.

10. Obtain the ROC ("Receiver Operating Characteristic") curve. Originally developed after the Pearl Harbor attack to classify, the ROC curve is perhaps the most useful summary of the model performance. The way I explain it to students is as follows. We know that our model is making about 23.2% errors, but do we make those errors uniformly for all customers? That would be unfortunate as the model will not be that decisive. If we take a consumer with a 99% predicted probability, one with a 50% probability, and one with a 1% probability, we would hope that the model will be rather certain that the first customer will be retained, rather certain that the third will not, but may be comfortable with a large error for the second.

    The ROC curve depicts just that; it ranks all customers from the highest to the lowest predicted probability of retention (equivalently, varies the cutoff/threshold from high to low). Starting from coordinate (0,0)—bottom left—on the graph, it maps all customers in decreasing order of probability (i.e., from "best" to "worst"). A correct prediction moves the curve up, while a mistake moves the curve left. A perfect, 100% accurate classifier would first correctly predict all positives and would then correctly predict all negatives; that is, the curve would go straight up until (0,1), and then turn and be horizontal (flat) until (1,1). This is, of course, not possible in practice, and the "steps" in the curve reflect the occasional mistakes the model makes. As mentioned earlier, a good model will make few positive mistakes for the best customers and few negative mistakes for the worst—the ROC curve will thus have a nearly vertical section close to (0,0), and a nearly horizontal section close to (1,1). We observe something similar in our case too:



It is worth noting that a model that is just guessing will clearly have an ROC curve at a 45-degree line—such a model would be equally likely to make a correct or incorrect prediction regardless of whether the customer has a high or low predicted probability.

11. A measure that summarizes the ROC curve in one number is Area Under the Curve, or AUC.

```
AUC.tmp <- performance(logistic_ROC_prediction,"auc") # Create AUC
data
logistic_AUC <- as.numeric(AUC.tmp@y.values) # Calculate AUC
logistic_AUC # Display AUC value
```

AUC of an error-free classifier would be 100%, and an AUC of a random guess would be 50%. In-between: AUC of 90%+ = excellent, 80–90% = very good, 70–80% = good, 60–70% = so-so, and below 60% = not much value.

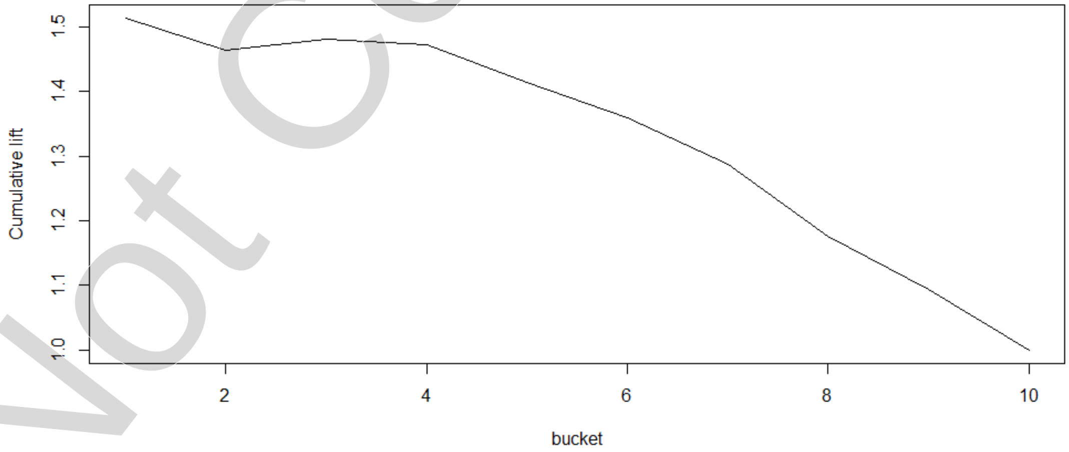
In our case, AUC = 0.8419—not a bad result at all!

The concept of AUC is closely connected to the Gini index used in economics. Gini index = 2 × AUC − 1.

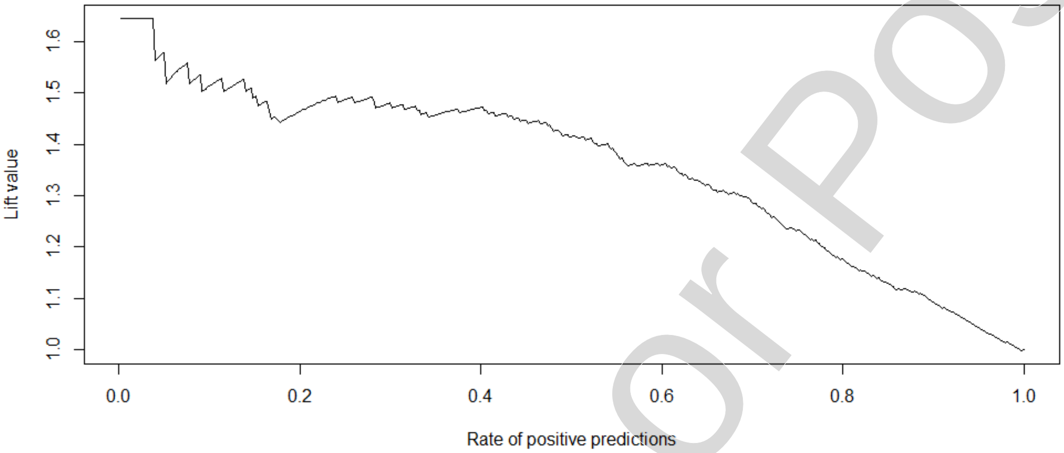
12. The last metric I present is the lift chart, a measure of cumulative performance of the model. There are two ways to plot such curves: one based on buckets (e.g., based on deciles as is coded with 10 buckets below), and another based on every single data point:

```
plotLift(logistic_probabilities,      testing$Retained.in.2012.,
cumulative = TRUE, n.buckets = 10) # Plot Lift chart

### An alternative way is to plot a lift curve not by buckets, but
on all data points
Lift <- performance(logistic_ROC_prediction,“lift”,“rpp”)
plot(lift)
```

In either case, the logic is the same. If the model was just guessing, in the top 10% of customers, it would correctly predict 50 × 60.73% ≈ 31 retained customers from the 500 customers in the training data. From the ROC curve, however, our model makes what looks like 4 mistakes (2 mistakes twice) in the top 10% (i.e., correctly predicts 46 retained customers). The lift is 46 / 31 ≈ 1.5. Extending this logic to the top 20%, top 30%, and so on plots the curve below:



And a similar, but somewhat less smooth, picture is observed if we apply this logic to each data point instead of aggregating by deciles:



Logistic Regression Analyses for the B case

To conclude the analysis using the logistic regression model, repeat the steps for the B case. The code is provided in the file STC (B) Logistic.R and is pasted in **Exhibit TN2**. Three nuances are worth noting:

- First, to ensure true apples-to-apples comparison, the B case code needs to be run after the A case in the same R session.
- Second, the B case starts with reading the additional B case data, merging them with the A case data, and then resplitting them into training and testing using the same vector of `inTrain` IDs (see step 4 above).
- Third, the new NPS variables need to be added to the logistic model prior to the stepwise variable selection.

After that, and noting the change of "_A" to "_B" throughout, the code is identical.

The resultant model includes three significant NPS variables. The confusion matrix also improves (higher accuracy) and does so in a balanced way (both sensitivity and specificity improve), the ROC curve improves (the plot below shows an overlay), and the AUC grows by nearly 2%. All in all, we see an improved performance from incorporating the NPS data, although perhaps a smaller improvement than the students might expect to see.

Model:

```
glm(formula = Retained.in.2012. ~ Special.Pay + Is.Non.Annual. +
    FRP.Active + CRM.Segment + School.Type + MDR.High.Grade +
    School.Sponsor + SPR.New.Existing + DepartureMonth +
    MajorProgramCode + SingleGradeTripFlag + SchoolSizeIndicator +
    NPS.2011 + NPS.2010 + NPS.2008, family = binomial(link =
    “logit”), data = training)
```

Resultant coefficients: three NPS scores are significant as highlighted below:

```
Coefficients:
```

