Mini Project 1: Predicting Customer Churn

Chris Corona, Jenish Simon, Rishi Borkar

2022-10-07

Abstract

In this mini-project, our task is to fit and evaluate several different types of models for classification. The data set is from a telephone service provider that tracked 20 metrics - 5 categorical and 15 quantitative - to try to predict whether or not the customer would churn. There are 3333 observations in this unbalanced data set with 2850 `FALSE` responses and 483 `TRUE` responses. After fitting various models, we compared the precision, recall, and F1 scores to determine which model has the best performance.

1 Executive Summary

Our analysis of the telecom customer data set achieved great results. We are able to predict with 89% overall accuracy whether or not a customer will churn. When our model predicts a churn, it predicts correctly 58% of the time. And of all the true churns, we are predicting 71% of them. These are decent numbers for such an unbalanced data set. We discovered that customers who churn have the following properties. They tend to have international plans, they tend to not have voice mail plans and have fewer voicemail messages, they tend to have slightly more day/evening/night/international minutes, and they tend to make more customer service calls. We also found that the customer's state, area code, account length, and number of calls were not related to churning.

2 Data Preprocessing

The data set is from a telephone service provider that tracked 20 metrics - 5 categorical and 15 quantitative - to try to predict whether or not the customer would churn. There are 3333 observations in this unbalanced data set with 2850 `FALSE` responses and 483 `TRUE` responses. As a preprocessing step, we split our data into train and test sets. We split at 70% to 30% so there are 2324 training and 1009 testing observations. Of the training observations, there are 1982 `FALSE` responses and 342 `TRUE` responses. Of the testing observations, there are 868 `FALSE` responses and 141 `TRUE` responses. This seems like reasonable unbalance compared to the original data set.

```r
# read in the data set
library(tidyverse)
data <- read_csv("Customer_telecom.csv")

# split data into train and test sets
set.seed(5544)
split_index <- sample(c(TRUE,FALSE), nrow(data), replace=TRUE, prob=c(0.7, 0.3))
train <- data[split_index,]
test <- data[!split_index,]
```

```r
# train and test sets seem reasonably unbalanced
table(train$churn)
```

```
##
## FALSE  TRUE
##  1982   342
```

```r
table(test$churn)
```

```
##
## FALSE  TRUE
##   868   141
```

3 Exploratory Analysis

In our Exploratory Data Analysis (EDA) we rigorously reviewed the data for any issues and reduced the number of predictors from 20 down to just 9. We first checked for any missing data; there is none. Then we checked the balance of the data set; it is very unbalanced with 2850 `FALSE` responses and 483 `TRUE` responses. We looked at the summary statistics to find any outliers; it appears that there are no extreme observations. The next task was to attempt to reduce the number of predictors by dropping any that are uncorrelated with the response or are strongly correlated with another predictor. We discovered that the `total_x_minutes` is strongly correlated with `total_x_charge` so we could drop one of them. We then dropped all the predictors that are uncorrelated with the response: `phone_number`, `state`, `area_code`, `account_length`, and `x_calls`.

```r
# take a peak at the data
head(train)
```

```
## # A tibble: 6 x 21
##   state accoun~1 area ~2 phone~3 inter~4 voice~5 numbe~6 total~7 total~8 total~9
##   <chr>    <dbl> <chr>   <chr>   <chr>   <chr>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KS         128 "\"415~ 382-46~ no      yes          25    265.     110    45.1
## 2 OH         107 "\"415~ 371-71~ no      yes          26    162.     123    27.5
## 3 OH          84 "\"408~ 375-99~ yes     no            0    299.      71    50.9
## 4 OK          75 "\"415~ 330-66~ yes     no            0    167.     113    28.3
## 5 AL         118 "\"510~ 391-80~ yes     no            0    223.      98    38.0
## 6 MA         121 "\"510~ 355-99~ no      yes          24    218.      88    37.1
## # ... with 11 more variables: ‘total eve minutes‘ <dbl>,
## #   ‘total eve calls‘ <dbl>, ‘total eve charge‘ <dbl>,
## #   ‘total night minutes‘ <dbl>, ‘total night calls‘ <dbl>,
## #   ‘total night charge‘ <dbl>, ‘total intl minutes‘ <dbl>,
## #   ‘total intl calls‘ <dbl>, ‘total intl charge‘ <dbl>,
## #   ‘customer service calls‘ <dbl>, churn <lgl>, and abbreviated variable names
## #   1: ‘account length‘, 2: ‘area code‘, 3: ‘phone number‘, ...
```

```r
# immediately drop phone number as a predictor
train <- train[,-4]

# check structure of data
str(train)
```

