Churn Management

Conor Dowd, Esha Banerjee, Jingying Bi, Kanyao Han, Minjia Zhu

January 27, 2018

```
library(bit64)
library(data.table)
library(ggplot2)
library(broom)
library(knitr)
library(Hmisc)
library(tidyverse)
set.seed(4242)
```

Data

```
load("/classes/37105/main/Assignment-5/Cell2Cell.RData")
# Remove missing values
cell2cell_DT = cell2cell_DT[complete.cases(cell2cell_DT)]
cell2cell_DT = cell2cell_DT[!(cell2cell_DT$income==0 & cell2cell_DT$setprc==0),]
```

Model estimation

Estimate a logit model to predict the conditional churn probability.

```
train <- cell2cell_DT[cell2cell_DT$calibrat == 1, ]
validation <- cell2cell_DT[cell2cell_DT$calibrat == 0, ]

model_base = "churn ~ eqpdays + retcall + refurb + creditaa + months + setprcm +
              changem + changer + mailres + uniqsubs + actvsubs + recchrge +revenue +
              dropvce + credita + webcap + age1 + phones + children + roam +
              newcelly + incalls + prizmub + threeway + blckvce + setprc + incmiss +
              marryun + income + prizmtwn + custcare + unansvce + mcycle + occhmkr +
              peakvce + outcalls"
fit <- glm(formula(model_base), family = binomial(), data = train)

results_DT = as.data.table(tidy(fit))
kable(results_DT, digits = 5, format = 'pandoc')
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.22417	0.08667	2.58663	0.00969
eqpdays	0.00146	0.00007	19.68796	0.00000
retcall	0.71418	0.06025	11.85401	0.00000
refurb	0.25206	0.03258	7.73575	0.00000
creditaa	-0.40732	0.03821	-10.65975	0.00000
months	-0.02096	0.00193	-10.88631	0.00000
setprcm	-0.13767	0.04202	-3.27638	0.00105
changem	-0.00054	0.00006	-9.28211	0.00000
changer	0.00264	0.00040	6.65561	0.00000
mailres	-0.12640	0.02658	-4.75454	0.00000

term	estimate	std.error	statistic	p.value
uniqsubs	0.19390	0.02160	8.97565	0.00000
actvsubs	-0.21187	0.02978	-7.11507	0.00000
recchrge	-0.00524	0.00064	-8.15207	0.00000
revenue	0.00281	0.00041	6.91443	0.00000
dropvce	0.00682	0.00171	3.99111	0.00007
credita	-0.19438	0.03862	-5.03267	0.00000
webcap	-0.17378	0.04002	-4.34209	0.00001
age1	-0.00341	0.00078	-4.37977	0.00001
phones	0.04947	0.01278	3.87034	0.00011
children	0.10587	0.02680	3.95066	0.00008
roam	0.00616	0.00211	2.91585	0.00355
newcelly	-0.08554	0.02833	-3.01963	0.00253
incalls	-0.00377	0.00104	-3.60903	0.00031
prizmub	-0.04025	0.02557	-1.57375	0.11555
threeway	-0.03511	0.01201	-2.92234	0.00347
blckvce	0.00182	0.00116	1.56873	0.11671
setprc	0.00061	0.00028	2.15055	0.03151
incmiss	-0.20147	0.05759	-3.49807	0.00047
marryun	0.07560	0.03086	2.45025	0.01428
income	-0.01136	0.00576	-1.97127	0.04869
prizmtwn	0.06286	0.03300	1.90514	0.05676
custcare	-0.00848	0.00282	-3.00633	0.00264
unansvce	0.00059	0.00044	1.34668	0.17808
mcycle	0.13755	0.08796	1.56384	0.11785
occhmkr	0.28921	0.18938	1.52717	0.12672
peakvce	-0.00059	0.00021	-2.85236	0.00434
outcalls	0.00022	0.00058	0.37243	0.70958

Prediction: Accounting for oversampling

```
#################### Training ####################
#Check the size of traning sample
sample_size = nrow(train)+1

#Create a training sample with one half of observations randomly chosen from the original data
#with response Y = 1, and the other half randomly chosen from the original data with response Y = 0.
train_offset <- rbind(cell2cell_DT[churn==1][sample(.N, sample_size/2)],
                      cell2cell_DT[churn==0][sample(.N, sample_size/2)])



p_t_bar = mean(train_offset$churn)
p_v_bar = mean(validation$churn)
offset = (log(p_t_bar)-log(1-p_t_bar))-(log(p_v_bar)-log(1-p_v_bar)) # offset value
train_offset = train_offset[,offset_var := rep(offset,times=sample_size)] # offset var

