# Allstate_Purchase_Prediction.R

puj83

Wed Jul 01 15:16:15 2020

```
# install.packages("maps")

# Load required libraries
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

library(maps)

## Warning: package 'maps' was built under R version 3.5.3

# library(ROCR)
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(nnet)
library(ggplot2)


## LOADING DATA AND ADDING FEATURES ##

# Define column classes for reading in the data sets
colClasses <- c(rep("integer", 4), "character", rep("factor", 2),
                "integer", "factor", "integer", "factor", rep("integer", 3),
                rep("factor", 2), "integer", rep("factor", 7), "integer")

# Function for pre-processing and feature engineering
preprocess <- function(data) {

  # add features
  data$plan <- paste0(data$A, data$B, data$C, data$D, data$E, data$F, data$G)
  data$hour <- as.integer(substr(data$time, 1, 2))
  data$timeofday <- as.factor(ifelse(data$hour %in% 6:15, "day",
                                     ifelse(data$hour %in% 16:18, "evening",
                                            "night")))
  data$weekend <- as.factor(ifelse(data$day %in% 0:4, "No", "Yes"))
  data$family <- as.factor(ifelse(data$group_size > 2 & data$age_youngest <
                                    25 & data$married_couple==1, "Yes",
"No"))
  data$agediff <- data$age_oldest-data$age_youngest
  data$individual <- as.factor(ifelse(data$agediff==0 & data$group_size==1,
                                      "Yes", "No"))
  data$stategroup <- as.factor(ifelse(data$state %in% c("SD","ND"), "g1",
                                      ifelse(data$state %in% c("AL","WY"),
"g2",
                                             ifelse(data$state %in%
c("OK","ME","AR",
                                                                      "WV"),
"g3",
                                                    ifelse(data$state %in%
c("DC","NE","GA"),
                                                           "g5", "g4")))))

  # fix NA's for duration_previous and C_previous
  data$duration_previous[is.na(data$duration_previous)] <- 0
  levels(data$C_previous) <- c("1", "2", "3", "4", "none")
  data$C_previous[is.na(data$C_previous)] <- "none"

  # replace risk_factor NA's by predicting a value
  datanorisk <- data[is.na(data$risk_factor), ]
```

```r
  datarisk <- data[!is.na(data$risk_factor), ]
  lm.fit <- lm(risk_factor ~ age_youngest*group_size+married_couple+
                 homeowner, data=datarisk)
  lm.pred <- predict(lm.fit, newdata=datanorisk)
  data$risk_factor[is.na(data$risk_factor)] <- round(lm.pred, 0)

  # for car_age greater than 30, "round down" to 30
  data$car_age[data$car_age > 30] <- 30

  return(data)
}

# read in training set
train <- 
read.csv("C:/Users/puj83/OneDrive/Portfolio/AllState_Purchase_Prediction/train.csv", colClasses=colClasses)
train <- preprocess(train)

# trainsub is subset of train that only includes purchases
trainsub <- train[!duplicated(train$customer_ID, fromLast=TRUE), ]

# trainex is subset of train that excludes purchases
trainex <- train[duplicated(train$customer_ID, fromLast=TRUE), ]

# trainex2 only includes last quote before purchase
trainex2 <- trainex[!duplicated(trainex$customer_ID, fromLast=TRUE), ]

# changed is anyone who changed from their last quote
changed <- ifelse(trainsub$plan == trainex2$plan, "No", "Yes")
changelog <- ifelse(trainsub$plan == trainex2$plan, FALSE, TRUE)
trainsub$changed <- as.factor(changed)
trainex2$changed <- as.factor(changed)
trainsub$changelog <- changelog
trainex2$changelog <- changelog

# compute "stability" feature from trainex and add to trainex2
customerstability <- trainex %>% group_by(customer_ID) %>%
  summarise(quotes=n(), uniqueplans=n_distinct(plan),
            stability=(quotes-uniqueplans+1)/quotes)
trainex2$stability <- customerstability$stability

# compute "planfreq" feature on trainex2
nrowtrainex2 <- nrow(trainex2)
planfreqs <- trainex2 %>% group_by(plan) %>%
  summarise(planfreq=n()/nrowtrainex2)
trainex2 <- left_join(trainex2, planfreqs)
```

```
## Joining, by = "plan"
```