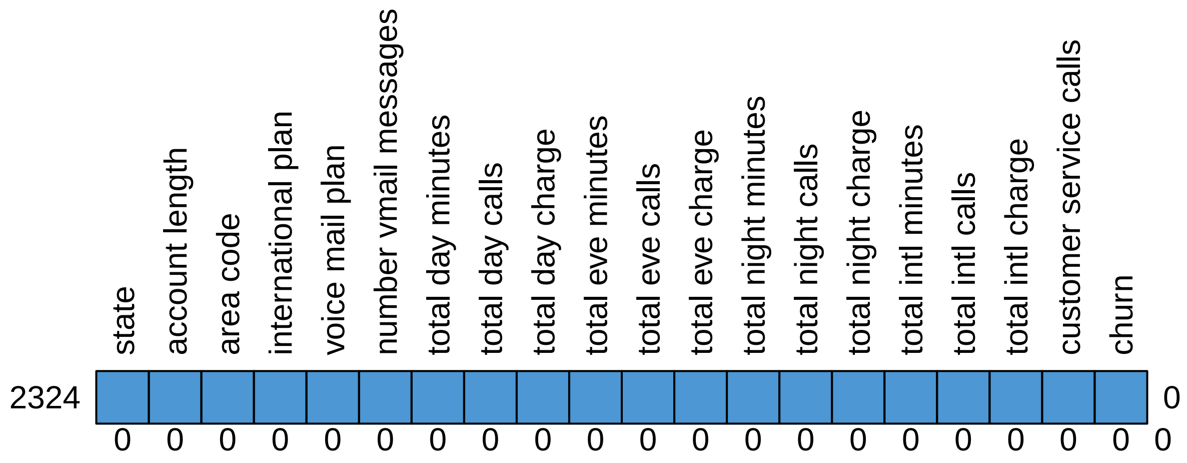
```
## tibble [2,324 x 20] (S3: tbl_df/tbl/data.frame)
##  $ state                 : chr [1:2324] "KS" "OH" "OH" "OK" ...
##  $ account length        : num [1:2324] 128 107 84 75 118 121 141 65 74 168 ...
##  $ area code             : chr [1:2324] "\"415\"" "\"415\"" "\"408\"" "\"415\"" ...
##  $ international plan    : chr [1:2324] "no" "no" "yes" "yes" ...
##  $ voice mail plan       : chr [1:2324] "yes" "yes" "no" "no" ...
##  $ number vmail messages : num [1:2324] 25 26 0 0 0 24 37 0 0 0 ...
##  $ total day minutes     : num [1:2324] 265 162 299 167 223 ...
##  $ total day calls       : num [1:2324] 110 123 71 113 98 88 84 137 127 96 ...
##  $ total day charge      : num [1:2324] 45.1 27.5 50.9 28.3 38 ...
##  $ total eve minutes     : num [1:2324] 197.4 195.5 61.9 148.3 220.6 ...
##  $ total eve calls       : num [1:2324] 99 103 88 122 101 108 111 83 148 71 ...
##  $ total eve charge      : num [1:2324] 16.78 16.62 5.26 12.61 18.75 ...
##  $ total night minutes   : num [1:2324] 245 254 197 187 204 ...
##  $ total night calls     : num [1:2324] 91 103 89 121 118 118 97 111 94 128 ...
##  $ total night charge    : num [1:2324] 11.01 11.45 8.86 8.41 9.18 ...
##  $ total intl minutes    : num [1:2324] 10 13.7 6.6 10.1 6.3 7.5 11.2 12.7 9.1 11.2 ...
##  $ total intl calls      : num [1:2324] 3 3 7 3 6 7 5 6 5 2 ...
##  $ total intl charge     : num [1:2324] 2.7 3.7 1.78 2.73 1.7 2.03 3.02 3.43 2.46 3.02 ...
##  $ customer service calls: num [1:2324] 1 1 2 3 0 3 0 4 0 1 ...
##  $ churn                 : logi [1:2324] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
# change strings into factors
train$state <- factor(train$state)
train$`area code` <- factor(train$`area code`)
train$`international plan` <- factor(train$`international plan`)
train$`voice mail plan` <- factor(train$`voice mail plan`)
# recheck structure of data to make sure factoring worked
str(train)
```

```
## tibble [2,324 x 20] (S3: tbl_df/tbl/data.frame)
##  $ state                 : Factor w/ 51 levels "AK","AL","AR",..: 17 36 36 37 2 20 50 16 40 13 ...
##  $ account length        : num [1:2324] 128 107 84 75 118 121 141 65 74 168 ...
##  $ area code             : Factor w/ 3 levels "\"408\"","\"415\"",..: 2 2 1 2 3 3 2 2 2 1 ...
##  $ international plan    : Factor w/ 2 levels "no","yes": 1 1 2 2 2 1 2 1 1 1 ...
##  $ voice mail plan       : Factor w/ 2 levels "no","yes": 2 2 1 1 1 2 2 1 1 1 ...
##  $ number vmail messages : num [1:2324] 25 26 0 0 0 24 37 0 0 0 ...
##  $ total day minutes     : num [1:2324] 265 162 299 167 223 ...
##  $ total day calls       : num [1:2324] 110 123 71 113 98 88 84 137 127 96 ...
##  $ total day charge      : num [1:2324] 45.1 27.5 50.9 28.3 38 ...
##  $ total eve minutes     : num [1:2324] 197.4 195.5 61.9 148.3 220.6 ...
##  $ total eve calls       : num [1:2324] 99 103 88 122 101 108 111 83 148 71 ...
##  $ total eve charge      : num [1:2324] 16.78 16.62 5.26 12.61 18.75 ...
##  $ total night minutes   : num [1:2324] 245 254 197 187 204 ...
##  $ total night calls     : num [1:2324] 91 103 89 121 118 118 97 111 94 128 ...
##  $ total night charge    : num [1:2324] 11.01 11.45 8.86 8.41 9.18 ...
##  $ total intl minutes    : num [1:2324] 10 13.7 6.6 10.1 6.3 7.5 11.2 12.7 9.1 11.2 ...
##  $ total intl calls      : num [1:2324] 3 3 7 3 6 7 5 6 5 2 ...
##  $ total intl charge     : num [1:2324] 2.7 3.7 1.78 2.73 1.7 2.03 3.02 3.43 2.46 3.02 ...
##  $ customer service calls: num [1:2324] 1 1 2 3 0 3 0 4 0 1 ...
##  $ churn                 : logi [1:2324] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

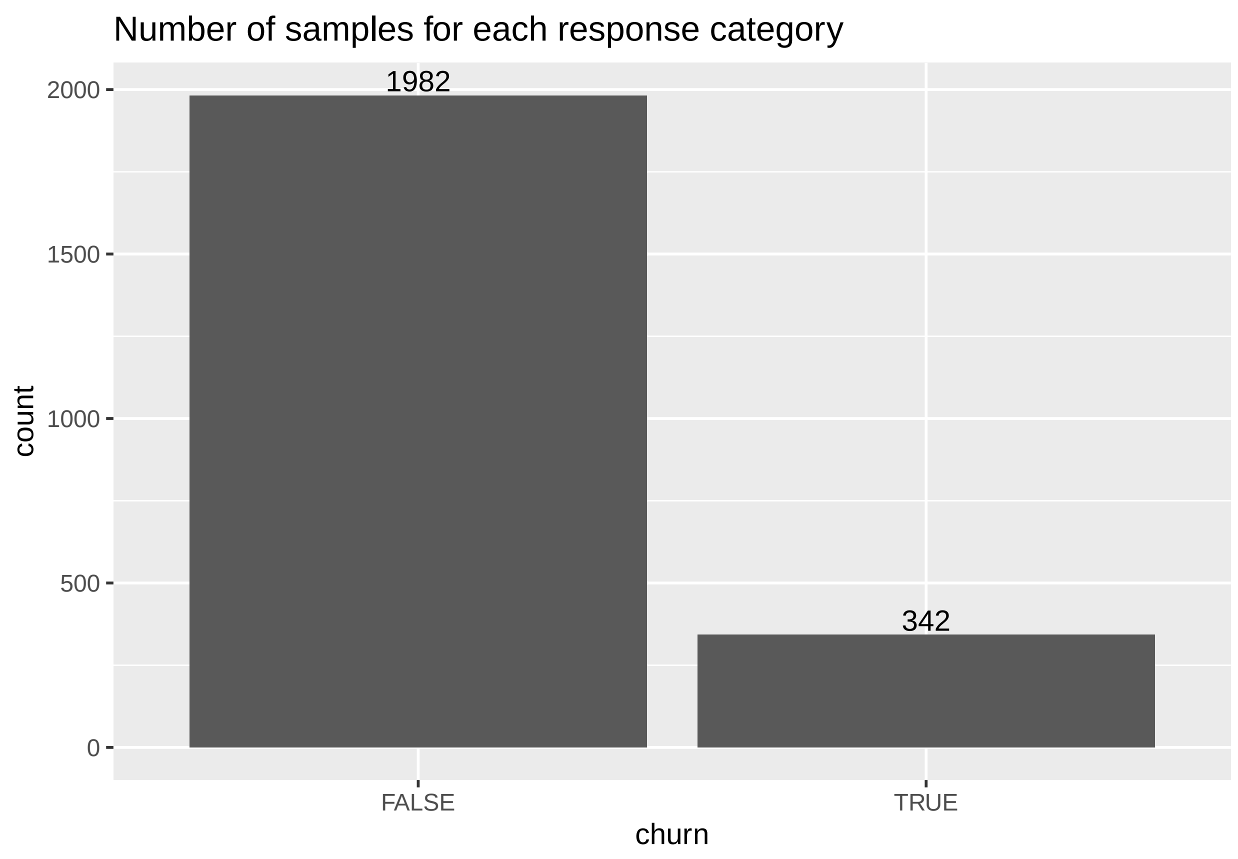
```
# check for missing data - NO MISSING DATA!!
library(mice)
md.pattern(train, rotate.names=TRUE)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```



```
##      state account length area code international plan voice mail plan
## 2324     1              1         1                  1               1
##          0              0         0                  0               0
##      number vmail messages total day minutes total day calls total day charge
## 2324                     1                 1               1                1
##                          0                 0               0                0
##      total eve minutes total eve calls total eve charge total night minutes
## 2324                 1               1                1                   1
##                      0               0                0                   0
##      total night calls total night charge total intl minutes total intl calls
## 2324                 1                  1                  1                1
##                      0                  0                  0                0
##      total intl charge customer service calls churn
## 2324                 1                      1     1 0
##                      0                      0     0  0
```