#re-estimate the logistic regression.
model_offset = paste(model_base, "+ offset(offset_var)")
fit_offset <- glm(formula(model_offset), family = binomial(), data = train_offset)

results_DT_offset = as.data.table(tidy(fit_offset))
kable(results_DT_offset, digits = 5, format = 'pandoc')
```

term	estimate	std.error	statistic	p.value
(Intercept)	-3.70375	0.08655	-42.79508	0.00000
eqpdays	0.00146	0.00007	19.74674	0.00000
retcall	0.69113	0.06003	11.51350	0.00000
refurb	0.30143	0.03272	9.21259	0.00000
creditaa	-0.44032	0.03800	-11.58715	0.00000
months	-0.02224	0.00191	-11.62920	0.00000
setprcm	-0.13243	0.04174	-3.17289	0.00151
changem	-0.00060	0.00006	-10.25183	0.00000
changer	0.00309	0.00041	7.56729	0.00000
mailres	-0.11010	0.02659	-4.14000	0.00003
uniqsubs	0.20707	0.02182	9.49134	0.00000
actvsubs	-0.21734	0.03005	-7.23265	0.00000
recchrge	-0.00583	0.00065	-8.94322	0.00000
revenue	0.00372	0.00042	8.75175	0.00000
dropvce	0.00617	0.00169	3.65887	0.00025
credita	-0.18193	0.03863	-4.70948	0.00000
webcap	-0.12349	0.03984	-3.09956	0.00194
age1	-0.00415	0.00078	-5.30777	0.00000
phones	0.04471	0.01269	3.52390	0.00043
children	0.13051	0.02683	4.86400	0.00000
roam	0.00106	0.00161	0.66040	0.50900
newcelly	-0.07779	0.02837	-2.74195	0.00611
incalls	-0.00294	0.00105	-2.80829	0.00498
prizmub	-0.06374	0.02561	-2.48828	0.01284
threeway	-0.03883	0.01298	-2.99131	0.00278
blckvce	0.00218	0.00118	1.85186	0.06405
setprc	0.00035	0.00028	1.22307	0.22130
incmiss	-0.13180	0.05770	-2.28408	0.02237

term	estimate	std.error	statistic	p.value
marryun	0.04622	0.03084	1.49901	0.13387
income	-0.00582	0.00575	-1.01277	0.31117
prizmtwn	0.02600	0.03288	0.79077	0.42908
custcare	-0.00842	0.00290	-2.90827	0.00363
unansvce	0.00026	0.00044	0.59966	0.54873
mcycle	0.13306	0.08874	1.49935	0.13378
occhmkr	0.30930	0.18941	1.63296	0.10248
peakvce	-0.00079	0.00021	-3.84160	0.00012
outcalls	0.00036	0.00058	0.62226	0.53377

Where you place `offset()` on the right-hand side of the formula is irrelevant.

Compare the average predicted response with the average observed response rate in the validation sample.

```
####################  Validation newdata ####################
# offset var is validation sample is 0
validation = validation[,offset_var := rep(0,times=nrow(validation))]
predicted_validation <-predict(fit_offset, newdata = validation, type="response")

# average predicted response in the validation sample
print('Predicted')
```

```
## [1] "Predicted"
```

```
mean(predicted_validation)
```

```
## [1] 0.01919376
```

```
# average observed response rate in the validation sample
print('Observed')
```

```
## [1] "Observed"
```

```
mean(validation$churn)
```

```
## [1] 0.0190397
```

The average predicted response rate is very similar to the observed one. The former is slightly higher than the latter.

Predictive power: Lift

```
liftTable <- function(score, observed, seg_num){
  DT = data.table(score = score,
                  observed = observed)
  # create the score_group
  DT[, group := cut_number(score, n = seg_num)]
  DT[, score_group := as.integer(group)]

  avg_response = mean(observed)

  DT_lift = DT[,list(avg_score=mean(score),
                     avg_obs=mean(observed),
                     lift=100*mean(observed)/avg_response,
                     lower=binconf(sum(observed),.N,0.05)[2],
                     upper=binconf(sum(observed),.N,0.05)[3]),
               by=score_group]

  return(DT_lift[order(DT_lift$score_group)])
}
```

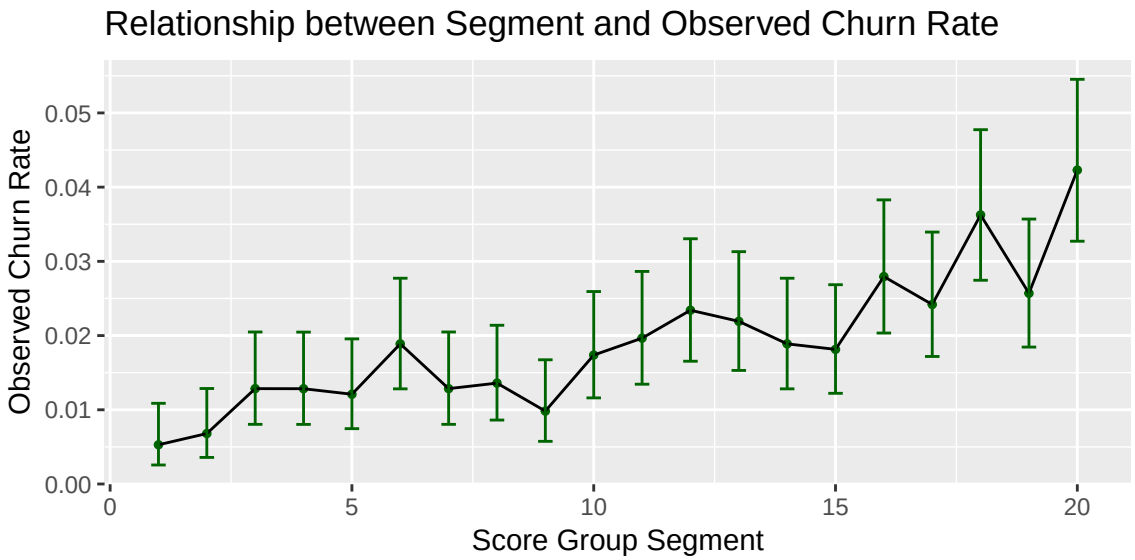
Calculate a lift table for 20 segments. Inspect the lift table.