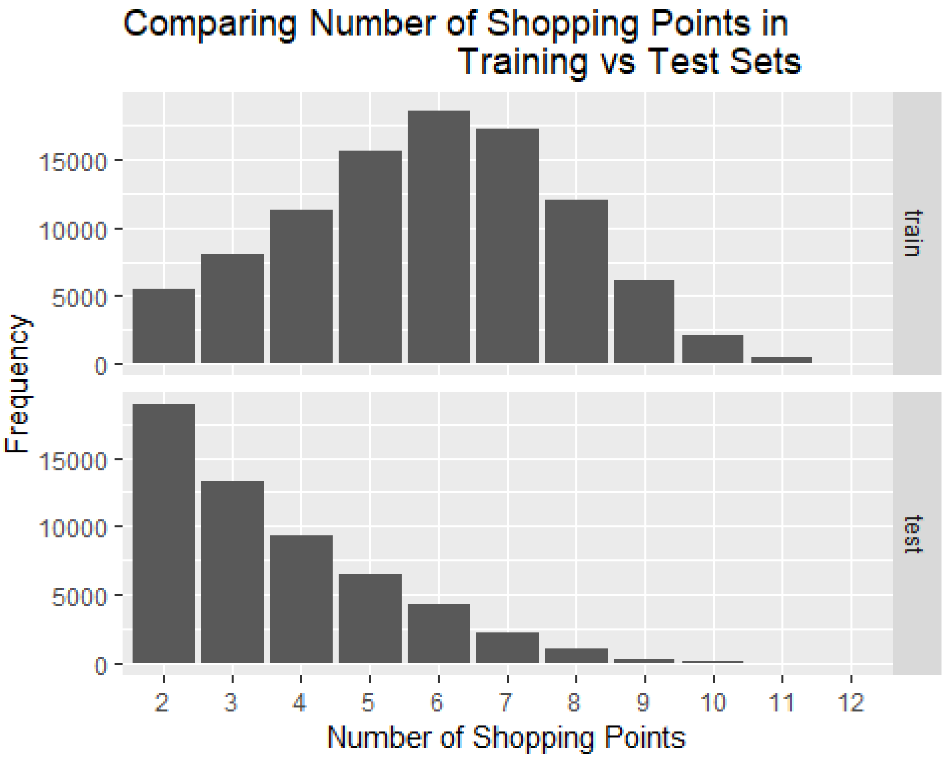
```r
# trainex3 is identical to trainex2 but also includes purchases
trainex3 <- cbind(trainex2, Apurch=trainsub$A, Bpurch=trainsub$B,
                  Cpurch=trainsub$C, Dpurch=trainsub$D, Epurch=trainsub$E,
Fpurch=trainsub$F,
                  Gpurch=trainsub$G, planpurch=trainsub$plan,
stringsAsFactors=FALSE)

# read in test set
test <-
read.csv("C:/Users/puj83/OneDrive/Portfolio/AllState_Purchase_Prediction/test
_v2.csv", colClasses=colClasses)
test <- preprocess(test)

# fix locations that are NA
s <- split(test$location, test$state)
s2 <- sapply(s, function(x) x[1])
NAstates <- test[is.na(test$location), "state"]
NAlocations <- s2[NAstates]
test$location[is.na(test$location)] <- NAlocations

# add "changed" variable and default to No
test$changed <- factor(rep("No", nrow(test)), levels=c("No", "Yes"))

# testsub only shows last (known) quote before purchase
testsub <- test[!duplicated(test$customer_ID, fromLast=TRUE), ]

# compute "stability" feature from test and add to testsub
customerstability <- test %>% group_by(customer_ID) %>% summarise(quotes=n(),

uniqueplans=n_distinct(plan), stability=(quotes-uniqueplans+1)/quotes)
testsub$stability <- customerstability$stability

# compute "planfreq" feature on testsub
nrowtestsub <- nrow(testsub)
planfreqs <- testsub %>% group_by(plan) %>%
summarise(planfreq=n()/nrowtestsub)
testsub <- left_join(testsub, planfreqs)
```

```
## Joining, by = "plan"
```

```
## DATA EXPLORATION ##
```