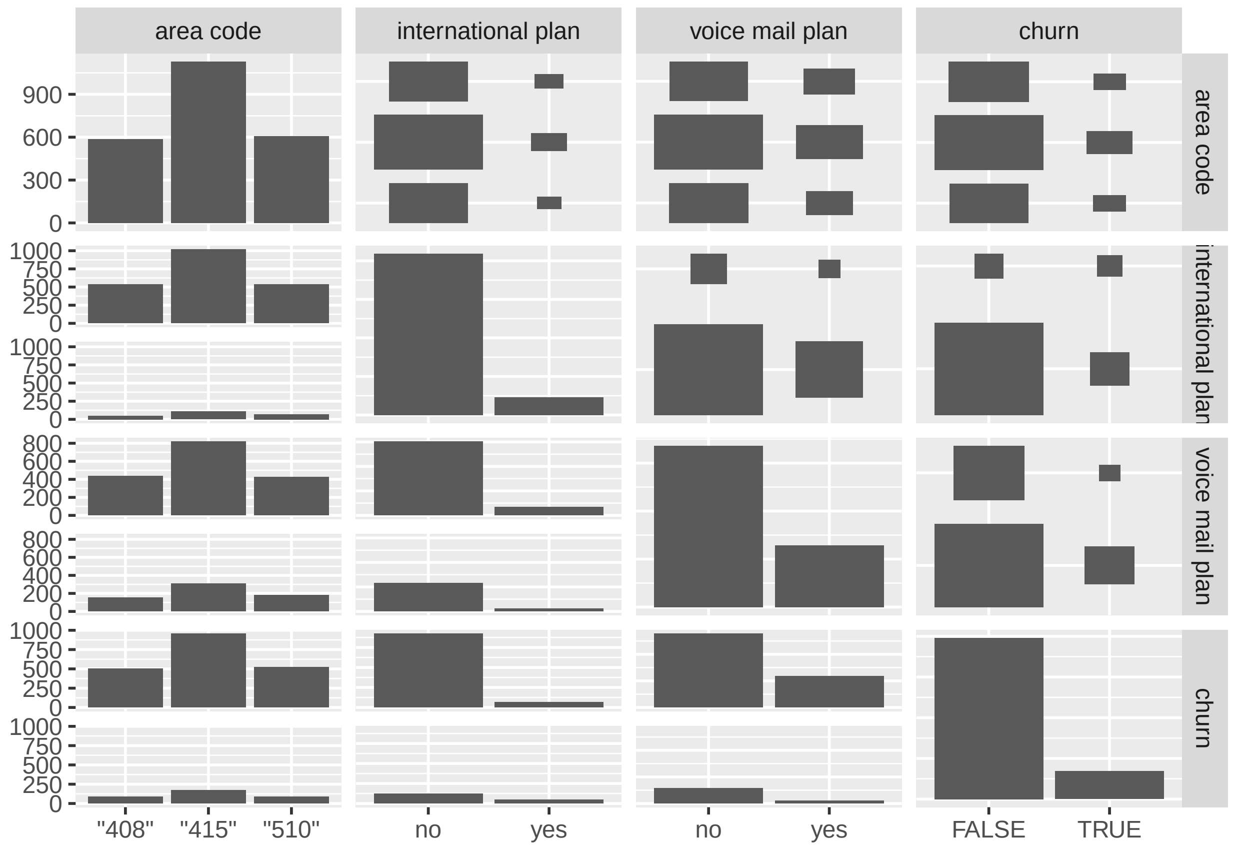
```
# check for balance - NOT BALANCED, FALSE=2850, TRUE=483
library(ggplot2)
ggplot(train, aes(x=churn)) +
  geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust=-0.2) +
  ggtitle("Number of samples for each response category")
```