```
# for 20 segments
lift_table_20 = liftTable(predicted_validation, validation$churn, 20)
kable(lift_table_20, format = 'pandoc')
```

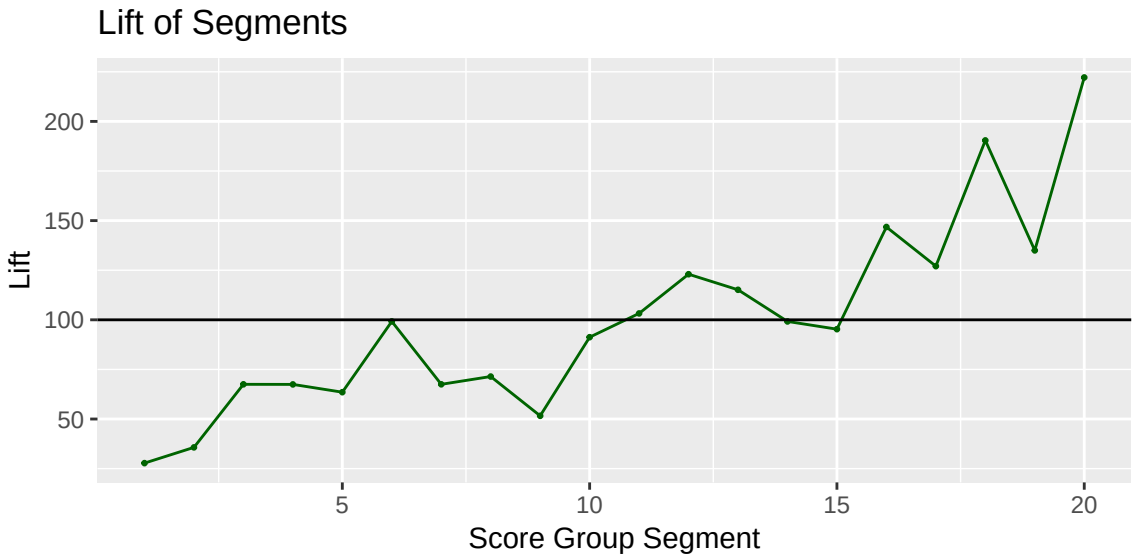
score_group	avg_score	avg_obs	lift	lower	upper
1	0.0070611	0.0052870	27.76834	0.0025634	0.0108731
2	0.0093392	0.0067976	35.70215	0.0035803	0.0128685
3	0.0107335	0.0128496	67.48836	0.0080380	0.0204820
4	0.0119125	0.0128399	67.43739	0.0080319	0.0204666
5	0.0129693	0.0120937	63.51846	0.0074577	0.0195549
6	0.0138857	0.0188822	99.17263	0.0128221	0.0277260
7	0.0147288	0.0128496	67.48836	0.0080380	0.0204820
8	0.0155159	0.0135952	71.40429	0.0086167	0.0213880
9	0.0163059	0.0098262	51.60875	0.0057514	0.0167392
10	0.0171031	0.0173716	91.23882	0.0116032	0.0259325
11	0.0179216	0.0196523	103.21750	0.0134461	0.0286399
12	0.0188051	0.0234139	122.97406	0.0165433	0.0330420
13	0.0197628	0.0219199	115.12721	0.0153048	0.0313032
14	0.0208241	0.0188822	99.17263	0.0128221	0.0277260
15	0.0220380	0.0181406	95.27769	0.0122203	0.0268510
16	0.0235101	0.0279456	146.77549	0.0203417	0.0382808
17	0.0252914	0.0241875	127.03692	0.0171846	0.0339454
18	0.0277361	0.0362538	190.41145	0.0274524	0.0477384
19	0.0315960	0.0256992	134.97672	0.0184481	0.0356966
20	0.0468338	0.0422961	222.14669	0.0327142	0.0545262

```
ggplot(lift_table_20, aes(score_group, avg_obs)) +
  geom_point(size = 1, color = "dark green") +
  geom_line(size = 0.5) +
```

```
  geom_errorbar(aes(ymin=lower, ymax=upper), width = 0.3, color = 'dark green') +
  labs(title = "Relationship between Segment and Observed Churn Rate",
       x = 'Score Group Segment',
       y = 'Observed Churn Rate')
```




```
ggplot(lift_table_20, aes(score_group, lift)) +
  geom_point(size = 0.5, color = "dark green") +
  geom_line(color = 'dark green', size = 0.5) +
  geom_hline(yintercept = 100)+
  labs(title = "Lift of Segments",
       x = 'Score Group Segment',
       y = 'Lift')
```



Generally, the observed churn rate will increase with the increase of group score.

Why do customers churn? — Effect sizes

```
# add a marginal_effect column
p = mean(predicted_validation)
results_DT_offset[, marginal_effect:= p*(1-p)*results_DT_offset$estimate]
```

```
# Calculate the standard deviation for all variables in the data.
sd_= cell2cell_DT[, lapply(.SD, sd)]
sd_DT = data.table(term    = names(sd_),
                   std_dev = c(t(sd_)))

