Data Science in Action

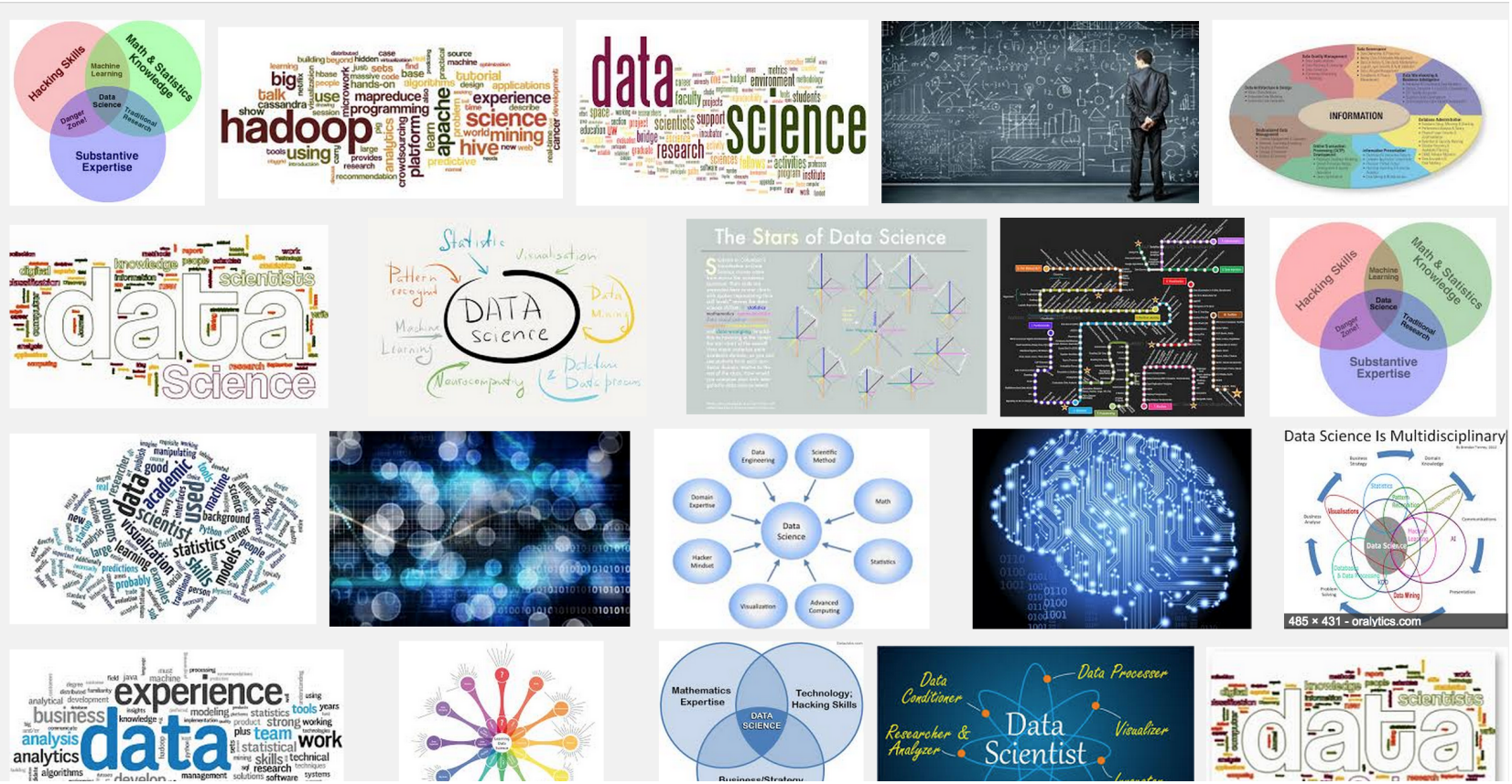
Ji Li

Data Scientist

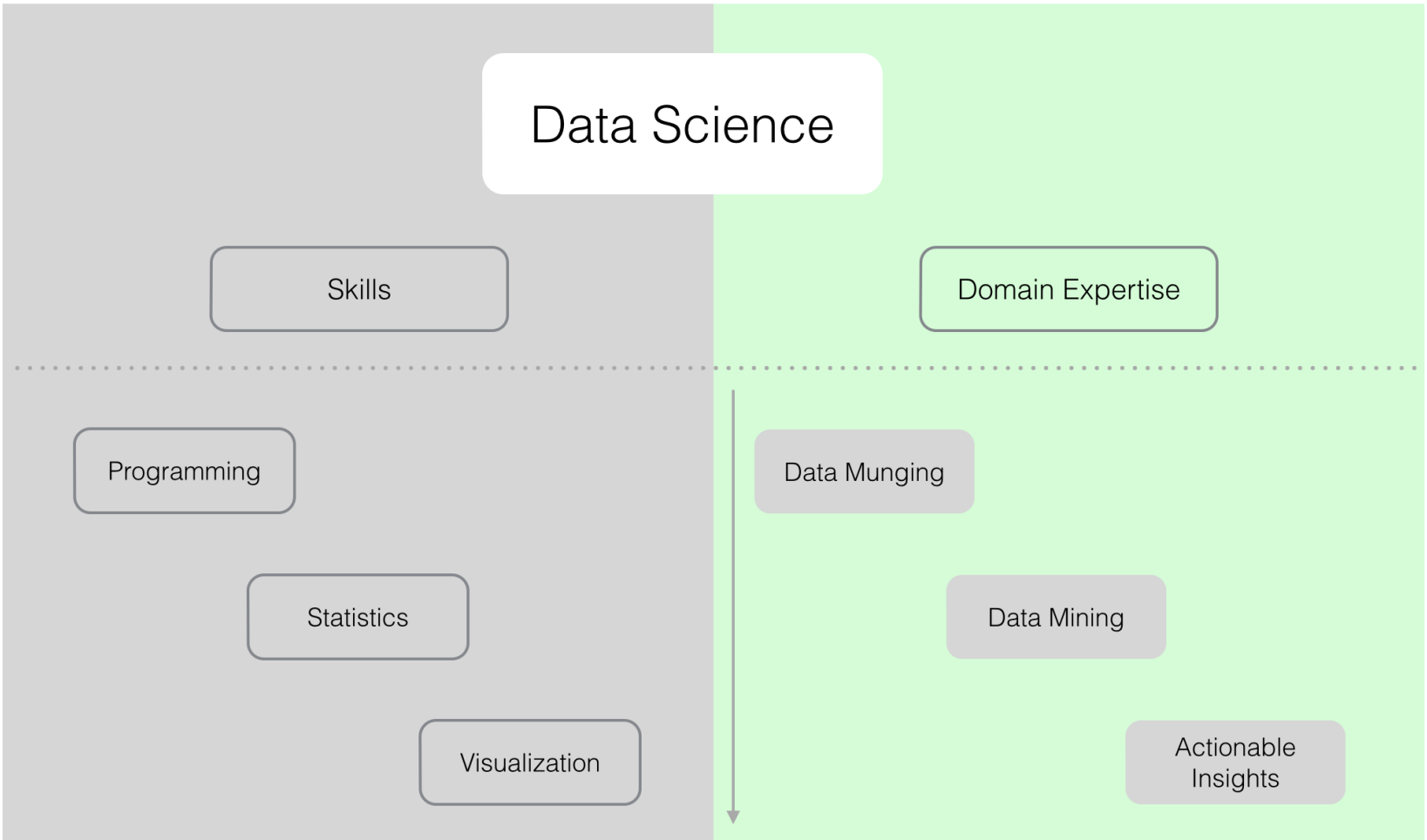
March 25, 2015

Outline

Data science on Google search

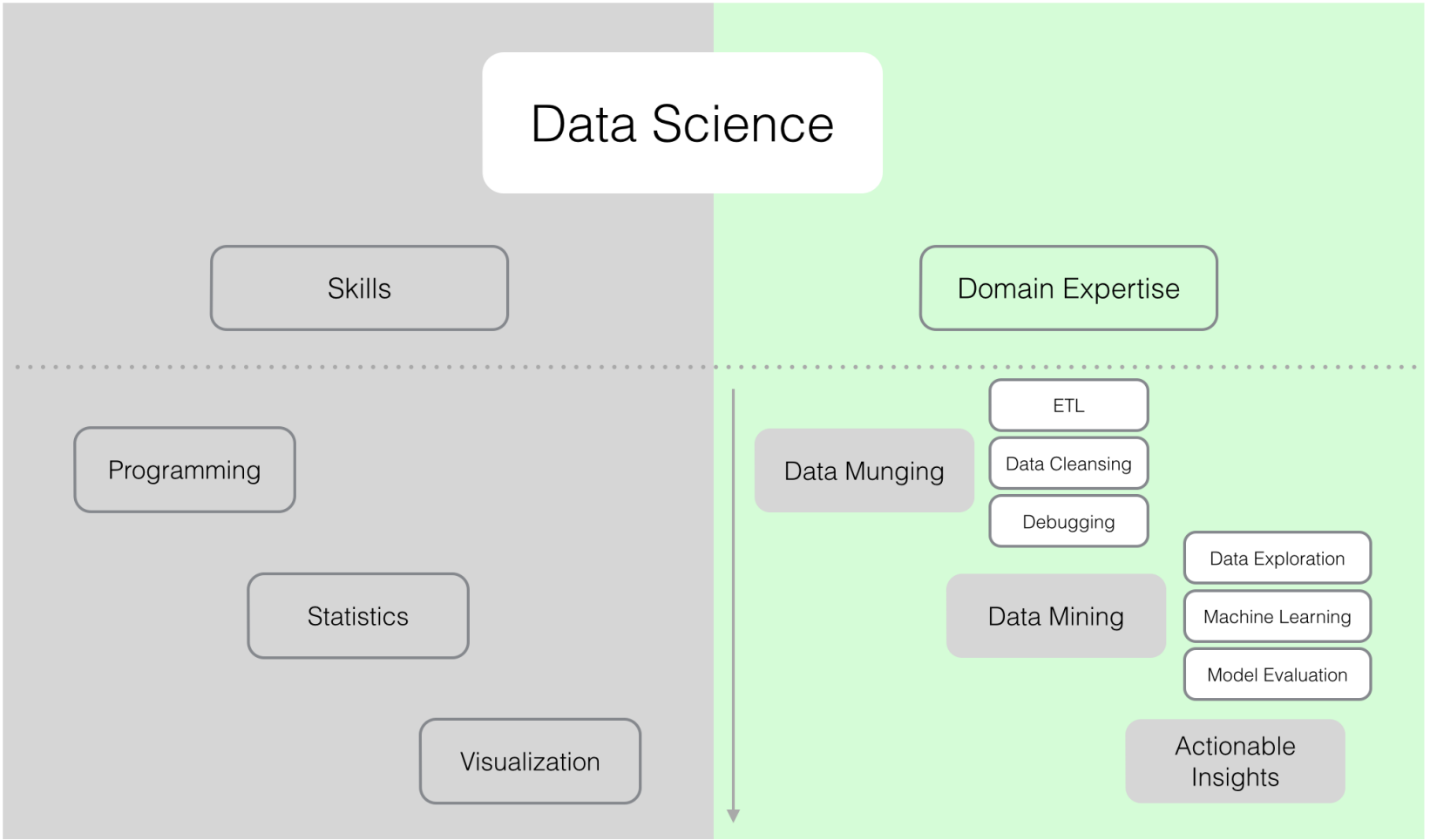


Data science from my point of view

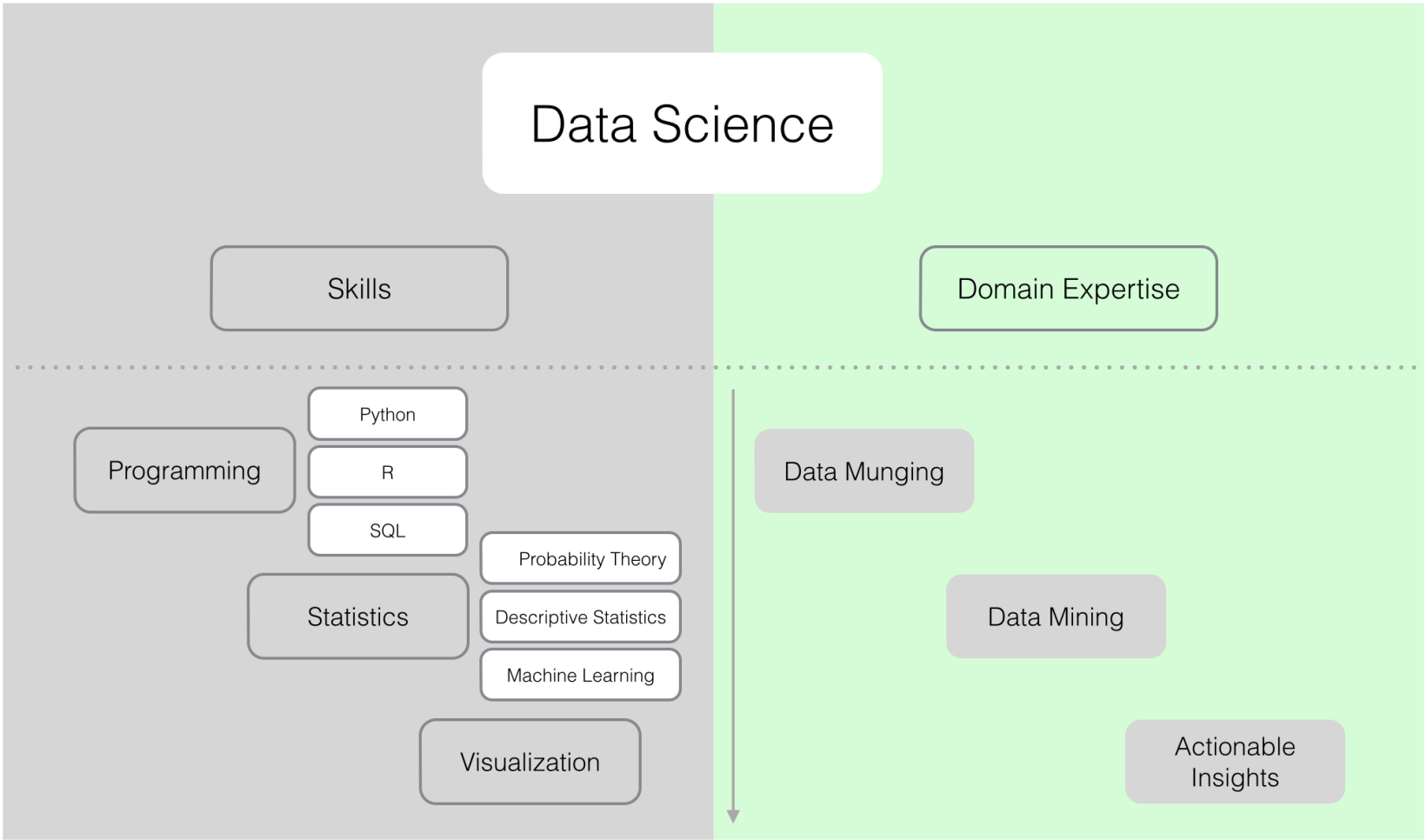


Outline

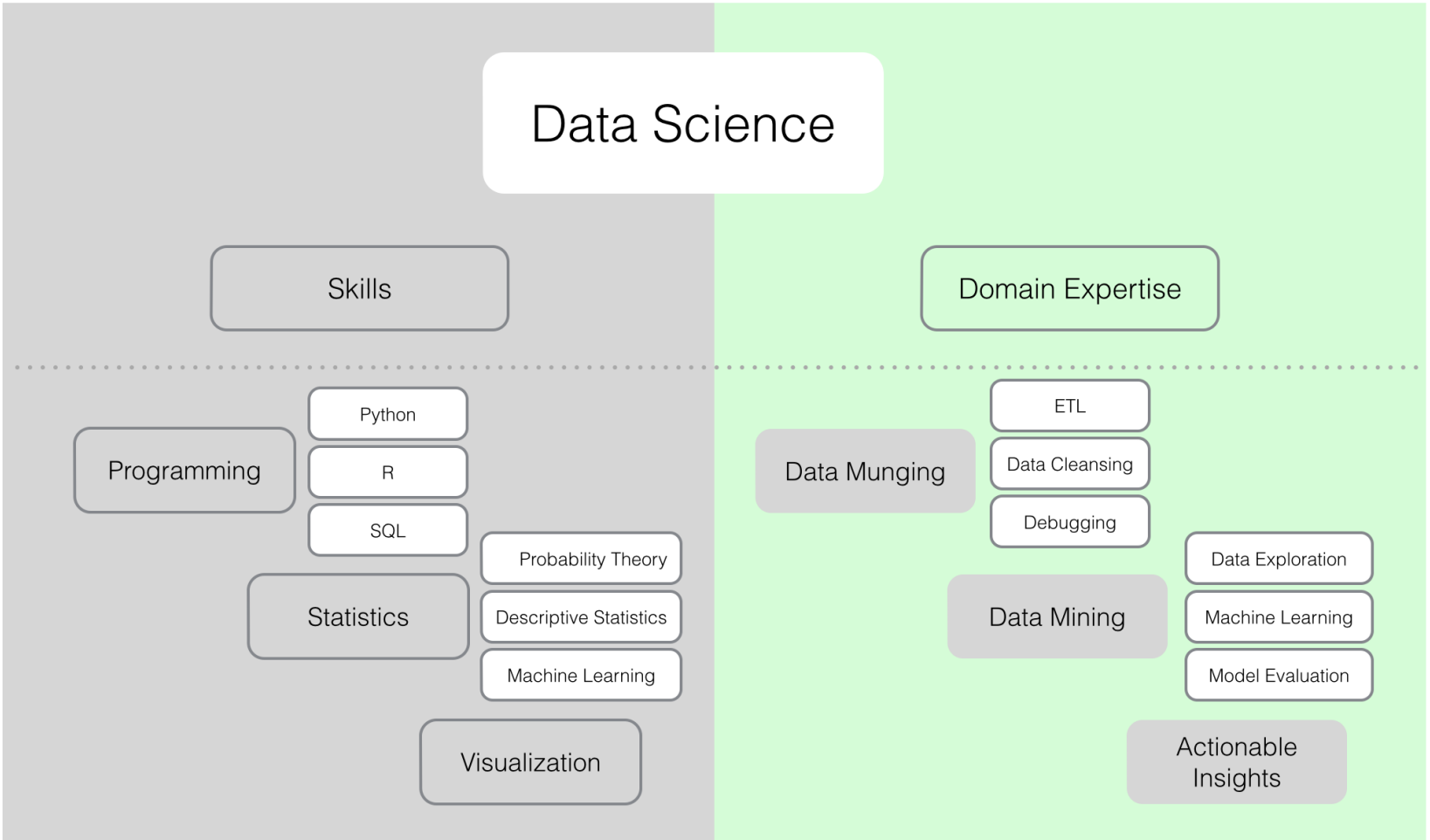
Domain expertise of a data scientist



Skills of a data scientist



Doing data science