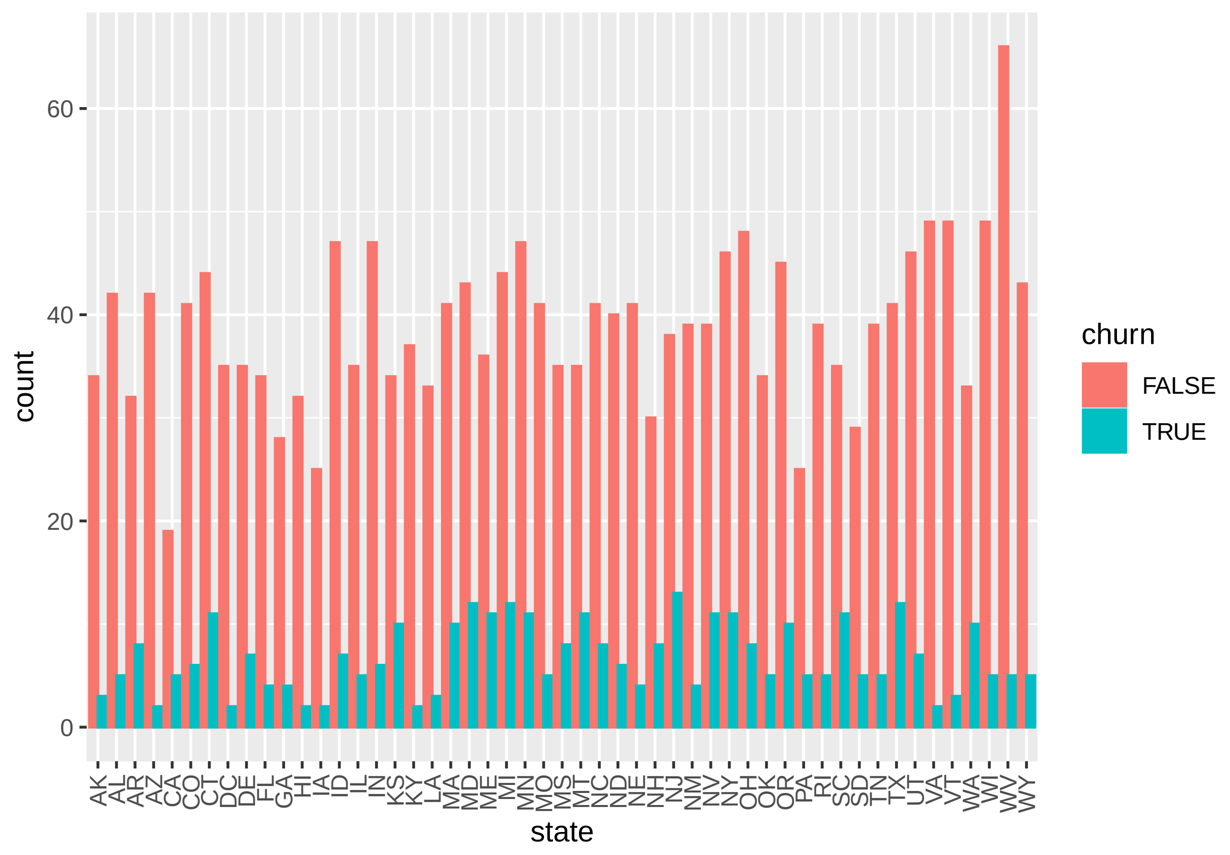
```
# summary statistics to check for any strangeness in the data - NO ISSUES OBSERVED
summary(train)
```

```
##      state      account length   area code    international plan voice mail plan
##  WV     :  71   Min.   :  1.0   "408": 588   no :2093           no :1681
##  MN     :  58   1st Qu.: 74.0   "415":1129   yes: 231           yes: 643
##  NY     :  57   Median :100.0   "510": 607
##  MI     :  56   Mean   :100.6
##  OH     :  56   3rd Qu.:126.0
##  CT     :  55   Max.   :243.0
##  (Other):1971
##  number vmail messages total day minutes total day calls total day charge
##  Min.   : 0.000        Min.   :  0.0     Min.   :  0.0   Min.   : 0.00
##  1st Qu.: 0.000        1st Qu.:144.5     1st Qu.: 87.0   1st Qu.:24.57
##  Median : 0.000        Median :181.5     Median :101.0   Median :30.86
##  Mean   : 7.984        Mean   :180.8     Mean   :100.6   Mean   :30.74
##  3rd Qu.:19.000        3rd Qu.:218.2     3rd Qu.:114.0   3rd Qu.:37.09
##  Max.   :50.000        Max.   :350.8     Max.   :165.0   Max.   :59.64
##
##  total eve minutes total eve calls  total eve charge total night minutes
##  Min.   :  0.0     Min.   :  0.00   Min.   : 0.00    Min.   : 45.0
##  1st Qu.:166.9     1st Qu.: 87.00   1st Qu.:14.19    1st Qu.:167.8
##  Median :201.3     Median :100.00   Median :17.11    Median :201.6
##  Mean   :200.9     Mean   : 99.94   Mean   :17.07    Mean   :201.2
##  3rd Qu.:235.2     3rd Qu.:113.00   3rd Qu.:19.98    3rd Qu.:235.4
##  Max.   :363.7     Max.   :168.00   Max.   :30.91    Max.   :395.0
##
##  total night calls total night charge total intl minutes total intl calls
##  Min.   : 33.0     Min.   : 2.030     Min.   : 0.00      Min.   : 0.000
##  1st Qu.: 87.0     1st Qu.: 7.550     1st Qu.: 8.40      1st Qu.: 3.000
##  Median :100.0     Median : 9.070     Median :10.30      Median : 4.000
##  Mean   :100.3     Mean   : 9.055     Mean   :10.22      Mean   : 4.488
##  3rd Qu.:114.0     3rd Qu.:10.590     3rd Qu.:12.10      3rd Qu.: 6.000
##  Max.   :175.0     Max.   :17.770     Max.   :20.00      Max.   :20.000
##
##  total intl charge customer service calls   churn
##  Min.   :0.000     Min.   :0.000          Mode :logical
##  1st Qu.:2.270     1st Qu.:1.000          FALSE:1982
##  Median :2.780     Median :1.000          TRUE :342
##  Mean   :2.759     Mean   :1.563
##  3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :5.400     Max.   :9.000
##
```

```r
# let's look at the few categorical predictors
library(GGally)
ggpairs(train[,c(3,4,5,20)], upper = list(continuous = GGally::wrap(ggally_cor,
                                               size = 3,
                                               color ="black",
                                               stars = F)))
```



```
# it seems state actually has some predictive capabilities (surprisingly)
# but only a few states are meaningful - it's not worth it to include all 51 states
ggplot(train, aes(x=state, fill=churn, color=churn)) +
  geom_bar(position="dodge") +
  scale_x_discrete(guide = guide_axis(angle = 90))
```



```r
lm.state <- glm(churn ~ state, data=train, family="binomial")
summary(lm.state)
```

```
##
## Call:
## glm(formula = churn ~ state, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7671  -0.6548  -0.5168  -0.3482   2.5451
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.42775    0.60228  -4.031 5.56e-05 ***