# Merge two table
DT_effect = merge(results_DT_offset, sd_DT, by='term')
kable(DT_effect, format = 'pandoc')
```

term	estimate	std.error	statistic	p.value	marginal_effect	std_dev
actvsubs	-0.2173426	0.0300502	-7.2326489	0.0000000	-0.0040916	0.6329761
age1	-0.0041480	0.0007815	-5.3077656	0.0000001	-0.0000781	19.9173731
blckvce	0.0021843	0.0011795	1.8518645	0.0640453	0.0000411	10.7175272
changem	-0.0006004	0.0000586	-10.2518338	0.0000000	-0.0000113	252.5190606
changer	0.0030940	0.0004089	7.5672935	0.0000000	0.0000582	39.1876658
children	0.1305122	0.0268323	4.8640016	0.0000012	0.0024569	0.4509585
credita	-0.1819279	0.0386301	-4.7094814	0.0000025	-0.0034249	0.3022842
creditaa	-0.4403222	0.0380009	-11.5871530	0.0000000	-0.0082892	0.3269604
custcare	-0.0084243	0.0028967	-2.9082732	0.0036343	-0.0001586	5.1644303
dropvce	0.0061695	0.0016862	3.6588715	0.0002533	0.0001161	9.0389626
eqpdays	0.0014617	0.0000740	19.7467414	0.0000000	0.0000275	258.9958807
incalls	-0.0029381	0.0010462	-2.8082925	0.0049805	-0.0000553	16.8074382
incmiss	-0.1318027	0.0577049	-2.2840817	0.0223667	-0.0024812	0.3235196
income	-0.0058229	0.0057495	-1.0127698	0.3111701	-0.0001096	2.7752960
mailres	-0.1101023	0.0265948	-4.1399971	0.0000347	-0.0020727	0.4967211
marryun	0.0462226	0.0308353	1.4990133	0.1338702	0.0008702	0.4498834
mcycle	0.1330562	0.0887423	1.4993543	0.1337818	0.0025048	0.1247640
months	-0.0222425	0.0019126	-11.6292022	0.0000000	-0.0004187	9.9099692
newcelly	-0.0777862	0.0283689	-2.7419512	0.0061075	-0.0014644	0.3981662
occhmkr	0.3092999	0.1894106	1.6329603	0.1024773	0.0058227	0.0605459
outcalls	0.0003593	0.0005774	0.6222635	0.5337686	0.0000068	35.5819822
peakvce	-0.0007907	0.0002058	-3.8415966	0.0001222	-0.0000149	107.4148812
phones	0.0447126	0.0126884	3.5239049	0.0004252	0.0008417	1.3943050
prizmtwn	0.0260023	0.0328824	0.7907682	0.4290793	0.0004895	0.3605817
prizmub	-0.0637366	0.0256147	-2.4882797	0.0128363	-0.0011999	0.4712847
recchrge	-0.0058252	0.0006514	-8.9432161	0.0000000	-0.0001097	24.1953646
refurb	0.3014287	0.0327192	9.2125920	0.0000000	0.0056745	0.3616340
retcall	0.6911253	0.0600274	11.5134984	0.0000000	0.0130107	0.1819463
revenue	0.0037161	0.0004246	8.7517486	0.0000000	0.0000700	44.8846109
roam	0.0010611	0.0016067	0.6604031	0.5089952	0.0000200	9.3717699
setprc	0.0003485	0.0002849	1.2230713	0.2213028	0.0000066	59.3236634
setprcm	-0.1324283	0.0417374	-3.1728909	0.0015093	-0.0024930	0.5000042
threeway	-0.0388283	0.0129803	-2.9913126	0.0027778	-0.0007310	1.1821937
unansvce	0.0002610	0.0004352	0.5996639	0.5487302	0.0000049	39.2722091
uniqsubs	0.2070698	0.0218167	9.4913396	0.0000000	0.0038982	0.8613097
webcap	-0.1234919	0.0398417	-3.0995608	0.0019381	-0.0023248	0.2972390

Bonus: If you want to be extra-elegant, you can think of creating a function that detects if a column is a dummy variable that only contains 0/1 values. Hint: Such a function would check if the column contains exactly two unique values, and if 0 and 1 are contained among those unique values (the `all` function may come in handy for this task).

```
is.binary <- function(v) {
  x <- unique(v)
  length(x) - sum(is.na(x)) == 2L
}
```

```
binary_ = apply(cell2cell_DT,2,is.binary)
binary_DT = data.table(term  = names(binary_),
                   is_binary = c(t(binary_)))
head(binary_DT, 10)
```

```
##         term is_binary
##  1: customer     FALSE
##  2: calibrat      TRUE
##  3:    churn      TRUE
##  4:  revenue     FALSE
##  5:      mou     FALSE
##  6: recchrge     FALSE
##  7: directas     FALSE
##  8:  overage     FALSE
##  9:     roam     FALSE
## 10:  changem     FALSE
```

```
# Merge binary table
DT_effect = merge(DT_effect, binary_DT, by='term')
DT_effect$one_sd_effect = DT_effect$estimate/DT_effect$std_dev
head(DT_effect, 10)
```