Outline

The quest

Definition of a churned customer

Goal

- To predict which customers will churn.

Data sets

- Customer renew data.
- Customer support data.
- Customer account and demographic data.
- Customer usage data.

The quest

Definition of a churned customer

Goal

- To predict which customers will churn.

Data sets

- Customer renew data.
- Customer support data.
- Customer account and demographic data.
- Customer usage data.

The quest

Definition of a churned customer

Goal

- To predict which customers will churn.

Data sets

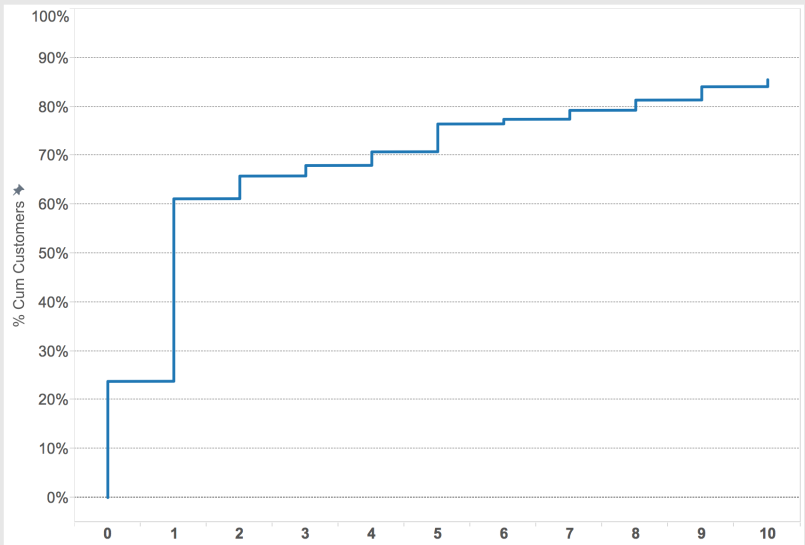
- Customer renew data.
- Customer support data.
- Customer account and demographic data.
- Customer usage data.

Customer renewal data

Raw data

customer_id	is_churn	days_renew
1	False.	0
2	False.	0
3	False.	0
4	False.	0
5	False.	0
6	False.	0
7	False.	0
8	False.	0
9	False.	0
10	False.	0
12	False.	0
13	False.	0