## stateAL      0.29952    0.76587   0.391   0.6957
## stateAR      1.04145    0.72041   1.446   0.1483
## stateAZ     -0.61677    0.94151  -0.655   0.5124
## stateCA      1.09275    0.78446   1.393   0.1636
## stateCO      0.50594    0.74418   0.680   0.4966
## stateCT      1.04145    0.69020   1.509   0.1313
## stateDC     -0.43445    0.94408  -0.460   0.6454
## stateDE      0.81831    0.73087   1.120   0.2629
## stateFL      0.28768    0.80135   0.359   0.7196
## stateGA      0.48184    0.80527   0.598   0.5496
## stateHI     -0.34484    0.94551  -0.365   0.7153
## stateIA     -0.09798    0.95013  -0.103   0.9179
## stateID      0.52351    0.72586   0.721   0.4708
## stateIL      0.48184    0.76897   0.627   0.5309
## stateIN      0.36936    0.74208   0.498   0.6187
## stateKS      1.20397    0.70154   1.716   0.0861 .
## stateKY     -0.49002    0.94325  -0.520   0.6034
## stateLA      0.02985    0.85228   0.035   0.9721
## stateMA      1.01676    0.69795   1.457   0.1452
## stateMD      1.15145    0.68508   1.681   0.0928 .
## stateME      1.24212    0.69385   1.790   0.0734 .
## stateMI      1.12847    0.68469   1.648   0.0993 .
## stateMN      0.97550    0.68915   1.416   0.1569
## stateMO      0.32361    0.76625   0.422   0.6728
## stateMS      0.95184    0.71855   1.325   0.1853
## stateMT      1.27030    0.69442   1.829   0.0674 .
## stateNC      0.79362    0.71564   1.109   0.2674
## stateND      0.53063    0.74459   0.713   0.4761
## stateNE      0.10047    0.79821   0.126   0.8998
## stateNH      1.10599    0.72186   1.532   0.1255
## stateNJ      1.35511    0.68263   1.985   0.0471 *
## stateNM      0.15048    0.79899   0.188   0.8506
## stateNV      1.16208    0.69231   1.679   0.0932 .
## stateNY      0.99700    0.68949   1.446   0.1482
## stateOH      0.63599    0.71315   0.892   0.3725
## stateOK      0.51083    0.76952   0.664   0.5068
## stateOR      0.92367    0.69640   1.326   0.1847
## statePA      0.81831    0.77637   1.054   0.2919
## stateRI      0.37362    0.76706   0.487   0.6262
## stateSC      1.27030    0.69442   1.829   0.0674 .
## stateSD      0.66989    0.77281   0.867   0.3860
## stateTN      0.37362    0.76706   0.487   0.6262
## stateTX      1.19908    0.68591   1.748   0.0804 .
## stateUT      0.54502    0.72618   0.751   0.4529
## stateVA     -0.77092    0.93960  -0.820   0.4119
## stateVT     -0.36546    0.84645  -0.432   0.6659
## stateWA      1.23383    0.70217   1.757   0.0789 .
## stateWI      0.14537    0.76364   0.190   0.8490
## stateWV     -0.15247    0.76019  -0.201   0.8410
## stateWY      0.27599    0.76551   0.361   0.7185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1941.7  on 2323  degrees of freedom
## Residual deviance: 1868.2  on 2273  degrees of freedom
## AIC: 1970.2
##
## Number of Fisher Scoring iterations: 5
```

```
lm.areacode <- glm(churn ~ `area code`, data=train, family="binomial")
summary(lm.areacode)
```

```
##
```

```
## Call:
## glm(formula = churn ~ `area code`, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5713  -0.5713  -0.5659  -0.5493   1.9829
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.75070    0.11615 -15.073   <2e-16 ***
## `area code`"415"  0.02061    0.14288   0.144    0.885
## `area code`"510" -0.06432    0.16484  -0.390    0.696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1941.7  on 2323  degrees of freedom
## Residual deviance: 1941.4  on 2321  degrees of freedom
## AIC: 1947.4
##
## Number of Fisher Scoring iterations: 4
```

```r
# drop state and area code
train <- train[,c(-1,-3)]

# now let's look for correlation between quantitative predictors/response
cor(train[,c(1,4:18)])
```