```
##         term      estimate    std.error  statistic      p.value
##  1: actvsubs -0.2173425885 3.005021e-02  -7.232649 4.736630e-13
##  2:     age1 -0.0041480073 7.814978e-04  -5.307766 1.109773e-07
##  3:  blckvce  0.0021843409 1.179536e-03   1.851865 6.404528e-02
##  4:  changem -0.0006004287 5.856794e-05 -10.251834 1.161193e-24
##  5:  changer  0.0030939570 4.088591e-04   7.567294 3.810802e-14
##  6: children  0.1305121954 2.683227e-02   4.864002 1.150360e-06
##  7:  credita -0.1819279024 3.863014e-02  -4.709481 2.483479e-06
##  8: creditaa -0.4403221919 3.800090e-02 -11.587153 4.787834e-31
##  9: custcare -0.0084242635 2.896655e-03  -2.908273 3.634307e-03
## 10:  dropvce  0.0061695311 1.686184e-03   3.658872 2.533283e-04
##     marginal_effect     std_dev is_binary one_sd_effect
##  1:   -4.091552e-03   0.6329761     FALSE -3.433662e-01
##  2:   -7.808772e-05  19.9173731     FALSE -2.082608e-04
##  3:    4.112100e-05  10.7175272     FALSE  2.038102e-04
##  4:   -1.130329e-05 252.5190606     FALSE -2.377756e-06
##  5:    5.824485e-05  39.1876658     FALSE  7.895232e-05
##  6:    2.456939e-03   0.4509585      TRUE  2.894106e-01
##  7:   -3.424858e-03   0.3022842      TRUE -6.018440e-01
##  8:   -8.289222e-03   0.3269604      TRUE -1.346714e+00
##  9:   -1.585898e-04   5.1644303     FALSE -1.631209e-03
## 10:    1.161436e-04   9.0389626     FALSE  6.825486e-04
```

Economics of churn management

```
# Create a function for calculating the economic value

# LTV calculation function, for baseline model, incentive = alpha
LTV_fun <- function(alpha, profit, year, incentive, r){
  LTV_0 <- profit
  LTV_1 <- LTV_0 + ((incentive * profit) / ((1 + r)))
  for (yr in (year - 2)){
    LTV <- LTV_1 + (((incentive * (alpha ^ yr)) * profit) / ((1 + r) ^ (yr + 1)))
  }
  return(LTV)
}

# economic value function

economic <- function(seg_num, gamma, margin, cost, year, r){
  DT = data.table(score = predicted_validation,
                  revenue = validation$revenue)

  # create the score_group
  DT[, group := cut_number(score, n = seg_num)]
  DT[, score_group := as.integer(group)]


  DT_economic = DT[,list(avg_score=mean(score * 12),
                         avg_revenue=mean(revenue * 12)),
                   by=score_group]
  DT_economic <- DT_economic %>%
    mutate(incentive = 1 - ((1- gamma) * avg_score),
           baseline = 1 - avg_score,
           profit = avg_revenue * margin,
           LTV_base = LTV_fun(baseline, profit, year, baseline, r),
           LTV_incentive = LTV_fun(baseline, profit, year, incentive, r),
           incentive_value = LTV_incentive - LTV_base - cost,
           ROI = incentive_value / cost) %>%
    select(score_group, avg_revenue, LTV_base,
           LTV_incentive, incentive_value, ROI) %>%
    as.data.table()


  return(DT_economic[order(score_group)])
}
```

Since we don't have the information about the margin, we assume that the margin is 40% (average margin in this industrial field).

```
value_100_0.25 <- economic(seg_num = 10, gamma = 0.25, margin = 0.4, cost = 100, year = 6, r = 0.1)
value_100_0.5 <- economic(seg_num = 10, gamma = 0.5, margin = 0.4,  cost = 100, year = 6, r = 0.1)
value_175_0.25 <- economic(seg_num = 10, gamma = 0.25, margin = 0.4,  cost = 175, year = 6, r = 0.1)
value_175_0.5 <- economic(seg_num = 10, gamma = 0.5, margin = 0.4, cost = 175, year = 6, r = 0.1)