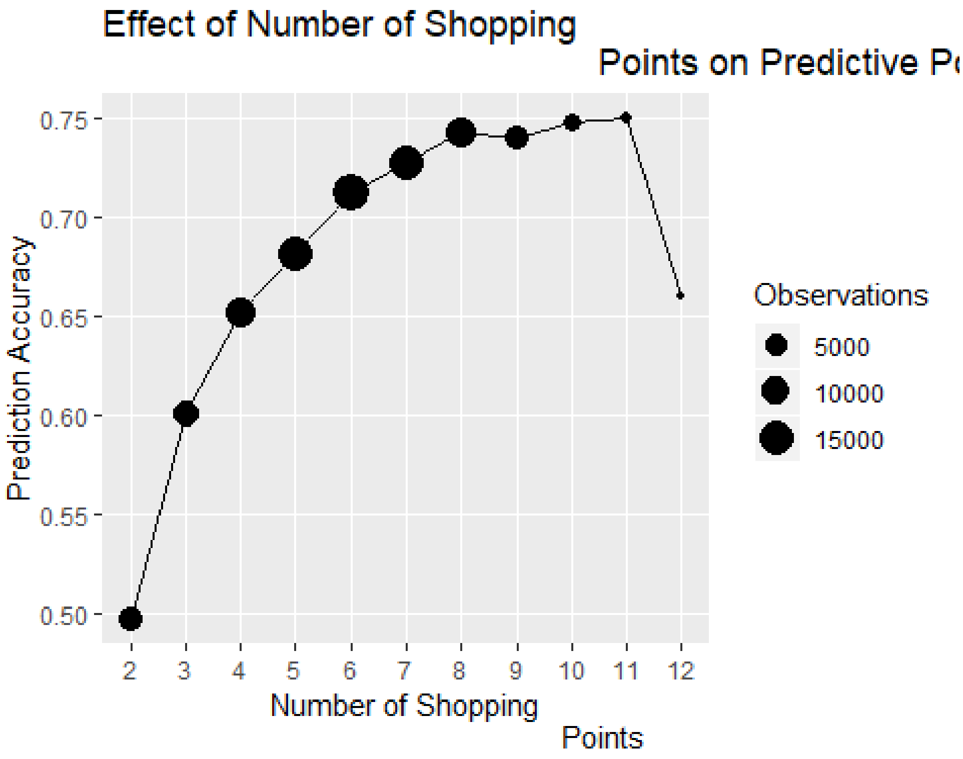
```r
# check for NA values
sapply(train, function(x) mean(is.na(x)))
```

```
##       customer_ID       shopping_pt       record_type               day
##                 0                 0                 0                 0
##              time             state          location        group_size
##                 0                 0                 0                 0
##         homeowner           car_age         car_value       risk_factor
##                 0                 0                 0                 0
##        age_oldest      age_youngest    married_couple        C_previous
##                 0                 0                 0                 0
## duration_previous                 A                 B                 C
##                 0                 0                 0                 0
##                 D                 E                 F                 G
##                 0                 0                 0                 0
##              cost              plan              hour         timeofday
##                 0                 0                 0                 0
##           weekend            family           agediff        individual
##                 0                 0                 0                 0
##        stategroup
##                 0

# risk_factor, C_previous, duration_previous
sapply(test, function(x) mean(is.na(x)))

##       customer_ID       shopping_pt       record_type               day
##                 0                 0                 0                 0
##              time             state          location        group_size
##                 0                 0                 0                 0
##         homeowner           car_age         car_value       risk_factor
##                 0                 0                 0                 0
##        age_oldest      age_youngest    married_couple        C_previous
##                 0                 0                 0                 0
## duration_previous                 A                 B                 C
##                 0                 0                 0                 0
##                 D                 E                 F                 G
##                 0                 0                 0                 0
##              cost              plan              hour         timeofday
##                 0                 0                 0                 0
##           weekend            family           agediff        individual
##                 0                 0                 0                 0
##        stategroup           changed
##                 0                 0

# risk_factor, C_previous, duration_previous, location

uniquetrainplan <- unique(train$plan)
uniquetestplan <- unique(test$plan)
# plans in train: 1809
# plans in test: 1596
# union: 1878 (69 plans in test that are not in train)
# intersection: 1527


## VISUALIZATIONS ##

# Viz 1: Number of shopping points
shoptrain <- data.frame(maxpoint=trainex2$shopping_pt, dataset=rep("train",
nrow(trainex2)))
shoptest <- data.frame(maxpoint=testsub$shopping_pt, dataset=rep("test",

nrow(testsub)))
shoppingpoints <- rbind(shoptrain, shoptest)
shoppingpoints$dataset <- as.factor(shoppingpoints$dataset)
ggplot(shoppingpoints) + aes(factor(maxpoint)) + geom_histogram(stat =
"count") +
  facet_grid(dataset ~ .) + labs(x="Number of Shopping Points",
                                 y="Frequency", title="Comparing Number of
Shopping Points in
                                 Training vs Test Sets")

## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

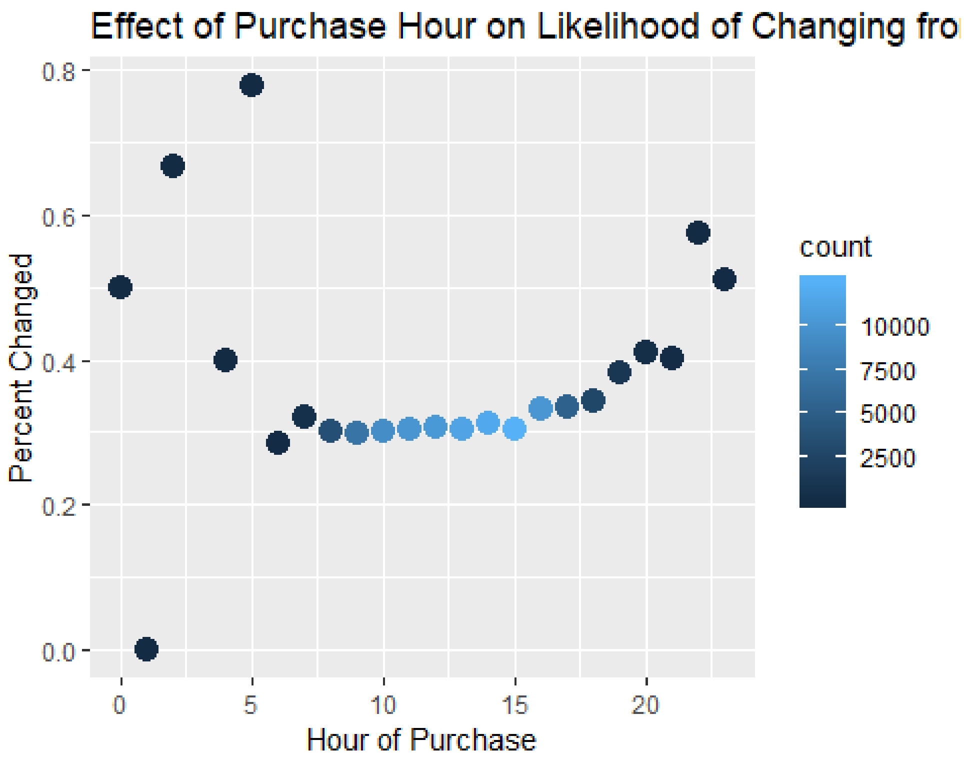