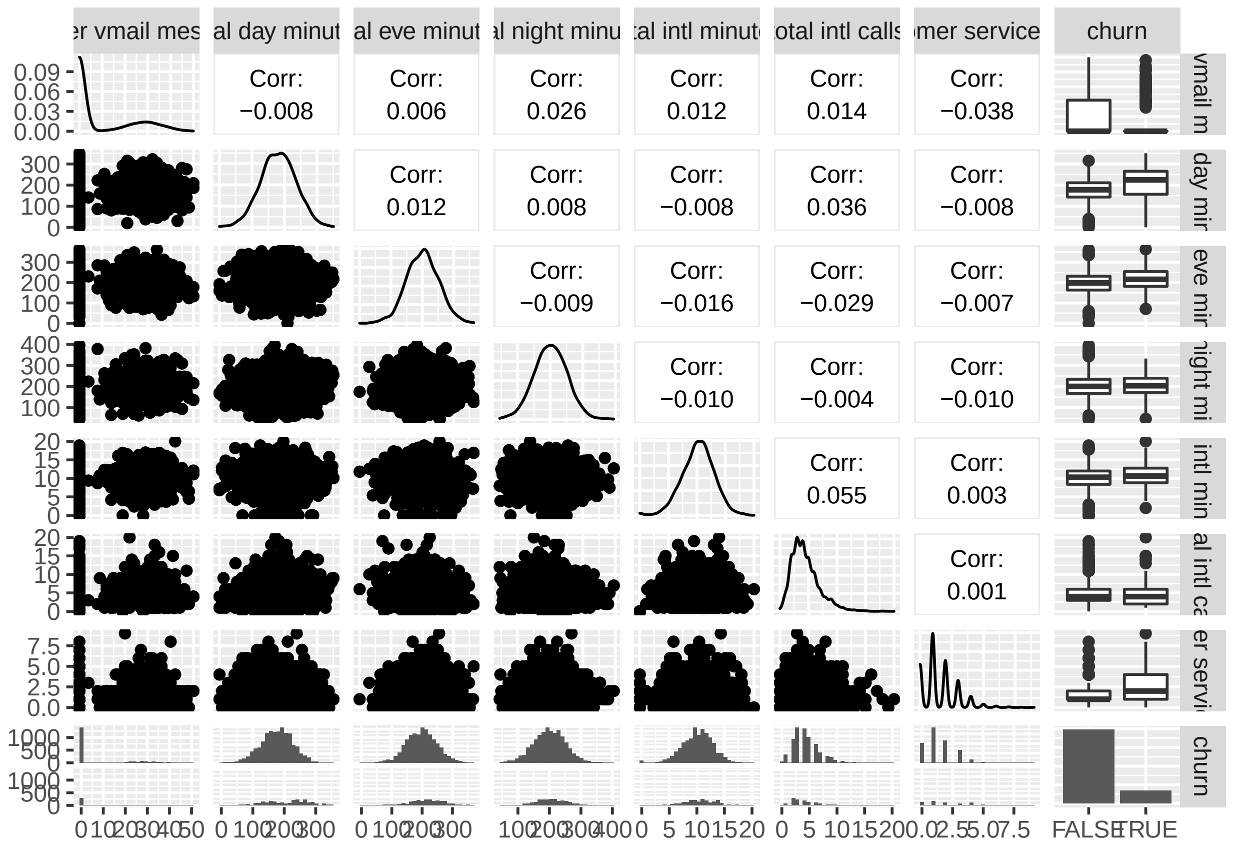
```
##                          account length number vmail messages total day minutes
## account length              1.000000000         -0.0091197514       0.005980054
## number vmail messages      -0.009119751          1.0000000000      -0.007649569
## total day minutes           0.005980054         -0.0076495689       1.000000000
## total day calls             0.057124673         -0.0001827063       0.026704102
## total day charge            0.005974354         -0.0076504970       0.999999953
## total eve minutes          -0.030341595          0.0058867921       0.011733779
## total eve calls             0.035158423         -0.0109235002       0.006402240
## total eve charge           -0.030336228          0.0059090930       0.011715351
## total night minutes        -0.010917426          0.0262121624       0.008101399
## total night calls          -0.014400808          0.0280944151       0.020256262
## total night charge         -0.010932673          0.0262048074       0.008081078
## total intl minutes          0.029865848          0.0118993602      -0.008318503
## total intl calls            0.009182218          0.0136590526       0.035796910
## total intl charge           0.029848620          0.0119368576      -0.008252823
## customer service calls      0.010122307         -0.0380357010      -0.007747051
## churn                       0.024936027         -0.0995517865       0.219118543
##                          total day calls total day charge total eve minutes
## account length              0.0571246735      0.005974354      -0.030341595
## number vmail messages      -0.0001827063     -0.007650497       0.005886792
## total day minutes           0.0267041024      0.999999953       0.011733779
## total day calls             1.0000000000      0.026699790      -0.020477245
## total day charge            0.0266997896      1.000000000       0.011746909
## total eve minutes          -0.0204772452      0.011746909       1.000000000
## total eve calls             0.0048449139      0.006402223       0.006560576
## total eve charge         -0.0204777479      0.011728479       0.999999779
## total night minutes       0.0279805876      0.008102666      -0.008596755
## total night calls        -0.0154157505      0.020260665       0.019546057
## total night charge        0.0279813010      0.008082331      -0.008600289
## total intl minutes        0.0114201070     -0.008322514      -0.015910292
## total intl calls         -0.0018728474      0.035795800      -0.028624071
## total intl charge         0.0115357291     -0.008256843      -0.015928408
## customer service calls   -0.0098893591     -0.007749856      -0.006512435
## churn                     0.0409608887      0.219113840       0.126495726
##                        total eve calls total eve charge total night minutes
## account length            0.0351584235     -0.030336228        -0.010917426
## number vmail messages    -0.0109235002      0.005909093         0.026212162
## total day minutes         0.0064022398      0.011715351         0.008101399
## total day calls           0.0048449139     -0.020477748         0.027980588
## total day charge          0.0064022233      0.011728479         0.008102666
## total eve minutes         0.0065605757      0.999999779        -0.008596755
## total eve calls           1.0000000000      0.006566003        -0.014922296
## total eve charge          0.0065660029      1.000000000        -0.008597833
## total night minutes      -0.0149222962     -0.008597833         1.000000000
## total night calls        -0.0000716459      0.019549814         0.018057638
## total night charge       -0.0148798647     -0.008601365         0.999999219
## total intl minutes        0.0211851239     -0.015917091        -0.010261690
## total intl calls          0.0324821312     -0.028620084        -0.004385542
## total intl charge         0.0211752297     -0.015935241        -0.010147464
## customer service calls   -0.0039216041     -0.006520952        -0.010265826
## churn                     0.0128916069      0.126478335         0.029516119
##                        total night calls total night charge total intl minutes
## account length              -0.0144008083       -0.010932673        0.029865848
## number vmail messages        0.0280944151        0.026204807        0.011899360
## total day minutes            0.0202562618        0.008081078       -0.008318503
## total day calls             -0.0154157505        0.027981301        0.011420107
## total day charge             0.0202606653        0.008082331       -0.008322514
## total eve minutes            0.0195460569       -0.008600289       -0.015910292
## total eve calls             -0.0000716459       -0.014879865        0.021185124
## total eve charge             0.0195498141       -0.008601365       -0.015917091
## total night minutes          0.0180576381        0.999999219       -0.010261690
## total night calls            1.0000000000        0.018050520       -0.002444292
## total night charge           0.0180505196        1.000000000       -0.010243788
## total intl minutes          -0.0024442920       -0.010243788        1.000000000
## total intl calls             0.0157556944       -0.004366073        0.055429270
## total intl charge           -0.0024213998       -0.010129647        0.999992813
## customer service calls      -0.0272582270       -0.010228364        0.003209505
## churn                        0.0033669412        0.029543144        0.062461065
##                        total intl calls total intl charge
## account length             0.0091822179       0.029848620
## number vmail messages      0.0136590526       0.011936858
## total day minutes          0.0357969099      -0.008252823
## total day calls           -0.0018728474       0.011535729
## total day charge           0.0357957999      -0.008256843
## total eve minutes         -0.0286240715      -0.015928408
## total eve calls            0.0324821312       0.021175230
## total eve charge          -0.0286200844      -0.015935241
## total night minutes       -0.0043855422      -0.010147464
## total night calls          0.0157556944      -0.002421400
## total night charge        -0.0043660726      -0.010129647
## total intl minutes         0.0554292700       0.999992813
## total intl calls           1.0000000000       0.055465685
## total intl charge          0.0554656854       1.000000000
## customer service calls     0.0008178319       0.003256039
## churn                     -0.0364967672       0.062517988
##                        customer service calls        churn
## account length                   0.0101223070  0.024936027
## number vmail messages           -0.0380357010 -0.099551786
## total day minutes               -0.0077470505  0.219118543
## total day calls                 -0.0098893591  0.040960889
## total day charge                -0.0077498563  0.219113840
## total eve minutes               -0.0065124350  0.126495726
## total eve calls                 -0.0039216041  0.012891607
## total eve charge                -0.0065209522  0.126478335
## total night minutes             -0.0102658261  0.029516119
## total night calls               -0.0272582270  0.003366941
## total night charge              -0.0102283643  0.029543144
## total intl minutes               0.0032095050  0.062461065
## total intl calls                 0.0008178319 -0.036496767
## total intl charge                0.0032560386  0.062517988
## customer service calls           1.0000000000  0.179089086
## churn                            0.1790890862  1.000000000
```

```
# drop uncorrelated categories - account length, day calls, eve calls, night calls
# drop one of the highly correlated pairs - day charge, eve charge, night charge
train <- train[,c(-1,-6,-7,-9,-10,-12,-13,-16)]

# final check of structure - 10 predictors, 1 response
str(train)
```

```
## tibble [2,324 x 10] (S3: tbl_df/tbl/data.frame)
##  $ international plan    : Factor w/ 2 levels "no","yes": 1 1 2 2 2 1 2 1 1 1 ...
##  $ voice mail plan       : Factor w/ 2 levels "no","yes": 2 2 1 1 1 2 2 1 1 1 ...
##  $ number vmail messages : num [1:2324] 25 26 0 0 0 24 37 0 0 0 ...
##  $ total day minutes     : num [1:2324] 265 162 299 167 223 ...
##  $ total eve minutes     : num [1:2324] 197.4 195.5 61.9 148.3 220.6 ...
##  $ total night minutes   : num [1:2324] 245 254 197 187 204 ...
##  $ total intl minutes    : num [1:2324] 10 13.7 6.6 10.1 6.3 7.5 11.2 12.7 9.1 11.2 ...
##  $ total intl calls      : num [1:2324] 3 3 7 3 6 7 5 6 5 2 ...
##  $ customer service calls: num [1:2324] 1 1 2 3 0 3 0 4 0 1 ...
##  $ churn                 : logi [1:2324] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
# plot the remaining quantitative variables' pairwise correlation
ggpairs(train[,3:10], upper = list(continuous = GGally::wrap(ggally_cor,
                                                   size = 3,
                                                   color ="black",
                                                   stars = F)))
```