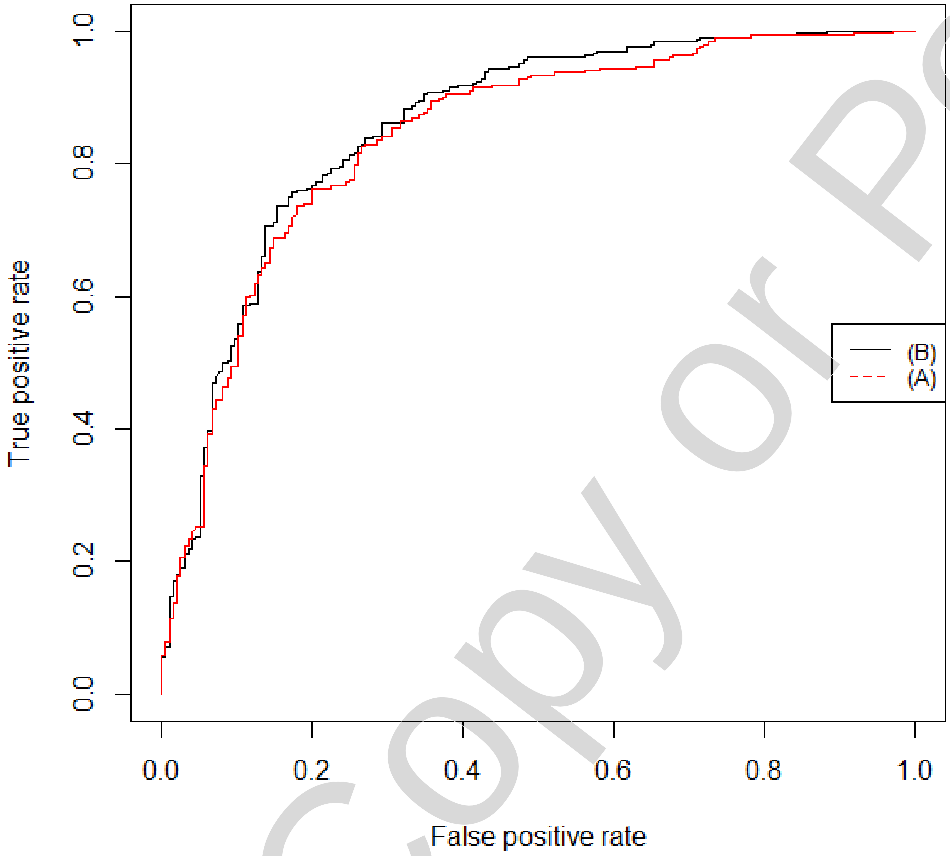
```
                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                         0.517383   1.361001   0.380  0.70384
Special.PayFIXED_NA                -0.807357   0.557028  -1.449  0.14723
Special.PayFR                      -0.870738   0.582825  -1.494  0.13518
Special.PaySA                       0.159446   0.649480   0.245  0.80607
Is.Non.Annual.1                    -2.370383   0.223787 -10.592  < 2e-16 ***
FRP.Active                          0.037660   0.006263   6.013 1.82e-09 ***
CRM.Segment10                       0.582678   0.385322   1.512  0.13049
CRM.Segment11                       1.472261   0.962632   1.529  0.12616
CRM.Segment2                       -0.238733   0.575165  -0.415  0.67809
CRM.Segment3                        0.605961   1.027346   0.590  0.55530
CRM.Segment4                        2.121981   0.760944   2.789  0.00529 **
CRM.Segment5                        0.520881   0.393415   1.324  0.18550
CRM.Segment6                        1.639815   0.735371   2.230  0.02575 *
CRM.Segment7                        0.402254   0.742670   0.542  0.58807
CRM.Segment8                        0.002969   0.596834   0.005  0.99603
CRM.SegmentOther.CRM.Segment       -0.983540   0.929862  -1.058  0.29018
School.TypeCHD                     -0.536352   0.382791  -1.401  0.16117
School.TypePrivate non-Christian    0.602438   0.413621   1.456  0.14526
School.TypePUBLIC                  -0.371458   0.291527  -1.274  0.20260
MDR.High.Grade12                   -1.263708   0.923611  -1.368  0.17124
MDR.High.Grade5                    -2.136595   1.060982  -2.014  0.04403 *
MDR.High.Grade6                    -2.592482   1.031663  -2.513  0.01197 *
MDR.High.Grade7                    -0.499979   1.105775  -0.452  0.65116
MDR.High.Grade8                    -0.305323   0.909828  -0.336  0.73718
MDR.High.Grade9                    -0.117308   1.001546  -0.117  0.90676
MDR.High.GradeFIXED_NA              0.884547   1.229802   0.719  0.47198
School.Sponsor1                     0.492818   0.287550   1.714  0.08656 .
SPR.New.ExistingNEW                -0.598676   0.184833  -3.239  0.00120 **
DepartureMonthFebruary              1.888995   0.804716   2.347  0.01890 *
DepartureMonthOther.DepartureMonth  2.081985   1.409653   1.477  0.13969
DepartureMonthJune                  0.134636   0.176701   0.762  0.44609
DepartureMonthMarch                 0.545559   0.216833   2.516  0.01187 *
DepartureMonthMay                   0.583059   0.207700   2.807  0.00500 **
MajorProgramCodeH                  -0.941070   0.373073  -2.522  0.01165 *
MajorProgramCodeI                  -0.791733   1.195595  -0.662  0.50784
MajorProgramCodeS                  -1.714934   0.671396  -2.554  0.01064 *
SingleGradeTripFlag1                0.912475   0.138235   6.601 4.09e-11 ***
SchoolSizeIndicatorL                0.757242   0.601053   1.260  0.20772
SchoolSizeIndicatorM-L              0.193083   0.595757   0.324  0.74586
SchoolSizeIndicatorS               -0.332973   0.596660  -0.558  0.57680
SchoolSizeIndicatorS-M              0.329737   0.597928   0.551  0.58131
NPS.2011                            0.047544   0.015771   3.015  0.00257 **
NPS.2010                            0.092179   0.019310   4.774 1.81e-06 ***
NPS.2008                            0.041136   0.016679   2.466  0.01365 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Confusion Matrix and Statistics: accuracy improves and in a balanced way.

```
           Reference
```

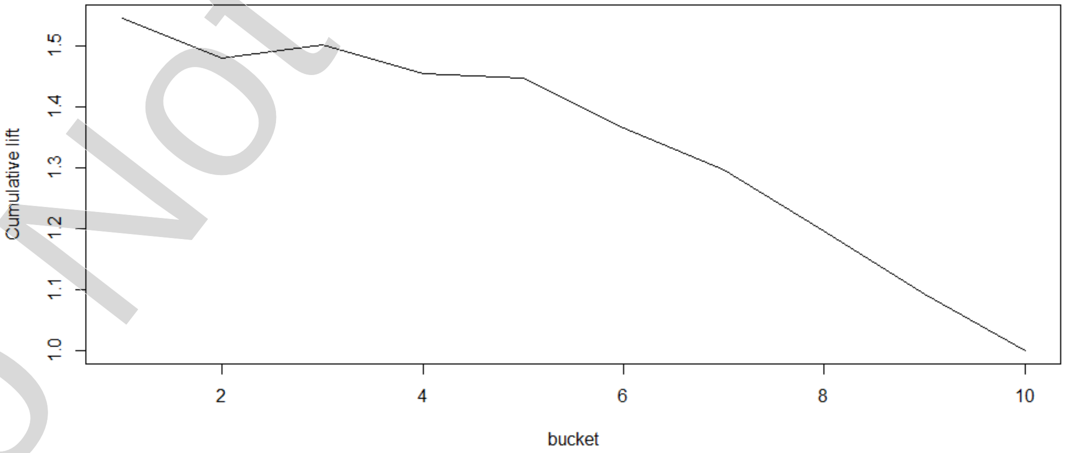
```
Prediction   0   1
         0 156  70
         1  40 234

               Accuracy : 0.78
                 95% CI : (0.7411, 0.8156)
    No Information Rate : 0.608
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5507
 Mcnemar's Test P-Value : 0.005692

            Sensitivity : 0.7697
            Specificity : 0.7959
         Pos Pred Value : 0.8540
         Neg Pred Value : 0.6903
             Prevalence : 0.6080
         Detection Rate : 0.4680
   Detection Prevalence : 0.5480
      Balanced Accuracy : 0.7828