value <- bind_rows("G = 0.25, C = 100" = value_100_0.25,
                   "G = 0.5,  C = 100"= value_100_0.5,
                   "G = 0.25, C = 175"= value_175_0.25,
```

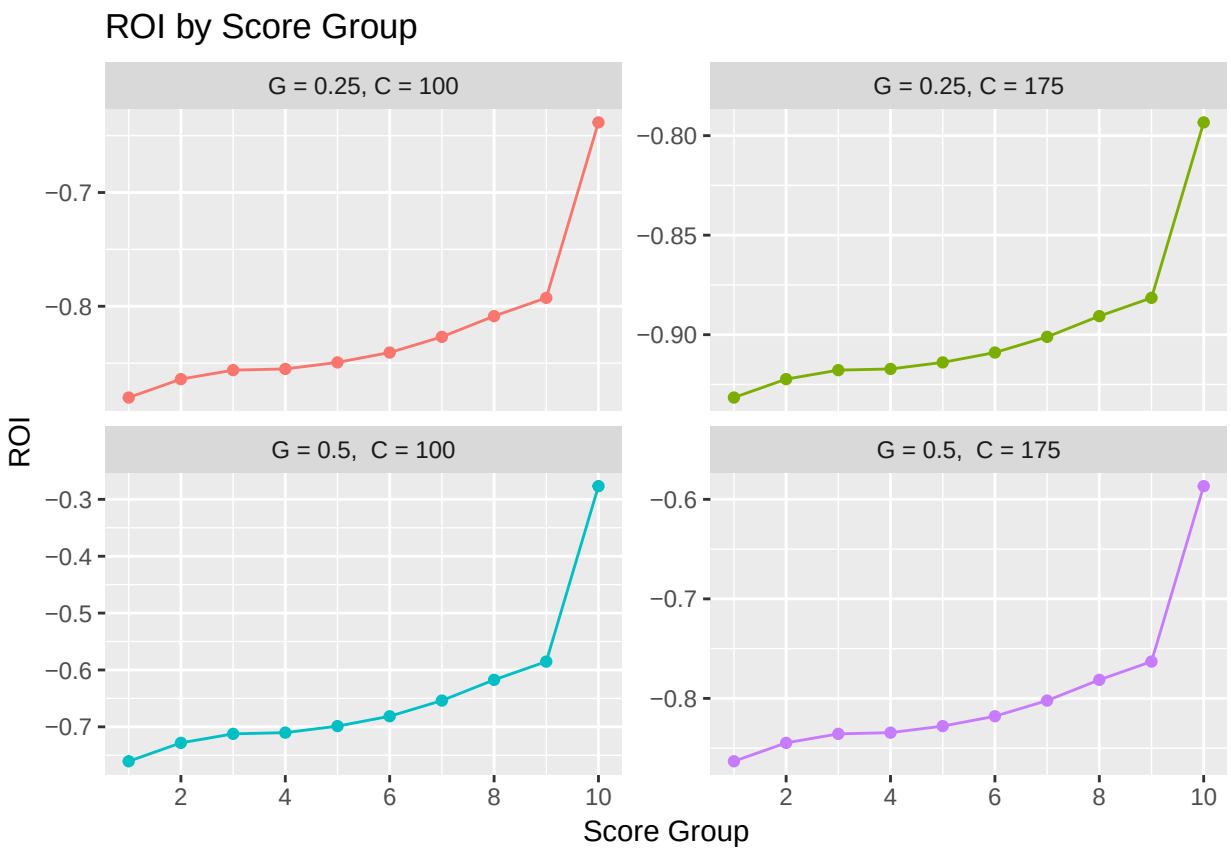
```
                    "G = 0.5,  C = 175"= value_175_0.5,
                    .id = "Gamma_Cost")

kable(value, format = 'pandoc')
```

Gamma_Cost	score_group	avg_revenue	LTV_base	LTV_incentive	incentive_value	ROI
G = 0.25, C = 100	1	922.1975	807.6783	819.6512	-88.02713	-0.8802713
G = 0.25, C = 100	2	796.8030	664.4473	678.0382	-86.40906	-0.8640906
G = 0.25, C = 100	3	733.7404	593.0207	607.4040	-85.61666	-0.8561666
G = 0.25, C = 100	4	672.1042	530.2263	544.7137	-85.51260	-0.8551260
G = 0.25, C = 100	5	646.2153	498.8116	513.8748	-84.93682	-0.8493682
G = 0.25, C = 100	6	634.9114	479.3876	495.3164	-84.07120	-0.8407120
G = 0.25, C = 100	7	638.9077	470.6262	487.9343	-82.69183	-0.8269183
G = 0.25, C = 100	8	646.8077	462.2059	481.3367	-80.86914	-0.8086914
G = 0.25, C = 100	9	624.7454	427.6617	448.3997	-79.26200	-0.7926200
G = 0.25, C = 100	10	802.2279	483.6078	519.7707	-63.83711	-0.6383711
G = 0.5, C = 100	1	922.1975	807.6783	831.6241	-76.05426	-0.7605426
G = 0.5, C = 100	2	796.8030	664.4473	691.6291	-72.81811	-0.7281811
G = 0.5, C = 100	3	733.7404	593.0207	621.7873	-71.23333	-0.7123333
G = 0.5, C = 100	4	672.1042	530.2263	559.2011	-71.02520	-0.7102520
G = 0.5, C = 100	5	646.2153	498.8116	528.9380	-69.87365	-0.6987365
G = 0.5, C = 100	6	634.9114	479.3876	511.2452	-68.14239	-0.6814239
G = 0.5, C = 100	7	638.9077	470.6262	505.2425	-65.38365	-0.6538365
G = 0.5, C = 100	8	646.8077	462.2059	500.4676	-61.73827	-0.6173827
G = 0.5, C = 100	9	624.7454	427.6617	469.1377	-58.52400	-0.5852400
G = 0.5, C = 100	10	802.2279	483.6078	555.9336	-27.67422	-0.2767422
G = 0.25, C = 175	1	922.1975	807.6783	819.6512	-163.02713	-0.9315836
G = 0.25, C = 175	2	796.8030	664.4473	678.0382	-161.40906	-0.9223375
G = 0.25, C = 175	3	733.7404	593.0207	607.4040	-160.61666	-0.9178095
G = 0.25, C = 175	4	672.1042	530.2263	544.7137	-160.51260	-0.9172149
G = 0.25, C = 175	5	646.2153	498.8116	513.8748	-159.93682	-0.9139247
G = 0.25, C = 175	6	634.9114	479.3876	495.3164	-159.07120	-0.9089783
G = 0.25, C = 175	7	638.9077	470.6262	487.9343	-157.69183	-0.9010962
G = 0.25, C = 175	8	646.8077	462.2059	481.3367	-155.86914	-0.8906808
G = 0.25, C = 175	9	624.7454	427.6617	448.3997	-154.26200	-0.8814972
G = 0.25, C = 175	10	802.2279	483.6078	519.7707	-138.83711	-0.7933549
G = 0.5, C = 175	1	922.1975	807.6783	831.6241	-151.05426	-0.8631672
G = 0.5, C = 175	2	796.8030	664.4473	691.6291	-147.81811	-0.8446749
G = 0.5, C = 175	3	733.7404	593.0207	621.7873	-146.23333	-0.8356190
G = 0.5, C = 175	4	672.1042	530.2263	559.2011	-146.02520	-0.8344297
G = 0.5, C = 175	5	646.2153	498.8116	528.9380	-144.87365	-0.8278494
G = 0.5, C = 175	6	634.9114	479.3876	511.2452	-143.14239	-0.8179565
G = 0.5, C = 175	7	638.9077	470.6262	505.2425	-140.38365	-0.8021923
G = 0.5, C = 175	8	646.8077	462.2059	500.4676	-136.73827	-0.7813616
G = 0.5, C = 175	9	624.7454	427.6617	469.1377	-133.52400	-0.7629943
G = 0.5, C = 175	10	802.2279	483.6078	555.9336	-102.67422	-0.5867098