Renewal Schedule

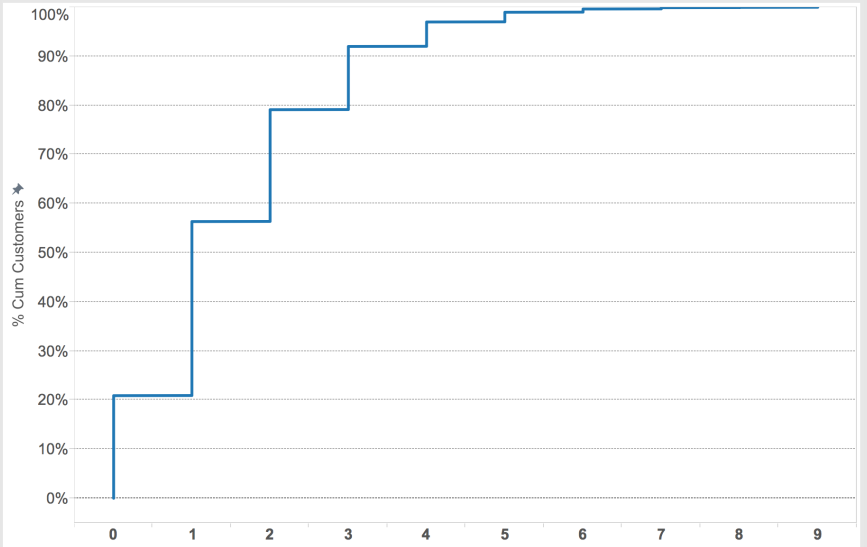


Customer service call data

Raw data

customer_id	cust_surv_calls
1	1
2	1
3	0
4	2
5	3
6	0
7	3
8	0
9	1

Cumulative distribution



Customer plan type and demographic

customer_id	state	account_len	area_code	phone	is_intl_plan	is_vmail_plan
1	KS	128	415	382-4657	no	yes
2	OH	107	415	371-7191	no	yes
3	NJ	137	415	358-1921	no	no
4	OH	84	408	375-9999	yes	no
5	OK	75	415	330-6626	yes	no
6	AL	118	510	391-8027	yes	no
7	MA	121	510	355-9993	no	yes
8	MO	147	415	329-9001	yes	no
9	LA	117	408	335-4719	no	no

Customer usage data

customer_id	vmail_messages	day_mins	day_calls	day_charge	eve_mins	eve_calls	eve_charge	night_mins	night_calls	night_charge	intl_mins	intl_calls	intl_charge
1	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10	3	2.7
2	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.7
3	0	243.4	114	41.38	121.2	110	10.3	162.6	104	7.32	12.2	5	3.29
4	0	299.4	71	50.9	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78
5	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73
6	0	223.4	98	37.98	220.6	101	18.75	203.9	118	9.18	6.3	6	1.7
7	24	218.2	88	37.09	348.5	108	29.62	212.6	118	9.57	7.5	7	2.03
8	0	157	79	26.69	103.1	94	8.76	211.8	96	9.53	7.1	6	1.92
9	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35
10	37	258.6	84	43.96	222	111	18.87	326.4	97	14.69	11.2	5	3.02
12	0	187.7	127	31.91	163.4	148	13.89	196	94	8.82	9.1	5	2.46
13	0	128.8	96	21.9	104.9	71	8.92	141.1	128	6.35	11.2	2	3.02
14	0	156.6	88	26.62	247.6	75	21.05	192.3	115	8.65	12.3	5	3.32
15	0	120.7	70	20.52	307.2	76	26.11	203	99	9.14	13.1	6	3.54
17	27	196.4	139	33.39	280.9	90	23.88	89.3	75	4.02	13.8	4	3.73
18	0	190.7	114	32.42	218.2	111	18.55	129.6	121	5.83	8.1	3	2.19
19	33	189.7	66	32.25	212.8	65	18.09	165.7	108	7.46	10	5	2.7
20	0	224.4	90	38.15	159.5	88	13.56	192.8	74	8.68	13	2	3.51
21	0	155.1	117	26.37	239.7	93	20.37	208.8	133	9.4	10.6	4	2.86

Churn data structure

```
$ customer_id    : int  1 2 3 4 5 6 7 8 9 10 ...
$ state          : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25 19 50 ...
$ account_len    : int  128 107 137 84 75 118 121 147 117 141 ...
$ area_code      : int  415 415 415 408 415 510 510 415 408 415 ...
$ phone          : Factor w/ 3333 levels "327-1058","327-1319",..: 1927 1576 1118 1708 111 2254 1048 81
$ is_intl_plan   : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 1 2 ...
$ is_vmail_plan  : Factor w/ 2 levels "no","yes": 2 2 1 1 1 1 2 1 1 2 ...
$ vmail_messages : int  25 26 0 0 0 0 24 0 0 37 ...
$ day_mins       : num  265 162 243 299 167 ...
$ day_calls      : int  110 123 114 71 113 98 88 79 97 84 ...
$ day_charge     : num  45.1 27.5 41.4 50.9 28.3 ...
$ eve_mins       : num  197.4 195.5 121.2 61.9 148.3 ...
$ eve_calls      : int  99 103 110 88 122 101 108 94 80 111 ...
$ eve_charge     : num  16.78 16.62 10.3 5.26 12.61 ...
$ night_mins     : num  245 254 163 197 187 ...
$ night_calls    : int  91 103 104 89 121 118 118 96 90 97 ...
$ night_charge   : num  11.01 11.45 7.32 8.86 8.41 ...
$ intl_mins      : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
$ intl_calls     : int  3 3 5 7 3 6 7 6 4 5 ...
$ intl_charge    : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
$ cust_surv_calls: int  1 1 0 2 3 0 3 0 1 0 ...
$ is_churn       : Factor w/ 2 levels "False.","True.": 1 1 1 1 1 1 1 1 1 1 ...
$ days_renew     : int  0 0 0 0 0 0 0 0 0 0 ...
```

Outline

Step 1 Load Packages

```
##############################################################################
### Step 1: Load Packages

# clean up space
rm(list = ls(all = TRUE))

# load packages
packages <- c(
  'data.table', 'plyr', 'dplyr', 'reshape2', 'stringr', # general
  'ggplot2', 'lattice', 'lubridate', # plotting and dates
  'pROC', 'ROCR', # ROC curves
  'caret', 'randomForest', 'unbalanced' # modeling
  )
lapply(packages, require, character.only = T)

# clean-up
rm(packages)
```

Step 2 Read Data and Construct Feature

```
##############################################################################
### Step 2: Read Data and Prepare Data                                     ###

# read data
rawdata <- read.csv("../data/churn/churn_fix.csv", header = T)

# check data
str(rawdata)

# create a working copy of the raw data
data <- rawdata

########################################
# Feature Construction: average mins per call

data$day_avg_mins <- ifelse(
  data$day_calls == 0, -1, data$day_mins / data$day_calls
  )
data$eve_avg_mins <- ifelse(
  data$eve_calls == 0, -1, data$eve_mins / data$eve_calls
  )
data$night_avg_mins <- ifelse(
  data$night_calls == 0, -1, data$night_mins / data$night_calls
  )
```

Step 2 Check Correlation

```
##########################################
# Correlation Matrix

# creating a copy for transforming columns to numeric
num_data <- data; vars <- colnames(num_data)
catVars <- vars[
  sapply(num_data[, vars], class) %in% c('factor','character')
  ]
for(v in catVars) {
  num_data[, v] <- as.numeric(as.factor(num_data[,v]))
  }

# correlation data
cor_data <- melt(cor(num_data))
names(cor_data) <- c('v1', 'v2', 'correlation')
cor_data <- mutate(cor_data, correlation = round(correlation, 2))
str(cor_data); head(cor_data) # check correlation data

# plot correlation
ggplot(cor_data, aes(v1, v2, fill = correlation, label = correlation)) +
  geom_tile() +
  geom_text(colour = "white") +
  scale_fill_gradient(low="white", high="black") +
  xlab("") + ylab("") +
  ggtitle("Correlation Matrix")
```

Step 2 Create State Groups

```
##########################################
# Create state groups

data$state <- as.character(data$state)
a <- as.data.frame(table(data$state))
states_select <- as.vector(a[a$Freq > 65, ][[1]])
data$state_group <- ifelse(
  data$state %in% states_select,
  data$state,
  "other"
  )

state_col <- c("state", "state_group")
for(v in state_col) {
  data[,v] <- as.factor(data[,v])
}

str(data[, state_col]) # check result

# clean up
rm(a, v, states_select, state_col)
```