       'Positive' Class : 1
```

ROC curves: For comparison purposes, the B case is in black and the A case is superimposed in red. The B curve is higher nearly everywhere, which implies fewer mistakes and higher accuracy:

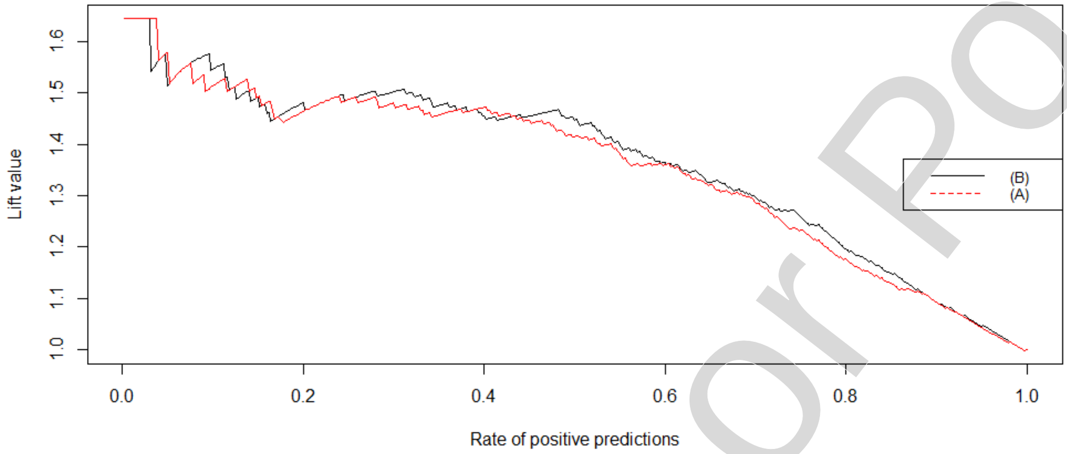


AUC=0.8574—this is nearly 2% larger than in the A case.

Lift charts: First, we plot one by decline:



Second, we plot one for all the data (with the B case in black and the A case superimposed in red). Again, apart from several isolated data points, the model with NPS scores has a higher lift.



In concluding the analyses using logistic regression, why is the NPS data's improvement so small? Two things contribute to this. First, some of the variables in the A case de facto measure satisfaction already. Second, it is perhaps not the NPS per se, but the change in the NPS, or even the presence of the score (whether it was missing or not), that matters more. This calls for some feature engineering, which I leave for the instructors and their students to do.

Concluding the analysis section overall, the case is accompanied with the R scripts for the following techniques, which together cover the vast majority of the classification techniques: CART, random forest, gradient boosting machines, and neural networks.

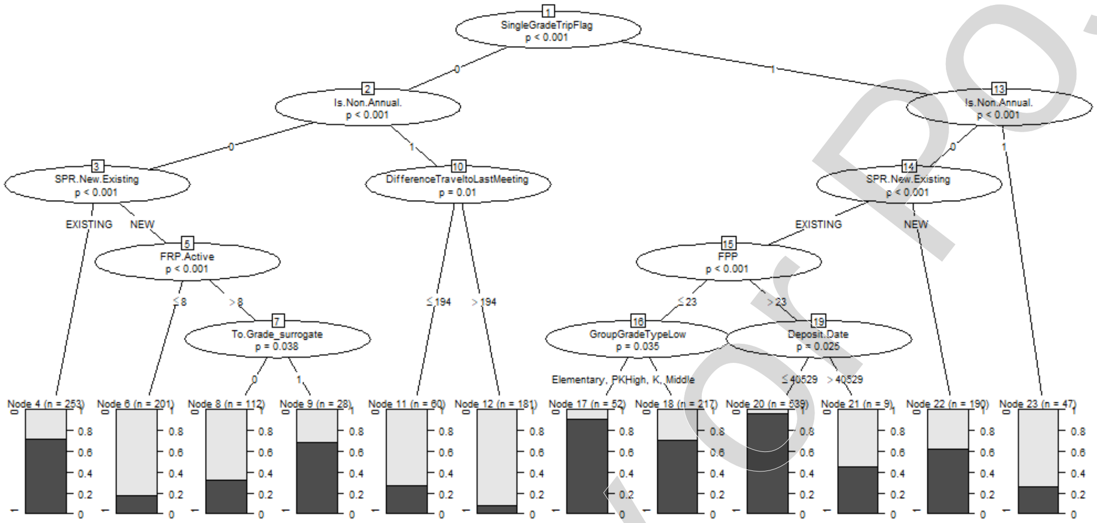
I briefly present the CART analyses below; the code for the other techniques is generally similar and I thus do not provide detailed discussion. The R code is presented in **Exhibit TN3** and in the corresponding supplementary file.

There are two main families of CART methods: conditional inferencing (`ctree`) and recursive partitioning (`rpart`). Both methods decide on binary splits at each node, but the former does so based on statistical theory (significance tests) while the latter is purely based on the maximization of fit (Gini coefficient). Ctree is generally slower, but has a potential to avoid some of the biases in rpart. Either method can effectively deal with missing values, but to ensure apples-to-apples comparison, I am running them after preprocessing data for logistic regression as discussed above.

The R code for `ctree` is:

```
ctree_tree<-ctree(Retained.in.2012.~.,data=training)  #  Run  ctree  on
training data
plot(ctree_tree, gp = gpar(fontsize = 7)) # Plot the tree (adjust font
size if needed)
```

This results in the following tree:



CART helps isolate the pockets of data with similar characteristics. Its clear advantage is the high level of interpretability. For instance, it is hardly a surprise that a customer with a `SingleGradeTripFlag=0` and `Is.Non.Annual=1` is not retained (lowest bucket in the middle): the group that went on a trip last year contained students from two classes (e.g., grades 7 and 8) and the school therefore skips a grade before the new cohort is available. A similar, intuitive logic also applies to the pocket of likely retains—see for yourself.

This tree has an accuracy of 79.6% on the testing data (i.e., better than logistic), but a slightly lower 82.6% AUC than the logistic regression model. The ROC curve will be discussed shortly.

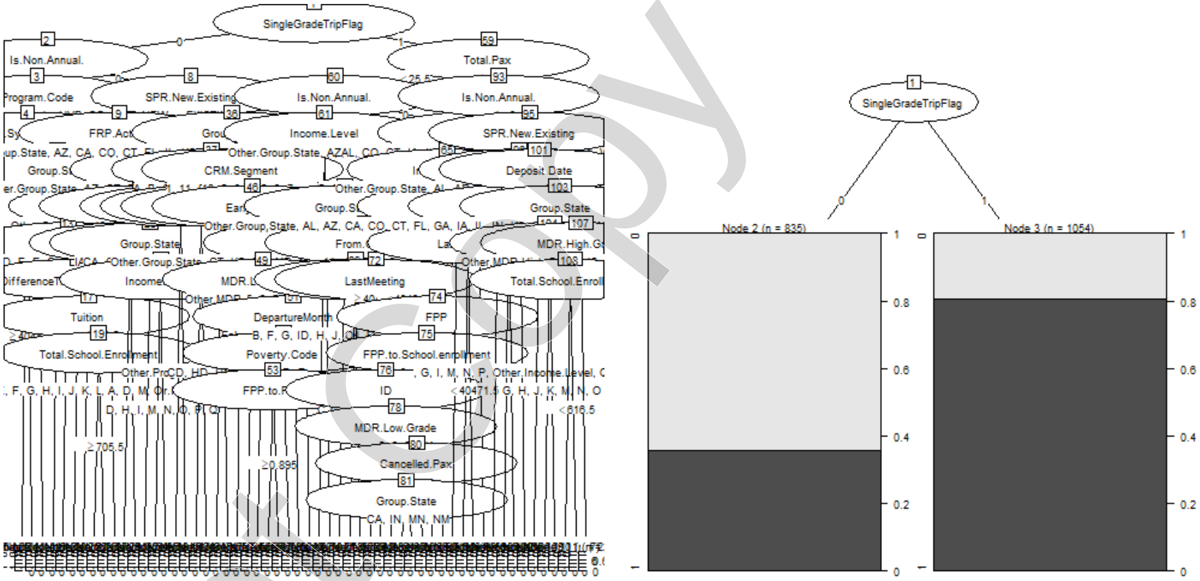
The R code for the `rpart` tree is slightly more complex because it requires specifying a complexity parameter, `cp`; a smaller `cp` will result in a larger tree. There are two ways to build such trees: "growing from small to large," and "pruning from large to small." I prefer the latter, especially if the dataset is not excessively large. That is, set a small `cp` first and build a large tree:

```
rpart_cp = rpart.control(cp = 0.0005)
rpart_tree<-rpart(Retained.in.2012.~.,data=training,   method=“class”,
control=rpart_cp) # Run ctree on training data
```

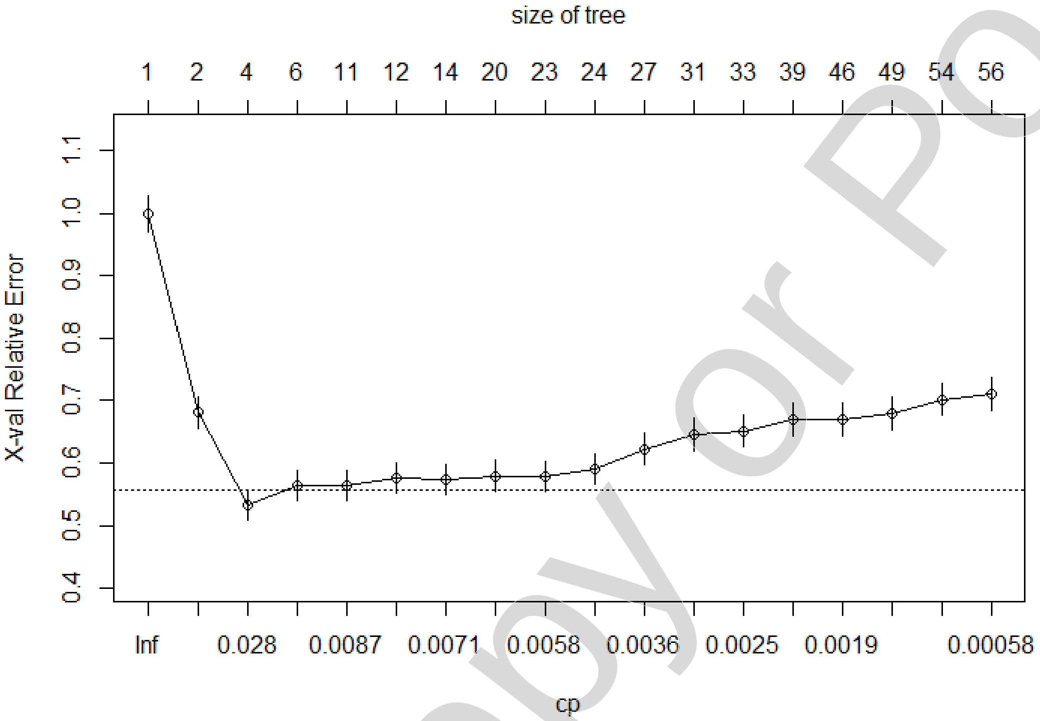
Then prune it:

```
prunned_rpart_tree<-prune(rpart_tree, cp=0.2) # Prune the tree. Play
with cp to see how the resultant tree changes
plot(as.party(prunned_rpart_tree), type = “extended”,gp = gpar(font
size = 7)) # Plot the tree (adjust font size if needed)
```

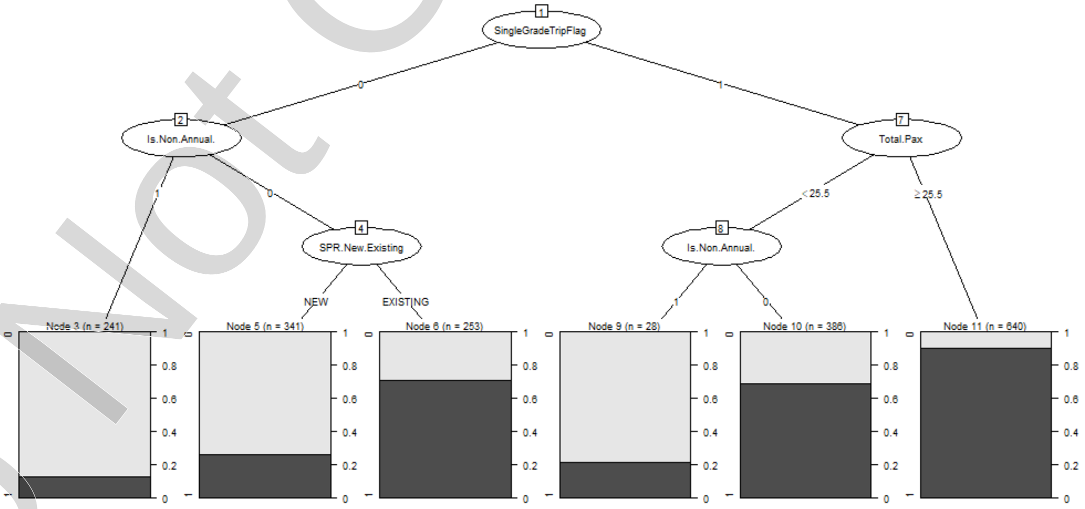
For comparison, I keep `cp=0.005` on the left, and set `cp=0.2` on the right, the difference is evident. A very low `cp` tree is more complex than necessary and is clearly over fit; a very high `cp` tree is “under fit” and misses on many reasonable dependencies in the data.



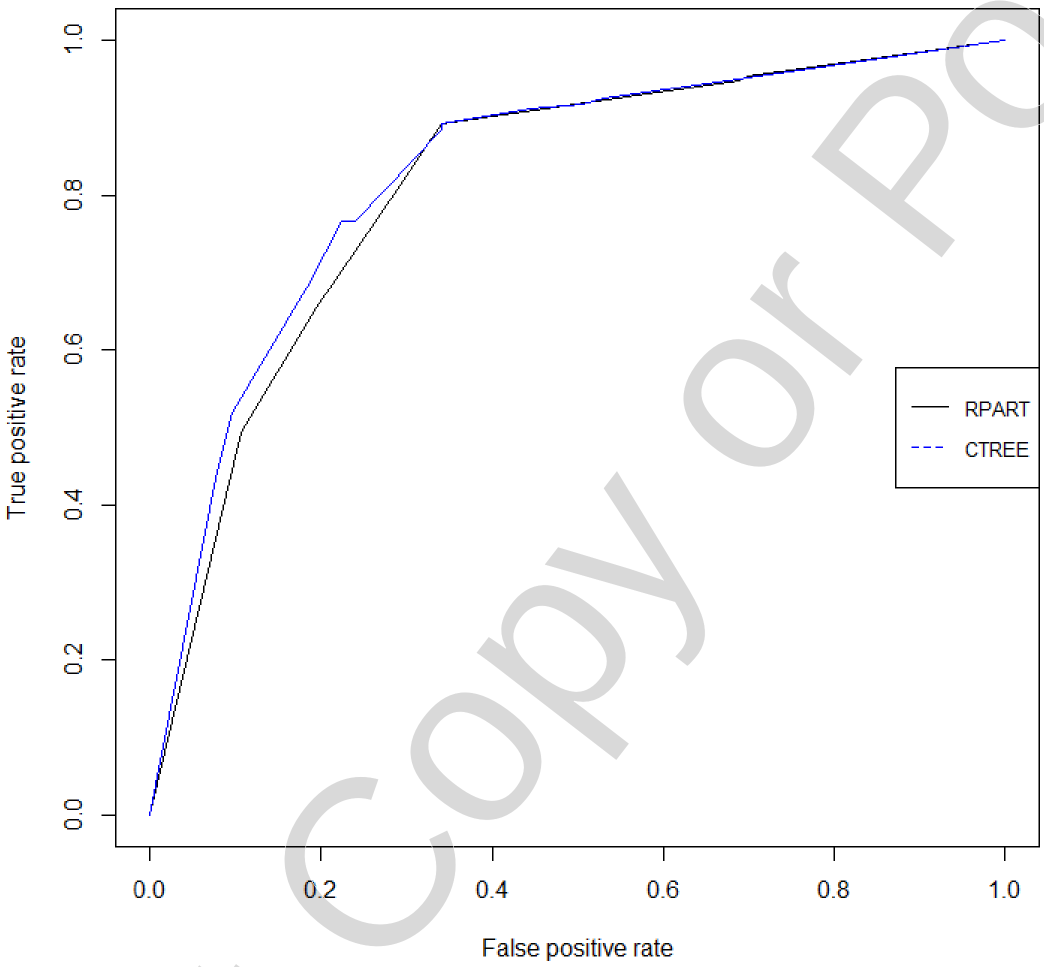
How to select `cp`? R command `plotcp(rpart_tree)` plots error as a function of `cp`:



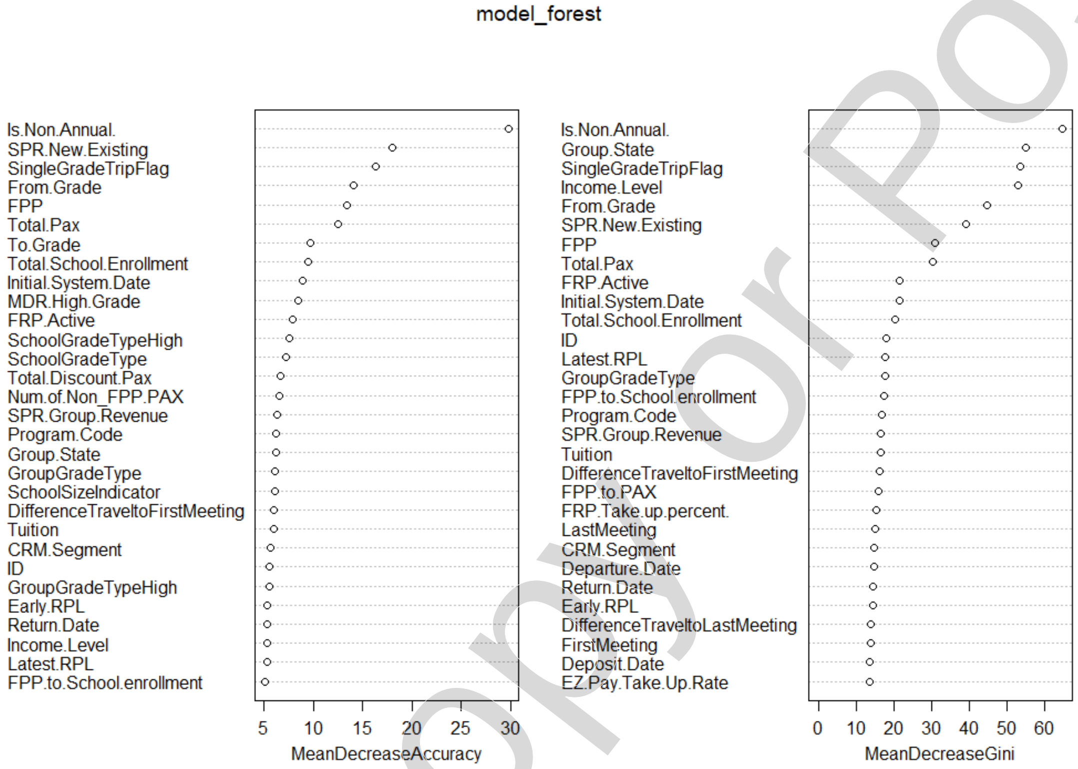
The dashed line is an upper-confidence interval from the minimal error, meaning that a range of `cp`s below the dashed line are indistinguishable. I will use `cp=0.01` and rerun the pruning code from above, which results in the following tree:



This tree has a confusion matrix accuracy of 80%—the highest of all the models considered thus far—yet its ROC curve is inferior to that of the ctree, as is its AUC at 81.33%. This is clearly because this tree is overly simplistic; it does not capture enough customer characteristics to truly distinguish the firm's "best" customers from the "good" ones.



The remaining methods (randomforst, xgboost, nn) also require tuning parameters. They lack the interpretability of CARTs, except perhaps the random forest, which produces a helpful "importance plot" below. Bringing up the analogy between a random forest and decision-making by a committee (e.g., board of directors or parliament) could be insightful too:



Pedagogy

I have used this case in three rather different settings: (1) MBA/EMBA/GEMBA electives on data science, (2) specialized graduate and undergraduate programs in management/business analytics, and (3) executive education programs in analytics. I will briefly summarize the pedagogy for each.

MBA/EMBA/GEMBA elective, "Data Science for Business," "Data Science and Machine Learning for Executives," etc.

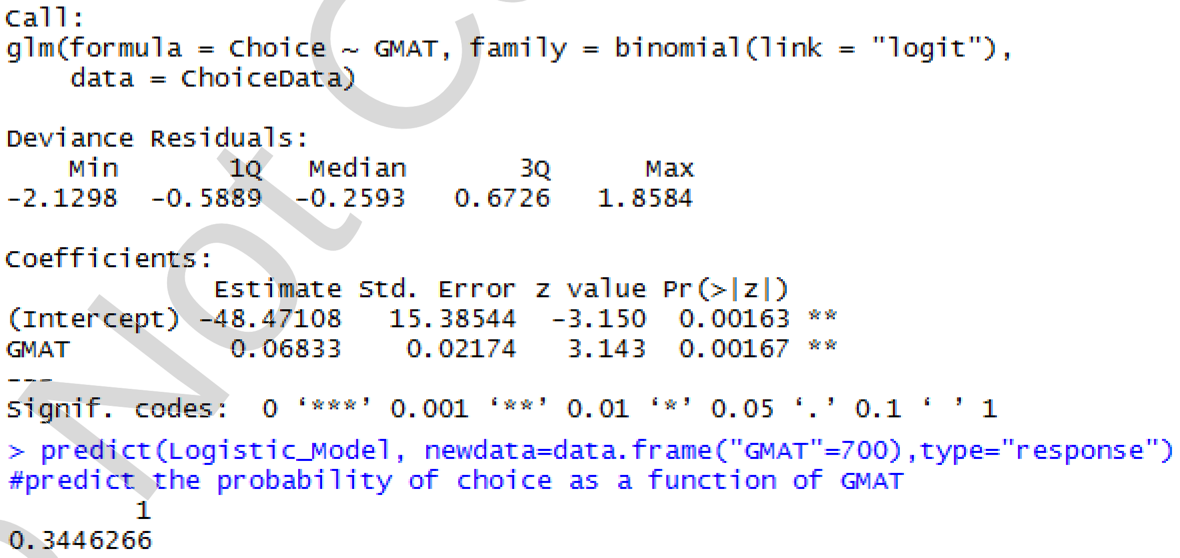
This course immediately follows on the core "statistics-like" course, which is based in Excel. That is, the students have no prior coding experience, and taking this course exposes them to both data science and coding simultaneously. This case, and classification in general, is their third topic in the course; the first is linear regression, which they already know from the core, but now see it implemented with coding in R for the first time, and the second is time series. Each topic consists of two 90-minute classes, with 90-minute TA-run tutorials following the first and third sections.

There is a similar pattern within each section: first, a simple problem with small data is addressed with a very simple code, which is provided to students (in general, all code is provided to students). Students run this code live in class and we discuss the technical issues and interpretation. For this case I assign the technical note "Modeling Discrete Choice: Categorical Dependent Variables, Logistic Regression, and Maximum Likelihood Estimation" (UVA-QA-0779), and the associated one-variable dataset. The goal is to predict a student's choice of a business school based on the GMAT score, and a particular task is to predict the probability for a person with GMAT = 700.

Here is the example of the code. Note that since this is the third topic, the students already know how to load data, check structure, split windows to check the diagnostics of the model, and make a prediction. In other words, this code is effectively identical to the linear regression code (`lm` command in R) they saw earlier. I emphasize that, effectively, the only difference is the "g" in `glm` (bolded below) and the binomial/logit qualifier below (which is, in fact, unnecessary as the `glm` will default to that anyway):