```
# Viz 2: Predictive power of final quote before purchase
s <- split(trainex2, trainex2$shopping_pt)
s2 <- sapply(s, function(x) sum(x$changed=="No")/nrow(x))
s2b <- sapply(s, nrow)
acclastentry <- data.frame(numshoppoints=as.integer(names(s2)), accuracy=s2,
                           Observations=s2b)
ggplot(acclastentry) + aes(numshoppoints, accuracy, size=Observations) +
  geom_point() + geom_line(size=0.5) + scale_x_continuous(breaks=1:12) +
  theme(panel.grid.minor=element_blank()) + labs(x="Number of Shopping
                                                 Points", y="Prediction
Accuracy", title="Effect of Number of Shopping
```

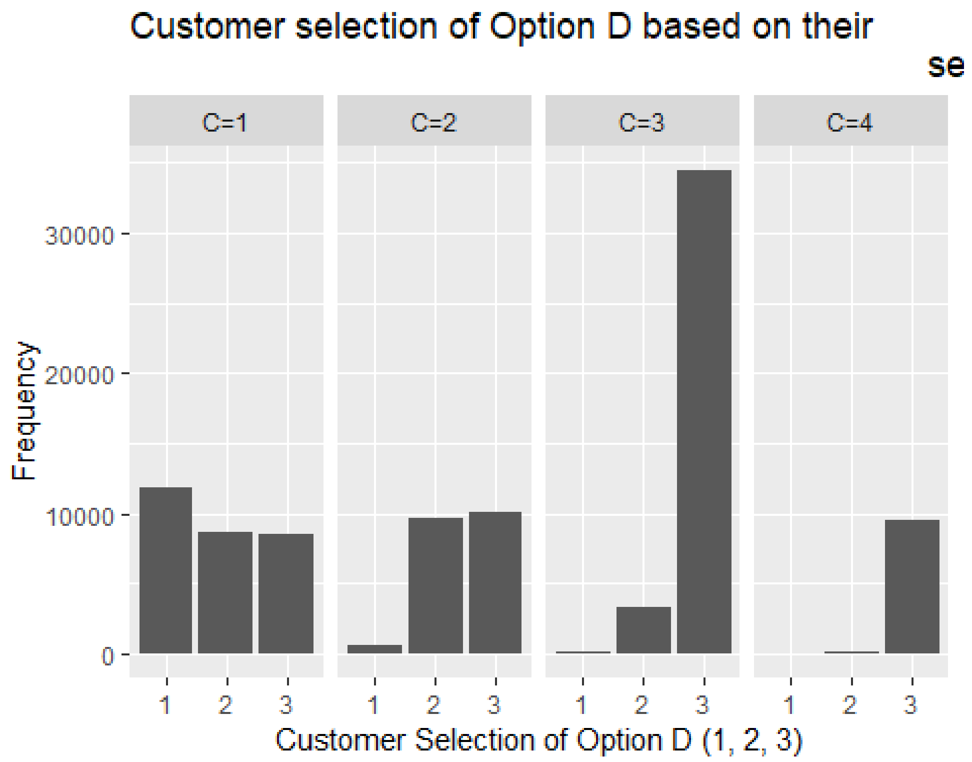
```
                                                  Points on Predictive Power
of Last Quote")
```