4 Model development and performance evaluation

Logistic Regression

We fit three logistic regression models with different predictors: all 9 predictors, top 6 most correlated predictors, and top 3 most correlated predictors. The first model uses all 9 predictors and all 9 appear to be good predictors since the p-values of their coefficients are all below the 0.05 alpha cutoff. Using a prediction threshold of 0.5, this model achieves a precision 0.56, recall 0.19, and F1 score 0.29. We will consider these numbers as our baseline for the remainder of the study. Looking at the confusion matrix, we noticed that this model is only predicting a total of 58 `TRUE` responses. This is a very small number for our test set of 1009 observations. Since the data set is unbalanced, we adjusted the prediction threshold such that we are predicting more `TRUE` responses. Empirically we found 0.35 to have the best results, now with 93 predicted `TRUE` responses. This threshold achieves better precision, recall, and F1 score compared to the baseline. We did not use accuracy as a metric to evaluate model performance because the data set is unbalanced. But sometimes using all the predictors is not the best model, and we decided to investigate this. For the second model, we dropped the predictors with the least significant p-values: `number_vmail_messages`, `total_night_minutes`, and `total_intl_calls`. It turns out this second model performed worse. And we investigated this idea further with the third model using only the 3 predictors with the strongest effects: `international_plan`, `voice_mail_plan`, and `customer_service_calls`. This model performed the worst of the three.

```r
# model 1: use all the predictors
lm1 <- glm(churn ~ ., data=train, family="binomial")
summary(lm1)
```

```
##
## Call:
## glm(formula = churn ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0513  -0.5207  -0.3423  -0.1921   3.2727
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.397106   0.618274 -13.582  < 2e-16 ***
## `international plan`yes   2.008705   0.171671  11.701  < 2e-16 ***
## `voice mail plan`yes     -2.464679   0.724420  -3.402 0.000668 ***
## `number vmail messages`   0.049470   0.022939   2.157 0.031040 *
## `total day minutes`       0.013745   0.001295  10.617  < 2e-16 ***
## `total eve minutes`       0.009110   0.001365   6.672 2.53e-11 ***
## `total night minutes`     0.003364   0.001313   2.562 0.010395 *
## `total intl minutes`      0.079475   0.023892   3.326 0.000880 ***
## `total intl calls`       -0.069604   0.028689  -2.426 0.015258 *
## `customer service calls`  0.439125   0.046742   9.395  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1941.7  on 2323  degrees of freedom
## Residual deviance: 1519.1  on 2314  degrees of freedom
## AIC: 1539.1
##
## Number of Fisher Scoring iterations: 6
```

```
lm1.prob <- predict(lm1, newdata=test, type="response")
lm1.pred <- rep(FALSE, nrow(test))
lm1.pred[lm1.prob > 0.5] <- TRUE
table(lm1.pred, test$churn)
```

```
##
## lm1.pred FALSE TRUE
##    FALSE   847  114
##    TRUE     21   27
```

```
cat("\n")
```

```
# a function to compute the accuracy, precision, recall, and F1 scores of a model
score <- function(prediction, actual) {
  accuracy <- mean(prediction == actual)
  cat("Accuracy: ", accuracy, "\n")
  precision <- sum(prediction == TRUE & prediction == actual)/sum(prediction == TRUE)
  cat("Precision: ", precision, "\n")
  recall <- sum(prediction == TRUE & prediction == actual)/sum(actual == TRUE)
  cat("Recall: ", recall, "\n")
  f1 <- 2*precision*recall/(precision+recall)
  cat("F1: ", f1, "\n")
  out <- rep(0,4)
  out[1:4] = c(accuracy, precision, recall, f1)
  return(out)
}

lm1.scores50 <- score(lm1.pred, test$churn)
```

```
## Accuracy:  0.8662042
## Precision:  0.5625
## Recall:  0.1914894
## F1:  0.2857143
```

```
# adjusting the response threshold to 0.35
lm1.pred <- rep(FALSE, nrow(test))
lm1.pred[lm1.prob > 0.35] <- TRUE
table(lm1.pred, test$churn)
```

```
##
## lm1.pred FALSE TRUE
##    FALSE   825   91
##    TRUE     43   50
```

```
cat("\n")
```

```
lm1.scores35 <- score(lm1.pred, test$churn)
```

```
## Accuracy:  0.8671952
## Precision:  0.5376344
## Recall:  0.3546099
## F1:  0.4273504
```

```
# model 2: remove the least significant predictors
lm2 <- glm(churn ~ . -`number vmail messages` -`total night minutes` -`total intl calls`, data=train, fa
#summary(lm2)
lm2.prob <- predict(lm2, newdata=test, type="response")
lm2.pred <- rep(FALSE, nrow(test))
lm2.pred[lm2.prob > 0.35] <- TRUE
table(lm2.pred, test$churn)
```

```
##
## lm2.pred FALSE TRUE
##    FALSE   821   97
##    TRUE     47   44
```

```
cat("\n")
```

```
lm2.scores35 <- score(lm2.pred, test$churn)
```

```
## Accuracy:  0.8572844
## Precision:  0.4835165
## Recall:  0.3120567
## F1:  0.3793103
```

```
# model 3: only use the most significant predictors
lm3 <- glm(churn ~ `international plan` + `voice mail plan` + `customer service calls`, data=train, fam
#summary(lm3)
lm3.prob <- predict(lm3, newdata=test, type="response")
lm3.pred <- rep(FALSE, nrow(test))
lm3.pred[lm3.prob > 0.35] <- TRUE
table(lm3.pred, test$churn)
```

```
##
## lm3.pred FALSE TRUE
##    FALSE   842  114
##    TRUE     26   27
```

```
cat("\n")
```

```
lm3.scores35 <- score(lm3.pred, test$churn)
```

```
## Accuracy:  0.8612488
## Precision:  0.509434
## Recall:  0.1914894
## F1:  0.2783505
```

LDA

The next class of model uses Linear Discriminant Analysis (LDA). This is a generative model that tries to determine the probability that a response is of class k, by computing its density function fk(x). To do this, it assumes that the density function is normal and each class has a shared covariance matrix Σ. We fit the LDA model using all the predictors since this was the best logistic regression model. We also compared the prediction thresholds at 0.5 and 0.35 (this is 1 - 0.65 from the logistic regression model and achieves the same effective threshold due to the semantics of the built-in R functions). The LDA model with the adjusted prediction threshold performed better than the 0.5 threshold. But LDA performed slightly worse than logistic regression.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
lda.fit <- lda(churn ~ ., data=train)
#lda.fit
#plot(lda.fit)

lda.prob <- predict(lda.fit, newdata=test)
lda.pred <- rep(TRUE, nrow(test))
lda.pred[lda.prob$posterior[,1] > 0.5] <- FALSE
table(lda.pred, test$churn)
```

```
##
## lda.pred FALSE TRUE
##    FALSE   837  110
##    TRUE     31   31
```

```
cat("\n")
```

```
lda.scores50 <- score(lda.pred, test$churn)
```

```
## Accuracy:  0.8602577
## Precision:  0.5
## Recall:  0.2198582
## F1:  0.3054187
```

```
# adjusting the response threshold to 0.65
lda.pred <- rep(TRUE, nrow(test))
lda.pred[lda.prob$posterior[,1] > 0.65] <- FALSE
table(lda.pred, test$churn)
```

```
##
## lda.pred FALSE TRUE
##    FALSE   815   94
##    TRUE     53   47
```

```
cat("\n")
```

```
lda.scores65 <- score(lda.pred, test$churn)
```

```
## Accuracy:  0.8543112
## Precision:  0.47
## Recall:  0.3333333
## F1:  0.3900415
```

QDA

A more complex version of LDA is called Quadratic Discriminant Analysis (QDA). This uses the same approach as LDA, except we add more flexibility by allowing the covariance matrix Σ to differ among classes. Again we used all the predictors for this model, and again we compared the prediction thresholds at 0.5 and 0.35. We found the adjusted threshold performed better than the 0.5 threshold. QDA performed much better than logistic regression and LDA. The precision is only slightly better, but the recall (and therefore F1 score) is significantly better.

```
qda.fit <- qda(churn ~ ., data=train)
#qda.fit

qda.prob <- predict(qda.fit, newdata=test)
qda.pred <- rep(TRUE, nrow(test))
qda.pred[qda.prob$posterior[,1] > 0.5] <- FALSE
table(qda.pred, test$churn)
```

```
##
## qda.pred FALSE TRUE
##    FALSE   814   62
##    TRUE     54   79
```

```
cat("\n")
```

```
qda.scores50 <- score(qda.pred, test$churn)
```

```
## Accuracy:  0.8850347
## Precision:  0.593985
## Recall:  0.5602837
## F1:  0.5766423
```

```
# adjusting the response threshold to 0.65
qda.pred <- rep(TRUE, nrow(test))
qda.pred[qda.prob$posterior[,1] > 0.65] <- FALSE
table(qda.pred, test$churn)
```

```
##
## qda.pred FALSE TRUE
##    FALSE   797   41
##    TRUE     71  100
```

```
cat("\n")
```

```
qda.scores65 <- score(qda.pred, test$churn)
```

```
## Accuracy:  0.888999
## Precision:  0.5847953
## Recall:  0.7092199
## F1:  0.6410256
```

Naive Bayes

The last model we fit is Naive Bayes (NB). This is another generative model, but it takes a different approach to simplifying the math of estimating distributional probabilities. Instead of assuming these class distributions belong to the normal family, it instead assumes that the predictors are independent of each other within each class. Again we used all the predictors for this model, but this time we did not adjust the threshold because Naive Bayes does not allow this - it simply predicts the class with the higher probability (essentially the threshold is 0.5). This model has the highest precision of all the models, but its recall (and therefore F1 score) is similar to logistic regression and LDA.