```
ChoiceData<-read.csv(file.choose()) # load data
str(ChoiceData) # make sure that the field types are interpreted
correctly (as numbers/integers, factors, etc.)
Logistic_Model<-glm(Choice    ~    GMAT,    data    =    ChoiceData,
family=“binomial”(link="logit")) # logistic regression is part of the
“generalized linear models” family, hence glm
summary(Logistic_Model) # summary of the model
par(mfrow=c(1,4)) # This command sets the plot window to show 1 row
of 4 plots
plot(Logistic_Model) # check the model using diagnostic plots
predict(Logistic_Model,
newdata=data.frame(“GMAT”=700),type=“response”)   #   predict   the
probability of choice as a function of GMAT
```

And here are the main outputs (the code will also show the diagnostic plots, which I excluded here):



```
Call:
glm(formula = Choice ~ GMAT, family = binomial(link = "logit"),
    data = ChoiceData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1298  -0.5889  -0.2593   0.6726   1.8584

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -48.47108   15.38544  -3.150  0.00163 **
GMAT          0.06833    0.02174   3.143  0.00167 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> predict(Logistic_Model, newdata=data.frame("GMAT"=700),type="response")
#predict the probability of choice as a function of GMAT
        1
0.3446266
```

After this, I introduce the metrics of classification via a mini-lecture: confusion matrix and its statistics, ROC curve, AUC/Gini, and lift.

We first proceed to the logistic regression code for the A case as discussed in the Analysis section above, and then discuss the A case CART. To start, we discuss the conceptual approach, what kinds of data are needed, what we are trying to do, and why; but most time in this session is spent on building the model and interpreting the results.

Two technical issues arise in varying quantities:

1. Several packages need to be installed for the confusion matrix, ROC, etc. [Analysis item #1]. While I advise the students to do this in advance, a significant fraction do not; this can take quite some time (over 10 minutes), and it may result in errors on some of the students' computers. My solution to this problem is to have a teaching assistant in class to address these issues with individual students. One distinction I make is between a warning (e.g., the package was made for the previous version of R, which is just fine) and an error. Both are displayed in red and often scare students.
2. Code builds upon previous steps; hence if a student skips a line of code, or "messes it up" one way or another, the code that follows will not work. A typical yet ironic example of this is loading the data. The code prompts the student to open a file, and for that a file explorer window opens up. Surprisingly, many students do not notice this window, but try to execute the rest of the code, which obviously does not work. It is not uncommon to hear "we are getting many errors" in a situation where one step was not completed, and a student attempted executing the next 10 steps, getting 10 errors as a result.
    a. A related issue is whether R is "busy." For instance, `stepAIC` command can take several minutes. There is no point trying to execute the code that relies on its result. But again, many students will not see the response and will attempt to run it multiple times.

Other than this, a three-hour class (or two 90-minute sessions) runs quickly and leaves students both impressed and overwhelmed. Most realize that they can now do something they were not even close to understanding before, yet classification seems more challenging than they anticipated. Therefore, following with a tutorial on the B case helps build comfort and allows the material to "sink in." Both the class discussion and the tutorial emphasize a point built in the preceding classes: that the R code given to students is effectively a template for classification, and the blocks of code can be easily copied and pasted for use in other projects and in their jobs.

"Predictive Modeling" core course in a specialized analytics program

This case, and the introduction to classification, occupies one seven-hour course module. This module is also third in the course, following the regression and time series modules. The main difference to the MBA elective is that the students are better prepared in terms of coding and R, which is introduced in the preceding course.

The logistic regression portion of the A case is effectively identical to the MBA course, with the only visible difference being that I ask students to explain to others the purposes of the code's different pieces.

For the A case's CART, after showing the students the basics, they break into groups, finalize the classification metrics, and compare the logistic regression results in their study groups, after which one or two groups present their findings.

While in groups, I also ask students to demonstrate over-fitting (i.e., create a table of AUCs for the `rpart` method using various `cp`s on both training and testing data), and observe that when the cp is lowered, the AUC improves with training data but not with testing data. The elaborate trees perform worse—exactly because they too elaborately capture the nuances of the training data, which may not be present in testing—overfitting!

Between the A and B cases, I introduce several other machine learning modeling techniques discussed earlier in the note: random forest, gradient boosting machines, and neural networks.

For the B case, we also resort to group work. I distribute a hard copy of the B case in class, provide the data, and ask students to incorporate the new data into their models and comment on the improvement. I typically give them an hour to work, after which we "regroup." One team presents, and we discuss the issues the class faced. At this point, many teams have working models, but few have good ones. We discuss the need for feature engineering, brainstorm some plausible features in class, and then break again for the groups to implement them and enrich their models.

At the end of the class, we discuss R markdowns (aka "notebooks")—a neat way to combine doing and presenting the analyses in a form of reproducible reports/presentations live-feeding from the data. I show one simple example but leave it for the students to recode their analyses of this case as RMD reports, if they wish to.

Executive education programs in analytics

With a nontechnical executive audience, the use of this case differs significantly from the previous two examples. We start by discussing the meaning of "churn management," or "predicting churn," and the managerial need to separate customers into those who will very likely repurchase, those who are extremely unlikely to repurchase, and those in the middle. We also emphasize the need to understand why a given customer is in each of these groups, and whether the underlying drivers are within the firm's control. This high-level discussion takes 30 to 60 minutes, depending greatly on the participants' desire to share their experiences and practices in their companies.

I then turn the conversation into a somewhat technical domain by role-playing a data scientist who walks the audience through the analyses (following the corresponding section of this note). In particular, I go through the logistic regression part in reasonable detail, running the R script live in class. Once the audience has been introduced to the metrics of classification, I only demonstrate those metrics for a variety of different models (and skip the details). The main goal of spending some time on technicalities is to familiarize the executive audience with the vocabulary and, to some extent, "demystify" machine learning. As one program participant put it, he now knows that when someone talks about a "random forest," this refers to a machine learning technique acting like a hiring committee, and not to a desire to take a walk without the specific forest in mind. I think there is a significant value in demystifying machine learning, and this session is usually well received.

Finally, in many of my programs, this case is followed by a session on hiring and retaining data scientists, so walking through a live demo of what they actually do is also helpful for building context for such a session.

Conclusion

Machine learning holds many promises for business, and as with any technical discipline, it is not without a certain degree of "mystery." This case is efficient in demystifying machine learning based on an easily understood context of predicting customer behavior and classifying which customers will churn and which will be retained. The case presents a realistic data analytic challenge that can be addressed with many machine-learning techniques. The case therefore achieves three simultaneous goals: learn about modeling retention and classification, learn about machine learning and data science, and learn to code. The relative importance of these goals varies drastically among the different courses in which I taught this case. But in all situations, this case has been well received by the students, who found working through it a valuable learning experience.

Exhibit TN1

Retention Modeling at Scholastic Travel Company (A) and (B)

Code: STC (A) Logistic.R