```
# Viz 3: Effect of purchase hour on the predictive power of the final quote
s3 <- split(trainsub, trainsub$hour)
s4 <- sapply(s3, function(x) sum(x$changed=="Yes")/nrow(x))
s5 <- as.data.frame(table(trainsub$hour))$Freq
changebyhour <- data.frame(hour=as.integer(names(s4)),
                           percentchanged=s4, count=s5)
ggplot(changebyhour) + aes(hour, percentchanged, color=count) +
  geom_point(size=4) + labs(x="Hour of Purchase", y="Percent Changed",
                            title="Effect of Purchase Hour on Likelihood of
Changing from Last Quote")
```

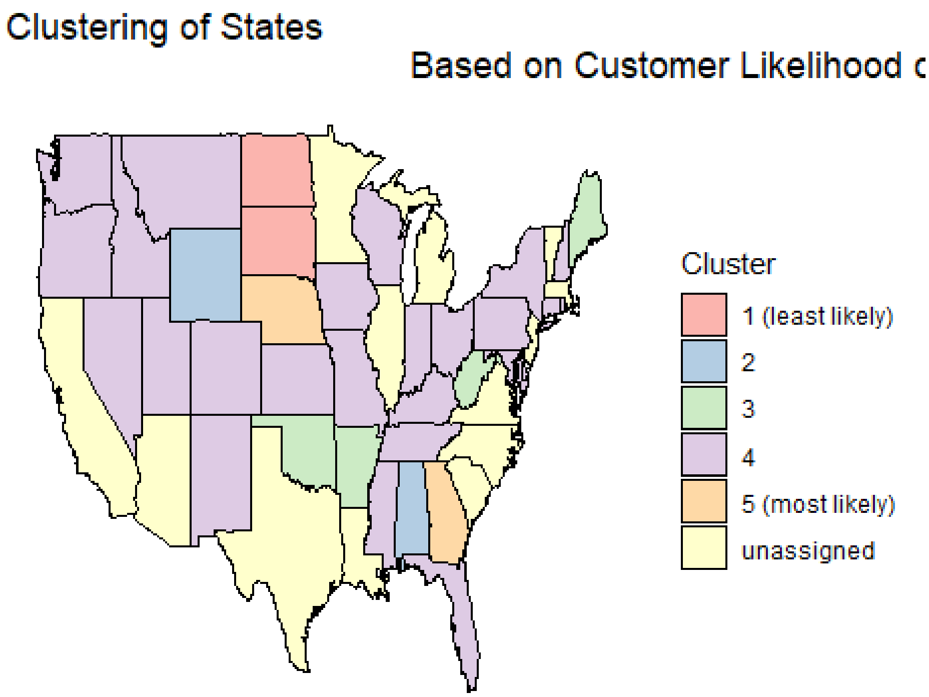