Step 2 Final Preparation and Checking

```
########################################
# remove un-related and redundant columns

cols_remove <- c(
  "customer_id", "days_renew", "state",
  "area_code", "phone", "is_vmail_plan",
  "day_charge", "eve_charge", "night_charge"
  )
data <- data[, setdiff(colnames(data), cols_remove)]

# clean up
rm(cols_remove)

########################################
# convert character to factor

vars <- colnames(data)
catVars <- vars[
  sapply(data[, vars], class) %in% c('character')
  ]
for(v in catVars) {
  data[,v] <- as.factor(data[,v])
}

# clean up
rm(vars, catVars, v)

########################################

# check missing data
if (nrow(data[!complete.cases(data),]) > 0) {
  print("There is missing data!!!")
}
```

Step 3 Splitting Data

```
#############################################################################
### Step 3: random forest                                                 ###

# first, we need to split data into training and testing
splitdf <- function(dataframe, seed = NULL, ratio = 0.5) {
  if (!is.null(seed))
    set.seed(seed)
  index <- 1:nrow(dataframe)
  trainindex <- sample(index, trunc(length(index) * ratio))
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset=trainset,testset=testset)
}
getseed = as.integer(runif(1, 1, 100000)); # random seeding
print(paste("Seeding for split:", getseed)) # record the seed for reproducibility

# split data by 2:1 ratio since the data set is small
splits <- splitdf(data, seed = getseed, ratio = 2/3)
lapply(splits, nrow)
lapply(splits, head)
training <- splits$trainset
testing <- splits$testset
```

Step 3 Random Forest

```
##############################################
# Run Random Forest

model <- randomForest(
  is_churn~.
  ,data = training
  ,ntree = 100
  ,mtry = 5 # take square root of number of features
  ,replace = T
  ,importance = T
  # ,do.trace = T #watch OOB
)
```

Step 3 ROC

```
##########################################
# Model evaluation: ROC

target = "is_churn"
train_target = grep(target, colnames(training))
test_target = grep(target, colnames(testing))

# plot ROC curves for training and testing data
training_actual <- training[, train_target]
training_predicted <- predict(
  model, newdata = training[ , - train_target], type = "prob")
testing_actual <- testing[, test_target]
testing_predicted <- predict(
  model, newdata = testing[ , - test_target], type = "prob")

par(mfrow = c(1,2))
par(pty = "s") # square plotting region
plot.roc(training_actual, training_predicted[, 2], col = "blue", print.auc = T)
plot.roc(testing_actual, testing_predicted[, 2], col = "red", print.auc = T)
par(mfrow = c(1,1))

# clean up
rm(training_actual, training_predicted,
   testing_actual, testing_predicted, target,
   testing, training, train_target, test_target)
```

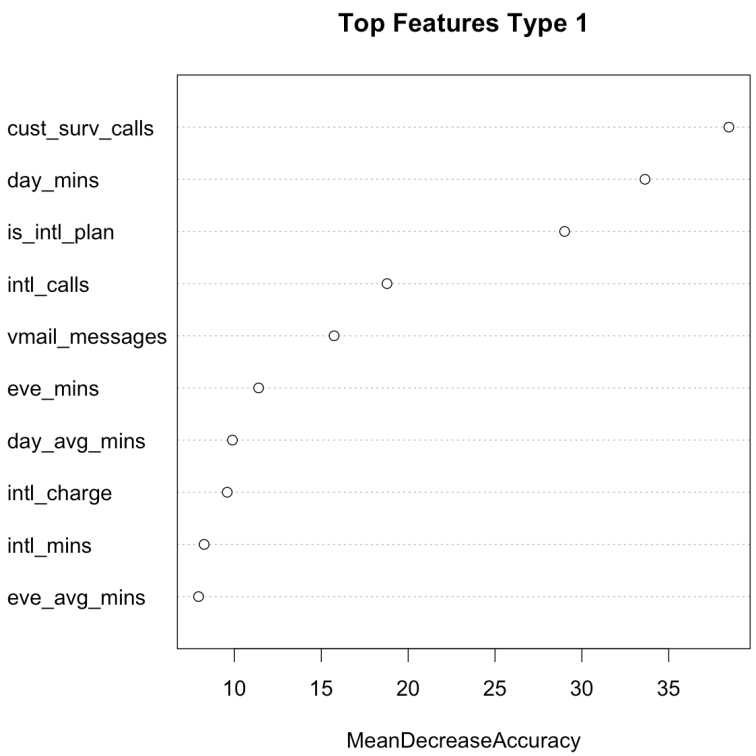
Step 3 Top Features

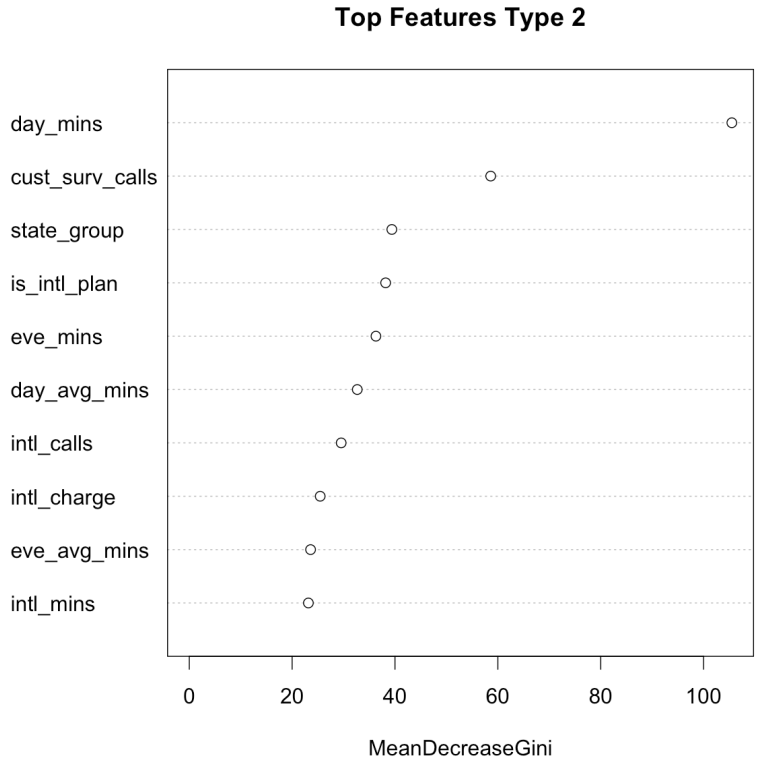
```
##########################################
# Model evaluation: Top Features

par(mfrow = c(1,2))
par(pty = "s") # square plotting region
varImpPlot(
  model
  ,type = 1 # mean decrease in accuracy
  ,n.var = 10
  ,main = "Top Features Type 1")
varImpPlot(
  model
  ,type = 2 # mean decrease in node impurity
  ,n.var = 10
  ,main = "Top Features Type 2")
par(mfrow = c(1,1))
```