```
if(“pacman” %in% rownames(installed.packages()) == FALSE)
{install.packages(“pacman”)} # Check if you have universal installer package,
install if not

pacman::p_load(“caret”,“ROCR”,“lift”,“glmnet”,“MASS”,“e1071”) # Check, and if
needed install the necessary packages

STCdata_A<-read.csv(file.choose()) # Load the data file to R

str(STCdata_A) # See if some data types were misclassified when importing data
from CSV

# Fixing incorrectly classified data types:
STCdata_A$From.Grade <- as.factor(STCdata_A$From.Grade)
STCdata_A$To.Grade <- as.factor(STCdata_A$To.Grade)
STCdata_A$Is.Non.Annual. <- as.factor(STCdata_A$Is.Non.Annual.)
STCdata_A$Days <- as.factor(STCdata_A$Days)
# STCdata_A$Departure.Date <- as.Date(STCdata_A$Departure.Date, origin=“1899-
12-30”)
# STCdata_A$Return.Date <- as.Date(STCdata_A$Return.Date, origin=“1899-12-30”)
# STCdata_A$Deposit.Date <- as.Date(STCdata_A$Deposit.Date, origin=“1899-12-
30”)
# STCdata_A$Early.RPL <- as.Date(STCdata_A$Early.RPL, origin=“1899-12-30”)
# STCdata_A$Latest.RPL <- as.Date(STCdata_A$Latest.RPL, origin=“1899-12-30”)
# STCdata_A$Initial.System.Date <- as.Date(STCdata_A$Initial.System.Date,
origin=“1899-12-30”)
STCdata_A$CRM.Segment <- as.factor(STCdata_A$CRM.Segment)
STCdata_A$Parent.Meeting.Flag <- as.factor(STCdata_A$Parent.Meeting.Flag)
STCdata_A$MDR.High.Grade <- as.factor(STCdata_A$MDR.High.Grade)
STCdata_A$School.Sponsor <- as.factor(STCdata_A$School.Sponsor)
STCdata_A$NumberOfMeetingswithParents                                       <-
as.factor(STCdata_A$NumberOfMeetingswithParents)
STCdata_A$SingleGradeTripFlag <- as.factor(STCdata_A$SingleGradeTripFlag)
# STCdata_A$FirstMeeting <- as.Date(STCdata_A$FirstMeeting, origin=“1899-12-
30”)
# STCdata_A$LastMeeting <- as.Date(STCdata_A$LastMeeting, origin=“1899-12-30”)
STCdata_A$Retained.in.2012. <- as.factor(STCdata_A$Retained.in.2012.)

# Create a custom function to fix missing values (“NAs”) and preserve the NA
info as surrogate variables
fixNAs<-function(data_frame){
  # Define reactions to NAs
  integer_reac<-0
  factor_reac<-“FIXED_NA”
  character_reac<-“FIXED_NA”
  date_reac<-as.Date(“1900-01-01”)
  # Loop through columns in the data frame and depending on which class the
variable is, apply the defined reaction and create a surrogate

  for (i in 1 : ncol(data_frame)){
    if (class(data_frame[,i]) %in% c("numeric","integer")) {
      if (any(is.na(data_frame[,i]))){
        data_frame[,paste0(colnames(data_frame)[i],"_surrogate")]<-
          as.factor(ifelse(is.na(data_frame[,i]),"1","0"))
        data_frame[is.na(data_frame[,i]),i]<-integer_reac
      }
    } else
      if (class(data_frame[,i]) %in% c("factor")) {
        if (any(is.na(data_frame[,i]))){
          data_frame[,i]<-as.character(data_frame[,i])
          data_frame[,paste0(colnames(data_frame)[i],"_surrogate")]<-
            as.factor(ifelse(is.na(data_frame[,i]),"1","0"))
          data_frame[is.na(data_frame[,i]),i]<-factor_reac
          data_frame[,i]<-as.factor(data_frame[,i])

        }
      } else {
        if (class(data_frame[,i]) %in% c("character")) {
          if (any(is.na(data_frame[,i]))){
            data_frame[,paste0(colnames(data_frame)[i],"_surrogate")]<-
              as.factor(ifelse(is.na(data_frame[,i]),"1","0"))
            data_frame[is.na(data_frame[,i]),i]<-character_reac
          }
        } else {
          if (class(data_frame[,i]) %in% c("Date")) {
            if (any(is.na(data_frame[,i]))){
              data_frame[,paste0(colnames(data_frame)[i],"_surrogate")]<-
                as.factor(ifelse(is.na(data_frame[,i]),"1","0"))
              data_frame[is.na(data_frame[,i]),i]<-date_reac
            }
          }
        }
      }
  }
  return(data_frame)
}

table(STCdata_A$Group.State) # check for rare categories

# Create another a custom function to combine rare categories into "Other."+the
name of the original variable (e.g., Other.State)
# This function has two arguments: the name of the dataframe and the count of
observation in a category to define "rare"
combinerarecategories<-function(data_frame,mincount){
  for (i in 1 : ncol(data_frame)){
    a<-data_frame[,i]
    replace <- names(which(table(a) < mincount))
    levels(a)[levels(a)             %in%             replace]             <-
paste("Other",colnames(data_frame)[i],sep=".")
```

```
    data_frame[,i]<-a }
  return(data_frame) }


# Apply the fixNAs and combinerarecategories functions to the data and then
split it into testing and training data.
STCdata_A<-fixNAs(STCdata_A)
STCdata_A<-combinerarecategories(STCdata_A,10) # combine categories with <10
values in STCdata into “Other”

set.seed(77850) # set a random number generation seed to ensure that the split
is the same everytime
inTrain <- createDataPartition(y = STCdata_A$Retained.in.2012.,
                               p = 1888/2389, list = FALSE)
training <- STCdata_A[ inTrain,]
testing <- STCdata_A[ -inTrain,]

# Select the variables to be included in the “base-case” model
# First include all variables use glm(Retained.in.2012.~ ., data=training,
family=“binomial”(link=“logit”)) Then see which ones have “NA” in coefficients
and remove those
model_logistic<-glm(Retained.in.2012.~ Special.Pay +
                  To.Grade + Group.State + Is.Non.Annual. +
                  Tuition + FRP.Active + FRP.Cancelled + FRP.Take.up.percent.+
                  Cancelled.Pax + Total.Discount.Pax + Initial.System.Date +
                  Poverty.Code + CRM.Segment + School.Type +
                  Parent.Meeting.Flag + MDR.Low.Grade + MDR.High.Grade +
                  Total.School.Enrollment + EZ.Pay.Take.Up.Rate +
                  School.Sponsor + SPR.New.Existing + FPP + FirstMeeting +
                  LastMeeting + DifferenceTraveltoFirstMeeting+
                  DepartureMonth + MajorProgramCode + SingleGradeTripFlag +
                  FPP.to.School.enrollment + FPP.to.PAX + SchoolSizeIndicator,
            data=training, family=“binomial”(link=“logit”))

summary(model_logistic)

# to add surrogates paste this to the list of variables; note, it will run quite
a bit slower
# Special.Pay_surrogate + Early.RPL_surrogate + Latest.RPL_surrogate +
# Initial.System.Date_surrogate + CRM.Segment_surrogate +
MDR.High.Grade_surrogate +
# Total.School.Enrollment_surrogate + FirstMeeting_surrogate +
# LastMeeting_surrogate + DifferenceTraveltoFirstMeeting_surrogate +
# DifferenceTraveltoLastMeeting_surrogate + FPP.to.School.enrollment_surrogate

## The model clearly has too many variables, most of which are insignificant

## Stepwise regressions. There are three approaches to running stepwise
regressions: backward, forward and “both”
## In either approach we need to specify criterion for inclusion/exclusion.
Most common ones: based on information criterion (e.g., AIC) or based on
significance
```

```
model_logistic_stepwiseAIC<-stepAIC(model_logistic,direction = c(“both”),trace
= 1) # AIC stepwise
summary(model_logistic_stepwiseAIC)

par(mfrow=c(1,4))
plot(model_logistic_stepwiseAIC) # Error plots: similar in nature to lm plots.
par(mfrow=c(1,1))

### Finding predictions: probabilities and classification
logistic_probabilities<-
predict(model_logistic_stepwiseAIC,newdata=testing,type=“response”) # Predict
probabilities
logistic_classification<-rep(“1”,500)
logistic_classification[logistic_probabilities<0.6073]=“0” # Predict
classification using 0.6073 threshold. Why 0.6073 - that's the average
probability of being retained in the data. An alternative code:
logistic_classification <- as.integer(logistic_probabilities >
mean(testing$Retained.in.2012. == “1”))
logistic_classification<-as.factor(logistic_classification)

### Confusion matrix
confusionMatrix(logistic_classification,testing$Retained.in.2012.,positive =
“1”)

#### ROC Curve
logistic_ROC_prediction <- prediction(logistic_probabilities,
testing$Retained.in.2012.)
logistic_ROC <- performance(logistic_ROC_prediction,“tpr”,“fpr”) # Create ROC
curve data
plot(logistic_ROC) # Plot ROC curve