```
ggplot(value, aes(score_group, ROI, color = Gamma_Cost)) +
  geom_line(show.legend = FALSE) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~ Gamma_Cost, scale = 'free_y') +
  scale_x_continuous(breaks = c(2, 4, 6, 8, 10)) +
  labs(title = "ROI by Score Group",
```

```
        x = 'Score Group')
```

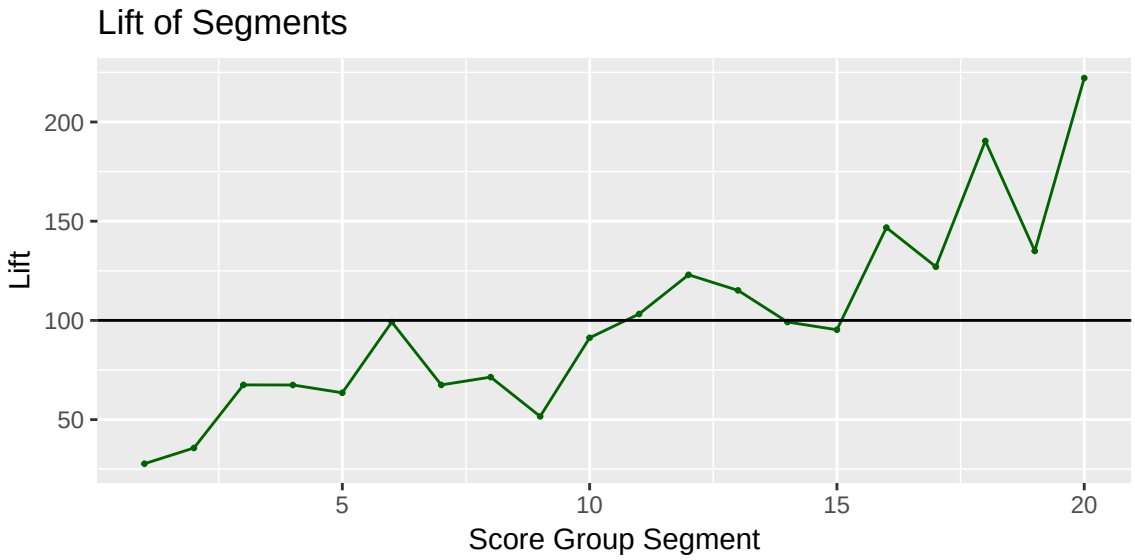


Summarize your main results

You can organize your main results along the four questions posed in the overview.

1. Customer churn at Cell2Cell is clearly somewhat predictable from customer information that C2C already has on hand. Our primary means of inferring this is by looking at how well our predictions perform. If we sort our predictions into 20 groups, and look at how our predictions compare to a base prediction of the mean-in-sample churn rate, we can see that we are able to do some differentiation. Clearly those groups rated as highest risk are more likely to churn in out-of-sample observations, while those rated low risk are less likely to churn.