Outline

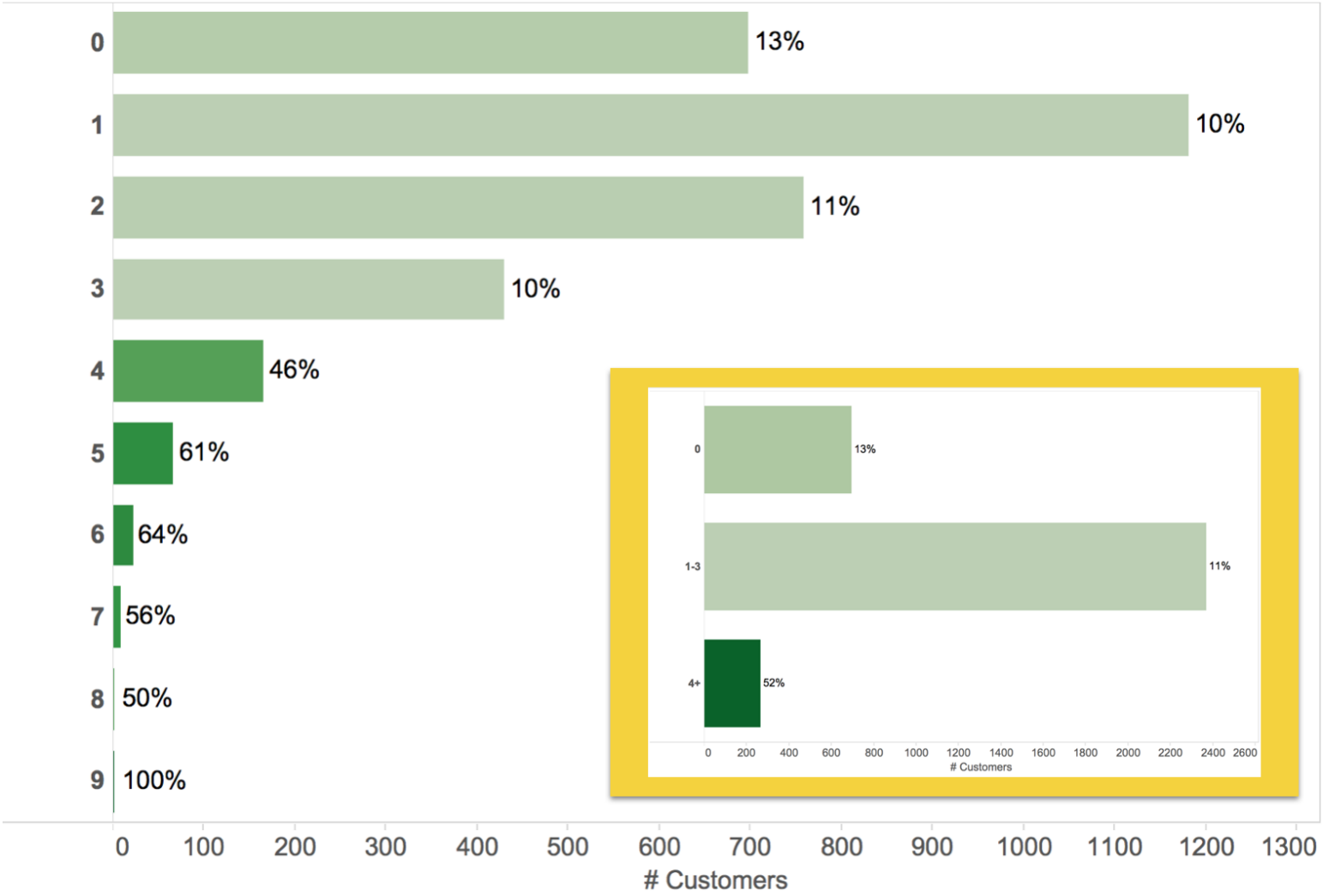
Churn Model: Top Features

Variable importance from random forest



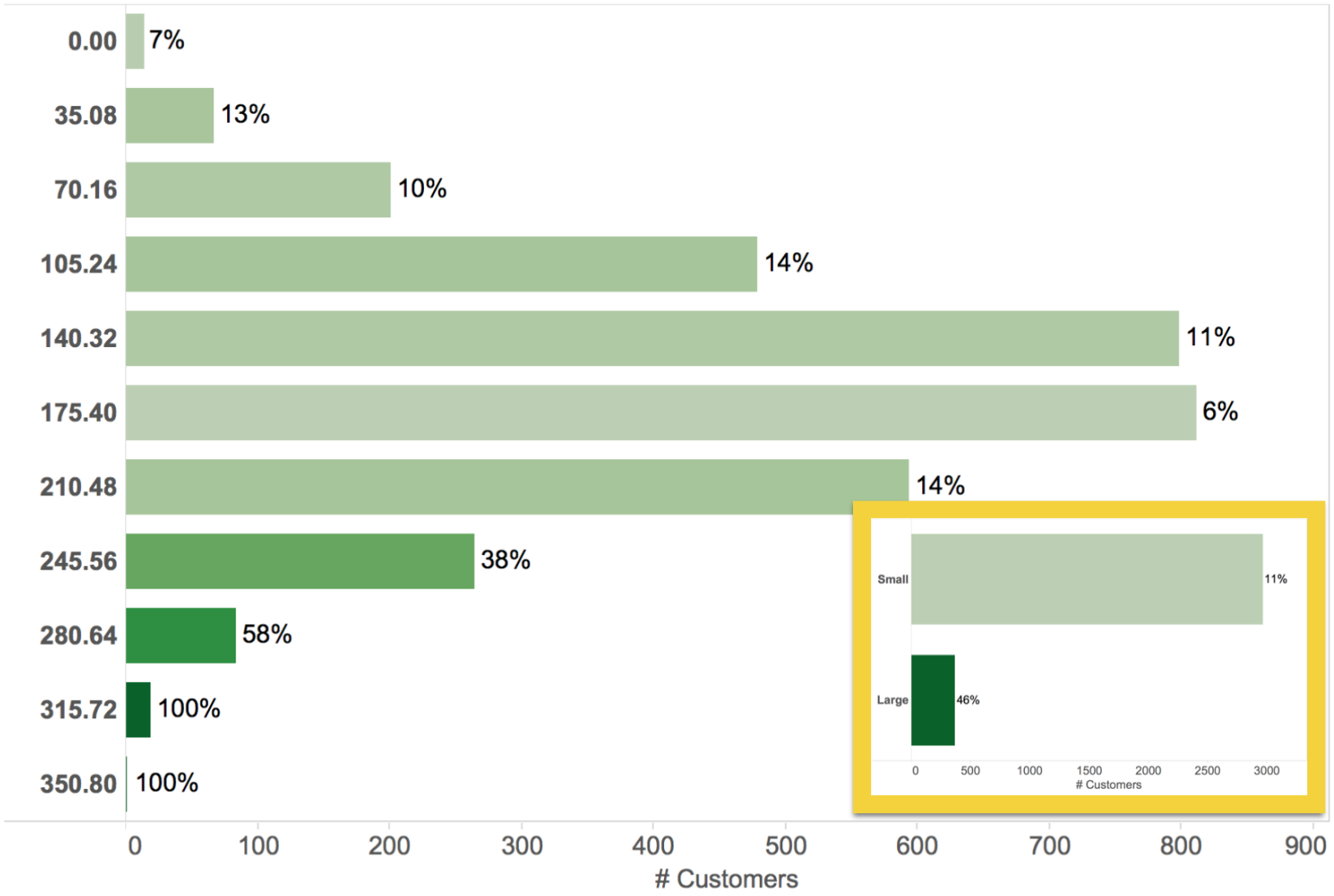


Top feature - customer service calls



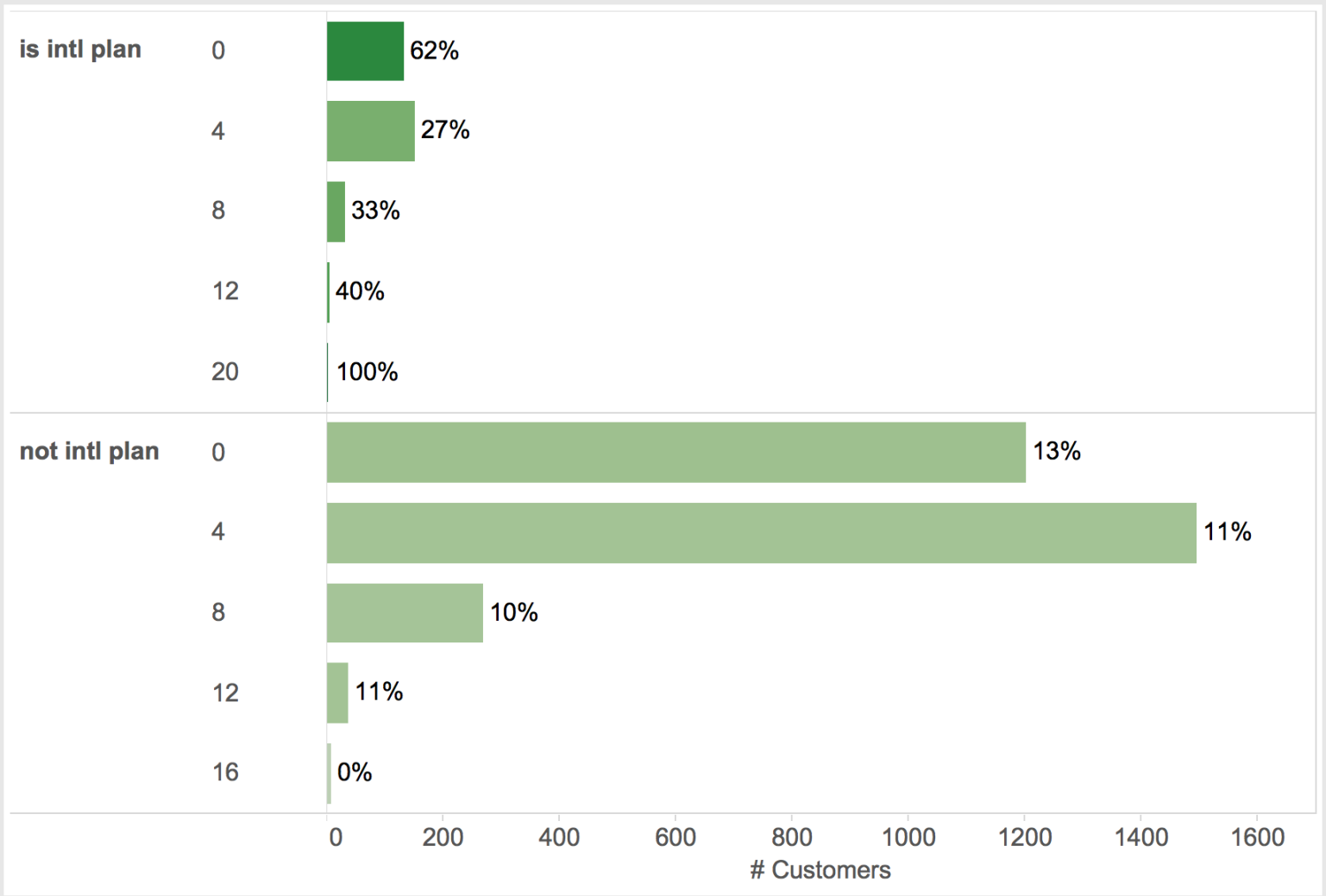
Churn Model: Top Features

Top feature - customer day time minutes



Churn Model: Top Features

Top feature - international plan and international calls



Outline

Churn analysis

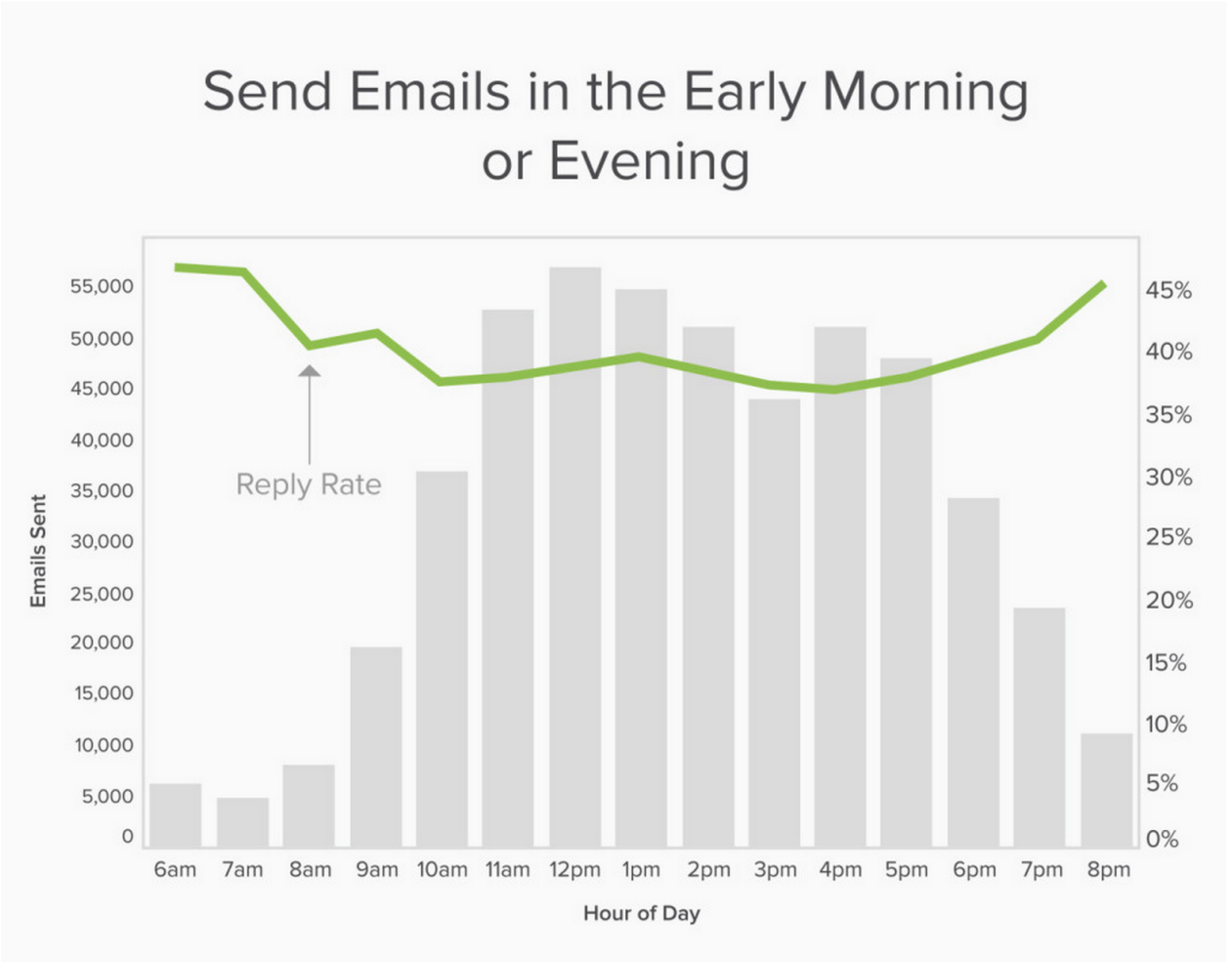
Is Intl Plan	Cust Surv Calls (group)	Day Mins Group	% Churn	Number of Records	Summary
is intl plan	0	Large	60%	15	Small Group
		Small	44%	68	Small Group
	1-3	Large	56%	36	Small Group
		Small	34%	176	
	4+	Large	100%	1	Outlier
		Small	67%	27	Small Group
not intl plan	0	Large	48%	60	Small Group
		Small	4%	554	Low Churn
	1-3	Large	43%	229	
		Small	4%	1,928	Low Churn
	4+	Large	44%	27	Small Group
		Small	50%	212	
Grand Total			**14%**	**3,333**	