#### AUC (area under curve)
AUC.tmp <- performance(logistic_ROC_prediction,“auc”) # Create AUC data
logistic_AUC <- as.numeric(AUC.tmp@y.values) # Calculate AUC
logistic_AUC # Display AUC value: 90+% - excellent, 80-90% - very good, 70-80%
- good, 60-70% - so so, below 60% - not much value

#### Lift chart
plotLift(logistic_probabilities, testing$Retained.in.2012., cumulative = TRUE,
n.buckets = 10)

### An alternative way is to plot a Lift curve not by buckets, but on all data
points
Lift <- performance(logistic_ROC_prediction,“lift”,“rpp”)
plot(Lift)
```

Exhibit TN2

Retention Modeling at Scholastic Travel Company (A) and (B)

Code: STC (B) Logistic.R

```
str(STCdata_B) # See if some data types were misclassified when importing
data from CSV

STCdata_merged = merge(STCdata_A, STCdata_B, by= ‘ID’) # merge the data

# Apply the fixNAs and combine categories functions to the merged data and
then split it into testing and training data.
STCdata_merged<-fixNAs(STCdata_merged)
STCdata_merged<-combinerarecategories(STCdata_merged, 10)

training <- STCdata_merged[ inTrain,]
testing <- STCdata_merged[ -inTrain,]

# Add the new NPS variables to the regression formula
model_logistic_B<-glm(Retained.in.2012.~ Special.Pay +
                      To.Grade + Group.State + Is.Non.Annual. +
                      Tuition + FRP.Active + FRP.Cancelled +FRP.Take.up.percent.+
                      Cancelled.Pax + Total.Discount.Pax + Initial.System.Date +
                      Poverty.Code +CRM.Segment+School.Type +Parent.Meeting.Flag+
                      MDR.Low.Grade + MDR.High.Grade + Total.School.Enrollment +
                      EZ.Pay.Take.Up.Rate + School.Sponsor +
                      SPR.New.Existing + FPP + FirstMeeting + LastMeeting +
                      DifferenceTraveltoFirstMeeting + DepartureMonth +
                      MajorProgramCode + SingleGradeTripFlag +
                      FPP.to.School.enrollment + FPP.to.PAX + SchoolSizeIndicator
                      + NPS.2011 + NPS.2010 + NPS.2009 + NPS.2008,
               data=training, family=“binomial”(link=“logit”))

model_logistic_stepwiseAIC_B<-stepAIC(model_logistic_B, direction =
c(“both”),trace = 1) # AIC stepwise
summary(model_logistic_stepwiseAIC_B)

par(mfrow=c(1,4))
plot(model_logistic_stepwiseAIC_B) # Error plots: similar nature to lm plots
par(mfrow=c(1,1))

### Finding predicitons: probabilities and classification
logistic_probabilities_B<-
predict(model_logistic_stepwiseAIC_B,newdata=testing,type=“response”)
# Predict probabilities
logistic_classification_B<-rep(“1”,500)
logistic_classification_B[logistic_probabilities_B<0.6073]=“0” # Predict
classification using 0.6073 threshold. Why 0.6073 - that's the average
probability of being retained in the data. An alternative code:
logistic_classification <- as.integer(logistic_probabilities >
mean(testing$Retained.in.2012. == “1”))
```

```
logistic_classification_B<-as.factor(logistic_classification_B)

### Confusion matrix
confusionMatrix(logistic_classification_B,testing$Retained.in.2012.,positive
= “1”)

#### ROC Curve
logistic_ROC_prediction_B <- prediction(logistic_probabilities_B,
testing$Retained.in.2012.)
logistic_ROC_B <- performance(logistic_ROC_prediction_B,“tpr”,“fpr”) # Create
ROC curve data
plot(logistic_ROC_B) # Plot ROC curve

plot(logistic_ROC, add=TRUE, col=“red”) # For comparison, overlay/add the ROC
curve from (A) in red
legend(“right”, legend=c(“(B)”, “(A)”), col=c(“black”, “red”), lty=1:2,
cex=0.8)

#### AUC (area under curve)
auc.tmp_B <- performance(logistic_ROC_prediction_B,“auc”) # Create AUC data
logistic_auc_testing_B <- as.numeric(auc.tmp_B@y.values) # Calculate AUC
logistic_auc_testing_B # Display AUC value: 90+% - excellent, 80-90% - very
good, 70-80% - good, 60-70% - so so, below 60% - not much value

#### Lift chart
plotLift(logistic_probabilities_B, testing$Retained.in.2012., cumulative =
TRUE, n.buckets = 10) # Plot Lift chart

### An alternative way is to plot a Lift curve not by buckets, but on all
data points
Lift_B <- performance(logistic_ROC_prediction_B,“lift”,“rpp”)
plot(Lift_B)

plot(Lift, add=TRUE, col=“red”) # For comparison, overlay/add the Lift curve
from (A) in red
legend(“right”, legend=c(“(B)”, “(A)”), col=c(“black”, “red”), lty=1:2,
cex=0.8)
```

Exhibit TN3

Retention Modeling at Scholastic Travel Company (A) and (B)

Code: STC (AB) CART.R

```
if(“pacman” %in% rownames(installed.packages()) == FALSE)
{install.packages(“pacman”)} # Check if you have universal installer package,
install if not

pacman::p_load(“caret”,“partykit”,“ROCR”,“lift”,“rpart”,“e1071”)

# Load the data, correct mis-classified datafields, fixNAs -- same as you did
in the logistic regression file
# To ensure “apples-to-apples” comparisons with logistic regression, use the
same training and testing -- the code below only works in the same R session
after you've ran the logistic regression code

# There are two families of CART algorithms: conditional inference trees (ctree
function; caret package) and recursive partitioning (rpart function; partykit
package)

# CTREE

ctree_tree<-ctree(Retained.in.2012.~.,data=training) # Run ctree on training
data
plot(ctree_tree, gp = gpar(fontsize = 7)) # Plot the tree (adjust font size if
needed)

### Finding predicitons: probabilities and classification
ctree_probabilities<-predict(ctree_tree,newdata=testing,type=“prob”) # Predict
probabilities -- with CTREE it is an array with 2 columns: for not retained
(class 0) and for retained (class 1)
ctree_classification<-rep(“1”,500)
ctree_classification[ctree_probabilities[,2]<0.6073]=“0”          # Predict
classification using 0.6073 threshold. Why 0.6073 – that’s the average
probability of being retained in the data.
confusionMatrix(ctree_classification,testing$Retained.in.2012.,positive = “1”)
# Display confusion matrix

# There is also a “shortcut” ctree_prediction<-
predict(ctree_tree,newdata=testing, type=“response”)
# But it by default uses threshold of 50%: works correctly for perfectly balanced
data, but incorrectly otherwise


#### ROC Curve
ctree_ROC_prediction <- prediction(ctree_probabilities[,2],
testing$Retained.in.2012.) # Calculate errors
ctree_ROC <- performance(ctree_ROC_prediction,“tpr”,“fpr”) # Create ROC curve
data
plot(ctree_ROC) # Plot ROC curve
```

```
#### AUC (area under curve)
AUC.tmp <- performance(ctree_ROC_prediction,“auc”) # Create AUC data
ctree_AUC <- as.numeric(AUC.tmp@y.values) # Calculate AUC
ctree_AUC # Display AUC value: 90+% - excellent, 80-90% - very good, 70-80% -
good, 60-70% - so so, below 60% - not much value

#### Lift chart
plotLift(ctree_probabilities[,2], testing$Retained.in.2012., cumulative =
TRUE, n.buckets = 10) # Plot Lift chart

### An alternative way is to plot a Lift curve not by buckets, but on all data
points
Lift_ctree <- performance(ctree_ROC_prediction,“lift”,“rpp”)
plot(Lift_ctree)

# RPART
# The rpart method has an important “complexity parameter”, cp, which determines
how big the tree is.

rpart_cp = rpart.control(cp = 0.0005)

rpart_tree<-rpart(Retained.in.2012.~.,data=training, method=“class”,
control=rpart_cp) # Run rpart on training data

printcp(rpart_tree) # Understand the relationship between the error and cp
plotcp(rpart_tree) # As a rule of thumb pick up the largest cp which does not
give a substantial drop in error

prunned_rpart_tree<-prune(rpart_tree, cp=0.01) #P rune the tree. Play with cp
to see how the resultant tree changes
plot(as.party(prunned_rpart_tree), type = “extended”,gp = gpar(font size = 7))
# Plot the tree (adjust font size if needed)

### Finding predicitons: probabilities and classification
rpart_probabilities<-predict(prunned_rpart_tree,newdata=testing,type=“prob”)
# Predict probabilities -- with RPART it is an array with 2 columns: for not
retained (class 0) and for retained (class 1)
rpart_classification<-rep(“1”,500)
rpart_classification[rpart_probabilities[,2]<0.6073]=“0” # Predict
classification using 0.6073 threshold. Why 0.6073 - that's the average
probability of being retained in the data. An alternative code:
logistic_classification <- as.integer(logistic_probabilities >
mean(testing$Retained.in.2012. == “1”))

confusionMatrix(rpart_classification,testing$Retained.in.2012.,positive = “1”)
# Display confusion matrix

# There is also a “shortcut” rpart_prediction<-
predict(rpart_tree,newdata=testing, type=“response”)
# But it by default uses threshold of 50%: works correctly for perfectly balanced
data, but incorrectly otherwise
```