```
ggplot(lift_table_20, aes(score_group, lift)) +
  geom_point(size = 0.5, color = "dark green") +
  geom_line(color = 'dark green', size = 0.5) +
  geom_hline(yintercept = 100)+
  labs(title = "Lift of Segments",
       x = 'Score Group Segment',
       y = 'Lift')
```



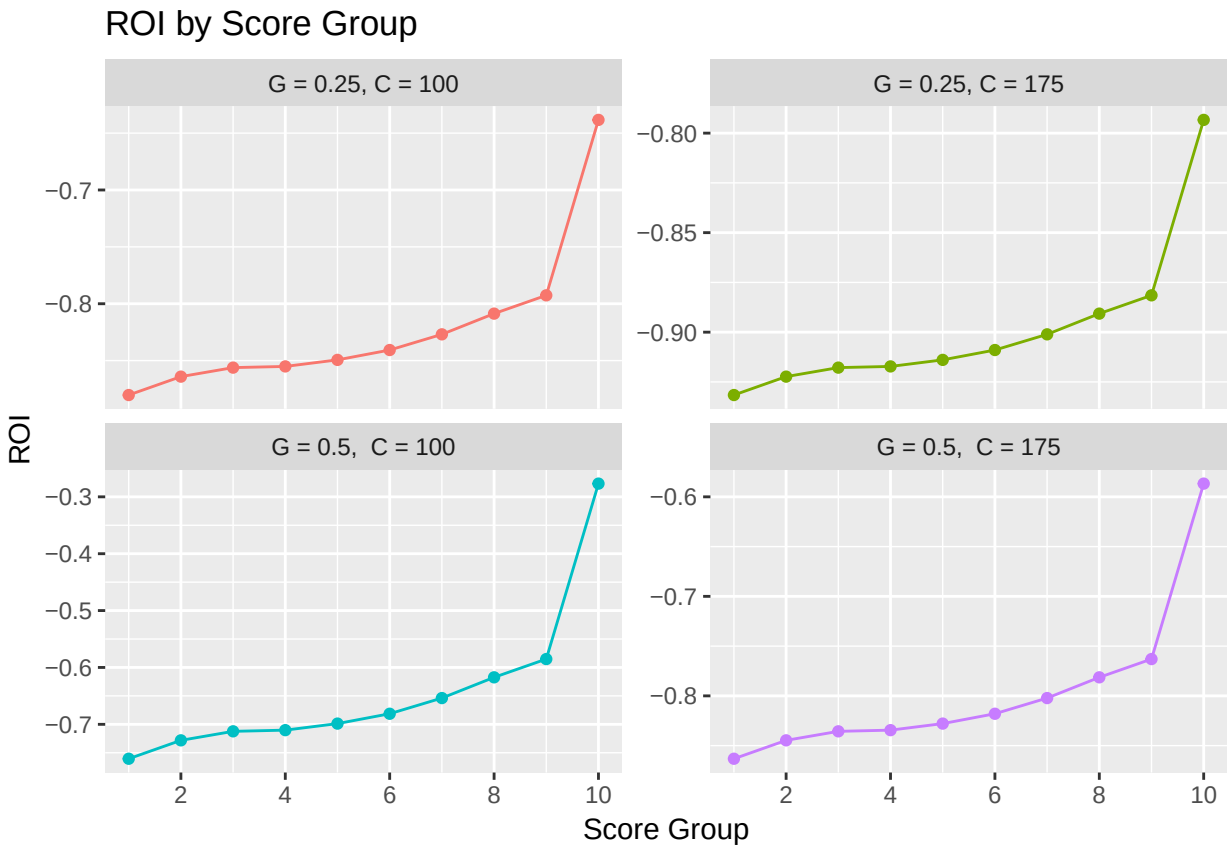
2. There are many factors influencing churn rates. Looking at all of them would be time consuming. Nevertheless, we can look at how large an effect a 1 standard deviation change in the underlying variable would have on the churn rate. In order to do this, we divide our estimates of the effect of a 1 unit change on the churn rate, by the standard deviation of each variable in question. As it turns out, there are several factors that stand out as highly important having done this. Whether or not someone is a homemaker, whether they own a motorcycle, whether they've called the customer retention group, and whether their phone is refurbished all are signs indicating a high likelihood of switching providers in the near future. Customer credit ratings of A or AA are also very predictive of churn rates, but indicate a lower likelihood of churning.

term	one_sd_effect
occhmkr	5.1085169
retcall	3.7985119
creditaa	-1.3467143
mcycle	1.0664631
refurb	0.8335187
credita	-0.6018440
webcap	-0.4154633

term	one_sd_effect
incmiss	-0.4074025
actvsubs	-0.3433662
children	0.2894106
setprcm	-0.2648544
uniqsubs	0.2404127
mailres	-0.2216583
newcelly	-0.1953612
prizmub	-0.1352401
marryun	0.1027435
prizmtwn	0.0721122
threeway	-0.0328442
phones	0.0320680
months	-0.0022445
income	-0.0020981
custcare	-0.0016312
dropvce	0.0006825
recchrge	-0.0002408
age1	-0.0002083
blckvce	0.0002038
incalls	-0.0001748
roam	0.0001132
revenue	0.0000828
changer	0.0000790
outcalls	0.0000101
peakvce	-0.0000074
unansvce	0.0000066
setprc	0.0000059
eqpdays	0.0000056
changem	-0.0000024

3. There are two sorts of incentives Cell2Cell could offer. It could sell its plans at a lower rate to persons with high credit ratings – this would change the pool of customers that Cell2Cell has, thus lowering their churn rate in the long term. Alternately, Cell2Cell could give targetted discounts to individuals who are extra likely to leave shortly, preventing that from happening. The single easiest thing to do would be to provide better offers to people who call customer retention lines. Those individuals are already in contact with Cell2Cell about a deal, and are very likely to leave. Sweetening the pot for them would help tremendously. Another group worth targetting is homemakers – this is also relatively straightforward in that it merely requires making calls to home phones during work hours. Offering deals to individuals who purchase refurbished phones or motorcycles are also possibilities.

4. Since we don't have the information about the margin, we assume that the margin is 40% (average margin in this industrial field). From these plots we can clearly see that both the scale of an incentive program, and its chance of success will not influence the genral economic value distribution and trend form among different segements (the economic value will increase with the group churn score increases). However, the specific economic values will change based on the cost and the success rate we select. Broadly speaking, we can clearly see that for groups which are unlikely to churn, the value of an incentive to the firm is much lower. Moreover, mechanically, as the size of the incentive grows and the success rate does not change, the value to the firm of the incentive falls. Therefore, it is very important to balance how much money the companies should invest in incentive and how high the success rate will be.


ROI by Score Group
G = 0.25, C = 100
G = 0.25, C = 175
G = 0.5, C = 100
G = 0.5, C = 175
ROI
Score Group
−0.7
−0.8
−0.80
−0.85
−0.90
−0.3
−0.4
−0.5
−0.6
−0.7
−0.6
−0.7
−0.8
2
4
6
8
10