Churn analysis

Is Intl Plan	Cust Surv Calls (group)	Day Mins Group	% Churn	# Customers	Summary	Avg Charge	Revenue if we got 10% of the churned customers back
is intl plan	0	Large	60%	15	Small Group	$ 72	$ 65
		Small	44%	68	Small Group	$ 56	$ 168
	1-3	Large	56%	36	Small Group	$ 72	$ 144
		Small	34%	176		$ 55	$ 325
	4+	Large	100%	1	Outlier	$ 78	$ 8
		Small	67%	27	Small Group	$ 56	$ 101
not intl plan	0	Large	48%	60	Small Group	$ 73	$ 212
		Small	4%	554	Low Churn	$ 55	$ 132
	1-3	Large	43%	229		$ 72	$ 706
		Small	4%	1,928	Low Churn	$ 55	$ 418
	4+	Large	44%	27	Small Group	$ 74	$ 89
		Small	50%	212		$ 54	$ 578
Grand Total			**14%**	**3,333**		$ 57	$ 2,660

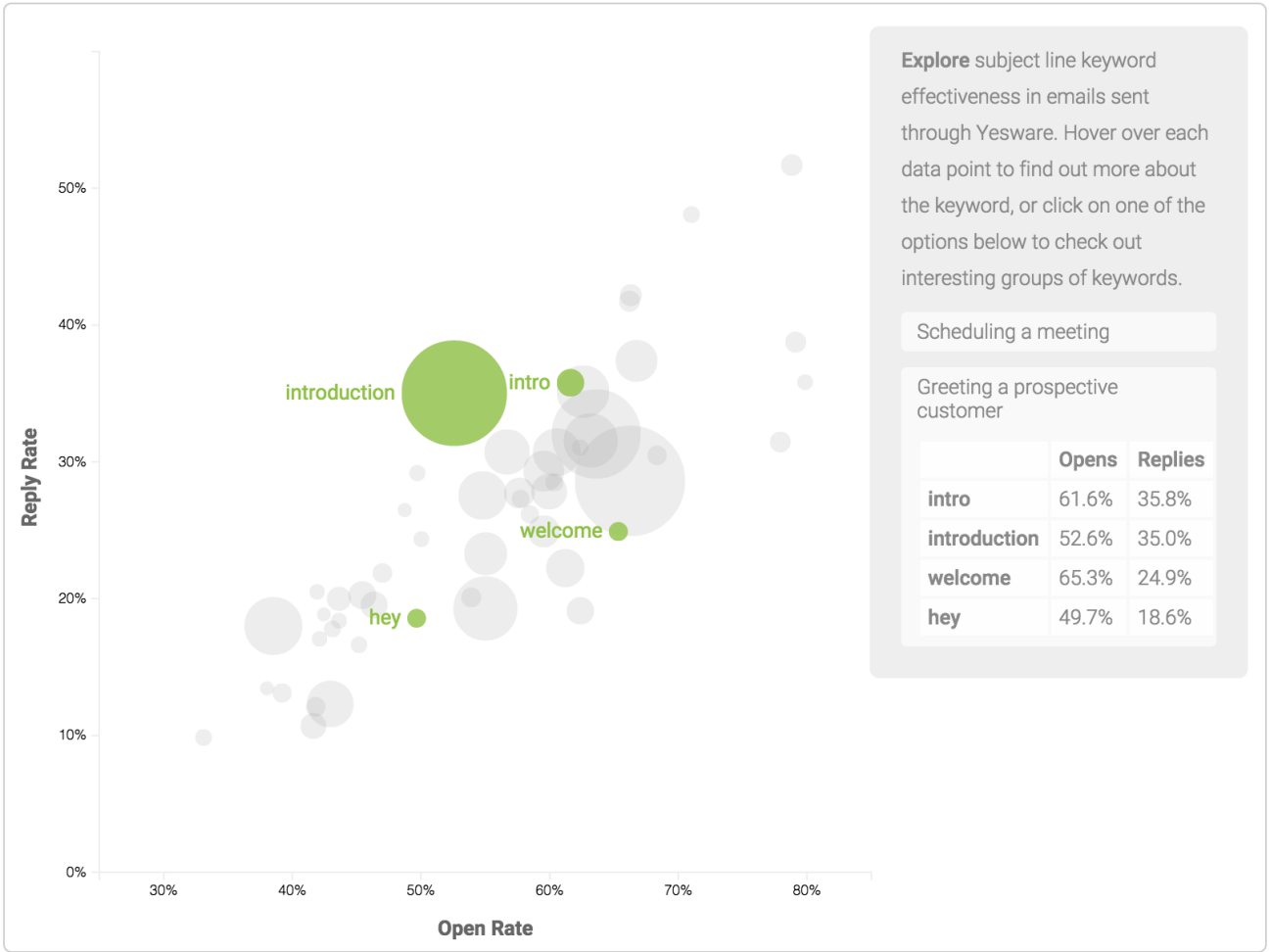
Outline

When to send an email



Best time to send email: `http://goo.gl/lVdD31`

D3 Visualization: Subject Line Analysis



Subject line key words: `http://goo.gl/PK9xh0`

From academia to industry

	Academia	Industry
Goal	Improve human knowledge	Make money
Success Criteria	Publish papers	Create and deliver business value
Approach	Finding a better way to do a hard thing	Finding the fastest way to do easy things
Importance of Speed	**Not as important as other things**	**Very important**

Why do we need a toolbox?

	Academia	Industry
Goal	Improve human knowledge	Make money
Success Criteria	Publish papers	Create and deliver business value
Approach	Finding a better way to do a hard thing	Finding the fastest way to do easy things
Importance of Speed	**Not as important as other things**	**Very important**

Python and R for data science

	Python	R
Powerful Packages	Numpy, Scipy, Pandas Scikit-Learn	caret, rROC, ggplot2, dplyr, reshape2
Community Support	fast growing	strongest support from the statistics community
Data Munging	****	***
Data Exploration	****	*****
Machine Learning	****	*****
Programming Interface	iPython	RStudio

iPython `http://goo.gl/zT4uPE` — RStudio `http://www.rstudio.com/`

Data manipulation tools

	Python	R	Unix	SQL	Scala
Powerful Packages / Library	★★★★★	★★★★★	**	*	★★★★
Community Support	★★★★★	★★★★★	★★★★	***	★★★★
Data Munging	★★★★	***	★★★★	★★★★	★★★★
Data Exploration	★★★★	★★★★★	**	***	***
Machine Learning	★★★★	★★★★★	*	**	★★★★

Data visualization tools

	Excel	R	Tableau	D3
Ease of Learning	★★★★★	★★★	★★★★★	★
Is Free	No	Yes	No	Yes
Good for Data Exploration	★★	★★★★	★★★★★	★★★
Flexibility in Data Representation	★★	★★★★	★★★★	★★★★★
Good for Reporting and Sharing	★★★★	★★★★	★★★	★★★★

D3 `http://d3js.org/`

Thank You

Please send your questions and feedbacks to me!