```
# Viz 4: Dependencies between options
C_names <- list("1"="C=1", "2"="C=2", "3"="C=3", "4"="C=4")
C_labeller <- function(variable, value){ return(C_names[value]) }
ggplot(trainsub, aes(D)) + geom_bar() + facet_grid(. ~ C,
                                                   labeller=C_labeller) +
labs(x="Customer Selection of Option D (1, 2, 3)",

y="Frequency", title="Customer selection of Option D based on their

selection for Option C")

## Warning: The labeller API has been updated. Labellers taking `variable`and
## `value` arguments are now deprecated. See labellers documentation.
```



```
# Viz 5: Clustering of states
# based on: http://is-r.tumblr.com/post/37708137014/us-state-maps-using-map-
data
states <- map_data("state")
states$grp <- as.factor(ifelse(states$region %in% c("south dakota",
                                                    "north dakota"), "1
(least likely)",
                               ifelse(states$region %in%
c("alabama","wyoming"), "2",
                                      ifelse(states$region %in%
c("oklahoma","maine","arkansas","west virginia"),
                                             "3",
                                             ifelse(states$region %in%
c("colorado","connecticut","delaware","florida",

"iowa","idaho","indiana","kansas","kentucky","maryland","missouri",

"mississippi","montana","new hampshire","new mexico","nevada",
                                                                         "new
york","ohio","oregon","pennsylvania","rhode island","tennessee",

"utah","washington","wisconsin"), "4",
                                                    ifelse(states$region %in%
c("district of columbia","nebraska","georgia"),
                                                           "5 (most likely)",
"unassigned"))))))
ggplot(states) + aes(x=long, y=lat, group=group, fill=grp) +
```

```r
  geom_polygon(color="black") + theme_bw() +
  theme(panel.border=element_blank()) + scale_y_continuous(breaks=c()) +
  scale_x_continuous(breaks=c()) + labs(title="Clustering of States
                                        Based on Customer Likelihood of
Changing from Last Quote", fill="Cluster",
                                        x="", y="") +
scale_fill_brewer(palette="Pastel1")
```



```r
# Building out a Random Forest Model

# random forests for predicting "changed"
rf.fit <- randomForest(changed ~ stategroup+cost+A+C+D+E+F+G+age_oldest+

age_youngest+car_value+car_age+shopping_pt+timeofday+weekend+risk_factor+
                         C_previous+duration_previous+stability+planfreq,
data=trainex2, mtry=5)
rf.pred <- ifelse(rf.fit$votes[, 2]>0.5, "Yes", "No")
```