```r
library(e1071)
nb.fit <- naiveBayes(churn ~ ., data=train)
#nb.fit

nb.pred <- predict(nb.fit, newdata=test)
table(nb.pred, test$churn)
```

```
##
## nb.pred FALSE TRUE
##   FALSE   844   90
##   TRUE     24   51
```

```r
cat("\n")
```

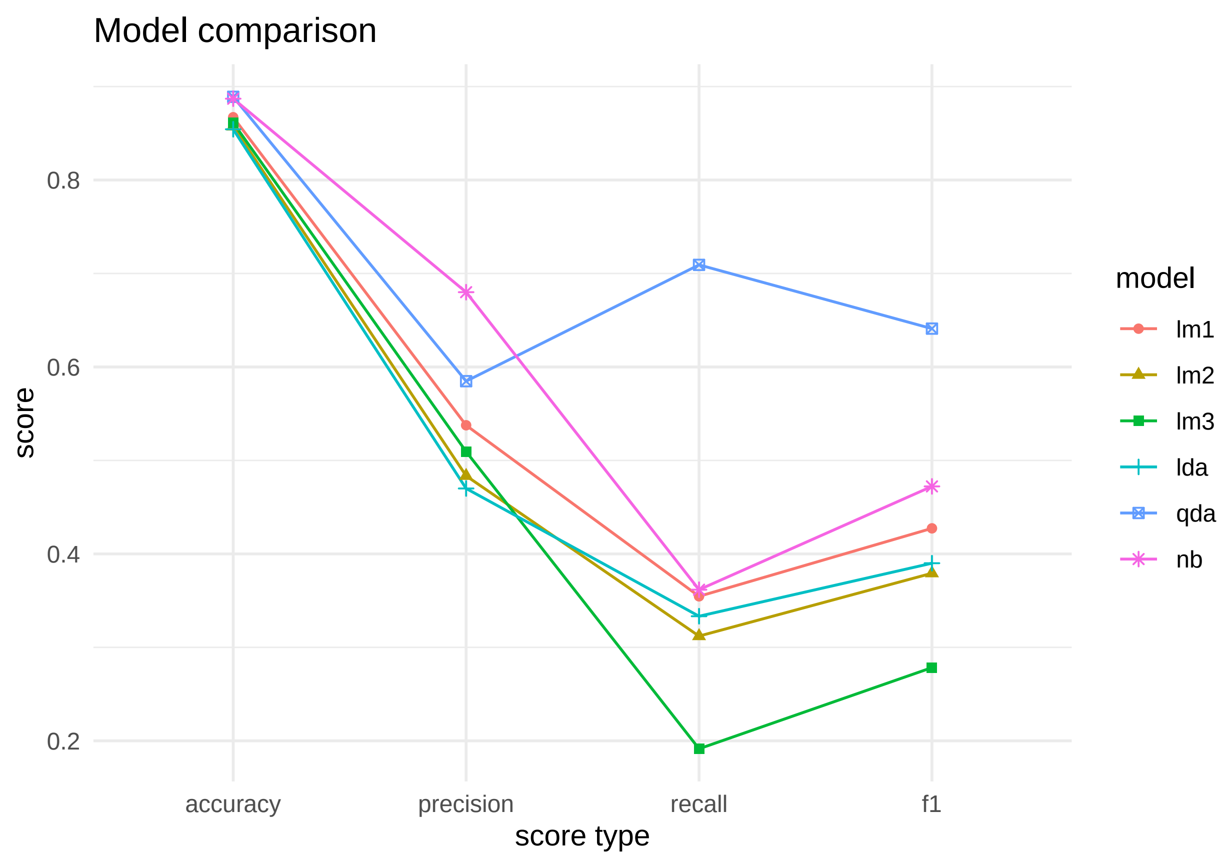
```r
nb.scores <- score(nb.pred, test$churn)
```

```
## Accuracy:  0.8870168
## Precision:  0.68
## Recall:  0.3617021
## F1:  0.4722222
```

```r
scores <- data.frame(score=c(lm1.scores35, lm2.scores35, lm3.scores35, lda.scores65, qda.scores65, nb.s
                     type=c(rep(c("accuracy","precision","recall","f1"),6)),
                     model=c(rep(c("lm1", "lm2", "lm3", "lda", "qda", "nb"),each=4)))

scores$model <- factor(scores$model, levels=c("lm1", "lm2", "lm3", "lda", "qda", "nb"))

ggplot(scores, aes(x=factor(type), y=score, group=model, color=model, shape=model)) +
  geom_line() +
  geom_point() +
  scale_x_discrete(limits=c("accuracy","precision","recall","f1")) +
  xlab("score type") +
  ggtitle("Model comparison") +
  theme_minimal()
```



5 Conclusion

The best model from our analysis is the QDA fit. The challenge with this data set is its imbalance. With so many `FALSE` responses, it is difficult to accurately predict `TRUE` responses. And the naive case of always predicting `FALSE` has a high accuracy of 86% (though it has 0% precision and recall). The QDA model predicts true 171 times and does so correctly on 100 of those. It is only missing 41 `TRUE` responses. This is much better than the naive case of always predicting `FALSE` and better than any of the other models in this analysis. It achieves 89% accuracy, 58% precision, 71% recall, and 64% F1 score. We can also determine the effect of the predictors. Having international plans, voice mail plans, fewer voicemail messages, slightly more day/evening/night/international minutes, and more customer service calls are all associated with churning. State, area code, account length, and number of calls were not related to churning.

```
# interpret output of best model (QDA)
qda.fit
```

```
## Call:
## qda(churn ~ ., data = train)
##
## Prior probabilities of groups:
##     FALSE      TRUE
## 0.8528399 0.1471601
##
## Group means:
##       `international plan`yes `voice mail plan`yes `number vmail messages`
## FALSE              0.06609485            0.2976791                8.542381
## TRUE               0.29239766            0.1549708                4.748538
##       `total day minutes` `total eve minutes` `total night minutes`
## FALSE            175.8409            198.1928              200.6092
## TRUE             209.5906            216.4474              204.8365
##       `total intl minutes` `total intl calls` `customer service calls`
## FALSE             10.14299           4.526236                 1.465691
## TRUE              10.63860           4.269006                 2.125731
```

```
cat("\n")
```

```
table(qda.pred, test$churn)
```

```
## 
## qda.pred FALSE TRUE
##    FALSE   797   41
##    TRUE     71  100
```

```
cat("\n")
```

```
results <- score(qda.pred, test$churn)
```

```
## Accuracy:  0.888999
## Precision:  0.5847953
## Recall:  0.7092199
## F1:  0.6410256
```