```
#### ROC Curve
rpart_ROC_prediction <- prediction(rpart_probabilities[,2],
testing$Retained.in.2012.) # Calculate errors
rpart_ROC <- performance(rpart_ROC_prediction,“tpr”,“fpr”) # Create ROC curve
data
plot(rpart_ROC) # Plot ROC curve

plot(ctree_ROC, add=TRUE, col=“blue”) # For comparison, overlay/add the ROC
curve from CTREE in blue
legend(“right”, legend=c(“RPART”, “CTREE”), col=c(“black”, “blue”), lty=1:2,
cex=0.8)

#### AUC (area under curve)
AUC.tmp <- performance(rpart_ROC_prediction,“auc”) # Create AUC data
rpart_AUC <- as.numeric(AUC.tmp@y.values) # Calculate AUC
rpart_AUC # Display AUC value: 90+% - excellent, 80-90% - very good, 70-80% -
good, 60-70% - so so, below 60% - not much value

#### Lift chart
plotLift(rpart_probabilities[,2], testing$Retained.in.2012., cumulative =
TRUE, n.buckets = 10) # Plot Lift chart

### An alternative way is to plot a Lift curve not by buckets, but on all data
points
Lift_rpart <- performance(rpart_ROC_prediction,“lift”,“rpp”)
plot(Lift_rpart)

# For comparison, overlay/add the Lift curve from CTREE in blue
plot(Lift_ctree, add=TRUE, col=“blue”)
legend(“right”, legend=c(“ctree”, “rpart”), col=c(“blue”, “black”), lty=1:2,
cex=0.8)
```

Exhibit TN4

Retention Modeling at Scholastic Travel Company (A) and (B)

Code: STC (AB) Random Forest.R

```
if(“pacman” %in% rownames(installed.packages()) == FALSE)
{install.packages(“pacman”)} # Check if you have universal installer package,
install if not

pacman::p_load(“caret”,“ROCR”,“lift”,“randomForest”) # Check, and if needed
install the necessary packages

# Load the data, correct mis-classified datafields, fixNAs -- same as you did
in the logistic regression file
# To ensure “apples-to-apples” comparisons with logistic regression, use the
same training and testing -- the code below only works in the same R session
after you've ran the logistic regression code


model_forest <- randomForest(Retained.in.2012.~ ., data=training,
                             importance=TRUE,proximity=TRUE,
                             cutoff = c(0.5, 0.5),type=“classification”)
# cutoffs need to be determined for class 0 and class 1. By default 50/50, but
need not be those necessarily
print(model_forest)
plot(model_forest)
importance(model_forest)
varImpPlot(model_forest)

### Finding predicitons: probabilities and classification
forest_probabilities<-predict(model_forest,newdata=testing,type=“prob”)
# Predict probabilities -- an array with 2 columns: for not retained (class 0)
and for retained (class 1)
forest_classification<-rep(“1”,500)
forest_classification[forest_probabilities[,2]<0.5]=“0” # Predict
classification using 0.5 threshold. Why 0.5 and not 0.6073? Use the same as in
cutoff above
confusionMatrix(forest_classification,testing$Retained.in.2012.,positive =
“1”) # Display confusion matrix. Note, confusion matrix actually displays a
better accuracy with threshold of 50%

# There is also a “shortcut” forest_prediction<-
predict(model_forest,newdata=testing, type=“response”)
# But it by default uses threshold of 50%: actually works better (more accuracy)
on this data


#### ROC Curve
forest_ROC_prediction <- prediction(forest_probabilities[,2],
testing$Retained.in.2012.) # Calculate errors
```

```
forest_ROC <- performance(forest_ROC_prediction,“tpr”,“fpr”) # Create ROC curve
data
plot(forest_ROC) # Plot ROC curve

#### AUC (area under curve)
AUC.tmp <- performance(forest_ROC_prediction,“auc”) # Create AUC data
forest_AUC <- as.numeric(AUC.tmp@y.values) # Calculate AUC
forest_AUC # Display AUC value: 90+% - excellent, 80-90% - very good, 70-80% -
good, 60-70% - so so, below 60% - not much value

#### Lift chart
plotLift(forest_probabilities[,2],   testing$Retained.in.2012.,  cumulative  =
TRUE, n.buckets = 10) # Plot Lift chart

### An alternative way is to plot a Lift curve not by buckets, but on all data
points
Lift_forest <- performance(forest_ROC_prediction,“lift”,“rpp”)
plot(Lift_forest)
```

Exhibit TN5

Retention Modeling at Scholastic Travel Company (A) and (B)

Code: STC (AB) Xgboost.R

```
if(“pacman” %in% rownames(installed.packages()) == FALSE)
{install.packages(“pacman”)} # Check if you have universal installer package,
install if not

pacman::p_load(“caret”,“ROCR”,“lift”,“xgboost”) # Check, and if needed install
the necessary packages

# Load the data, correct mis-classified datafields, fixNAs -- same as you did
in the logistic regression file
# To ensure “apples-to-apples” comparisons with logistic regression, use the
same training and testing -- the code below only works in the same R session
after you've ran the logistic regression code


training.x <-model.matrix(Retained.in.2012.~ ., data = training)
testing.x <-model.matrix(Retained.in.2012.~ ., data = testing)

model_XGboost<-xgboost(data = data.matrix(training.x[,-1]),
                       label=
as.numeric(as.character(training$Retained.in.2012.)),
                       eta = 0.1,
                       max_depth = 20,
                       nround=50,
                       objective = “binary:logistic”)

XGboost_prediction<-predict(model_XGboost,newdata=testing.x[,-1],
type=“response”) # Predict classification (for confusion matrix)
confusionMatrix(ifelse(XGboost_prediction>0.6073,1,0),testing$Retained.in.201
2.,positive = “1”) # Display confusion matrix

#### ROC Curve
XGboost_pred_testing <- prediction(XGboost_prediction,
testing$Retained.in.2012.) # Calculate errors
XGboost_ROC_testing <- performance(XGboost_pred_testing,“tpr”,“fpr”) #Create
ROC curve data
plot(XGboost_ROC_testing) # Plot ROC curve

#### AUC
auc.tmp <- performance(XGboost_pred_testing,“auc”) # Create AUC data
XGboost_auc_testing <- as.numeric(auc.tmp@y.values) # Calculate AUC
XGboost_auc_testing # Display AUC value: 90+% - excellent, 80-90% - very good,
70-80% - good, 60-70% - so so, below 60% - not much value

#### Lift chart
plotLift(XGboost_prediction, testing$Retained.in.2012., cumulative = TRUE,
n.buckets = 10) # Plot Lift chart
```

Exhibit TN6

Retention Modeling at Scholastic Travel Company (A) and (B)

Code: STC (AB) NN.R

```
if(“pacman”      %in%      rownames(installed.packages())      ==      FALSE)
{install.packages(“pacman”)} # Check if you have universal installer package,
install if not

pacman::p_load(“nnet”,“caret”,“lift”)  #  Check,  and  if  needed  install  the
necessary packages

# Load the data, correct mis-classified datafields, fixNAs -- same as you did
in the logistic regression file
# To ensure “apples-to-apples” comparisons with logistic regression, use the
same training and testing -- the code below only works in the same R session
after you've ran the logistic regression code

## start Neural Network analysis

my.grid <- expand.grid( .size = c(1,2,4),.decay = c(0.25,1,2)) # Tuning grid
for Neural Net

model_NN <- train(Retained.in.2012.~ Special.Pay +
            To.Grade + Group.State + Is.Non.Annual. + Tuition +
            FRP.Active + FRP.Cancelled + FRP.Take.up.percent. +
            Cancelled.Pax + Total.Discount.Pax + Initial.System.Date +
                  Poverty.Code    +    CRM.Segment    +    School.Type    +
                  Parent.Meeting.Flag  +  MDR.Low.Grade  +  MDR.High.Grade  +
                  Total.School.Enrollment     +     EZ.Pay.Take.Up.Rate     +
                  School.Sponsor +  SPR.New.Existing + FPP + FirstMeeting +
                  LastMeeting     +     DifferenceTraveltoFirstMeeting     +
                  DepartureMonth  + MajorProgramCode + SingleGradeTripFlag +
                  FPP.to.School.enrollment + FPP.to.PAX + SchoolSizeIndicator,
                  data = training, method = “nnet”, tuneGrid = my.grid, trace
                  = TRUE, na.action =na.omit)

plot(model_NN) # Visualize the relationship between the number of layers, decay,
and accuracy

NN_prediction<-predict(model_NN, newdata=testing) # Predict classification
confusionMatrix(NN_prediction,testing$Retained.in.2012.,positive = “1”)

# ROC curve and AUC

NN_probabilities_testing  <-predict(model_NN,newdata=testing,type  =  “prob”)
# Predict probabilities
NN_pred_testing          <-          prediction(NN_probabilities_testing[,2],
testing$Retained.in.2012.) # Calculate errors
NN_ROC_testing <- performance(NN_pred_testing,“tpr”,“fpr”) # Create ROC curve
data
```

```
plot(NN_ROC_testing) # Plot ROC curve

auc.tmp <- performance(NN_pred_testing,“auc”) # Create AUC data
NN_auc_testing <- as.numeric(auc.tmp@y.values) # Calculate AUC
NN_auc_testing # Display AUC value: 90+% - excellent, 80-90% - very good, 70-
80% - good, 60-70% - so so, below 60% - not much value

plotLift(NN_prediction,     testing$Retained.in.2012.,   cumulative   =   TRUE,
n.buckets = 10) # Plot Lift chart
```