# Customer Coversion

Michael Guel

10/28/2022

```
library(readxl)
library(ggplot2)
library(dplyr)
```

```
##
## 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(glmnet)
```

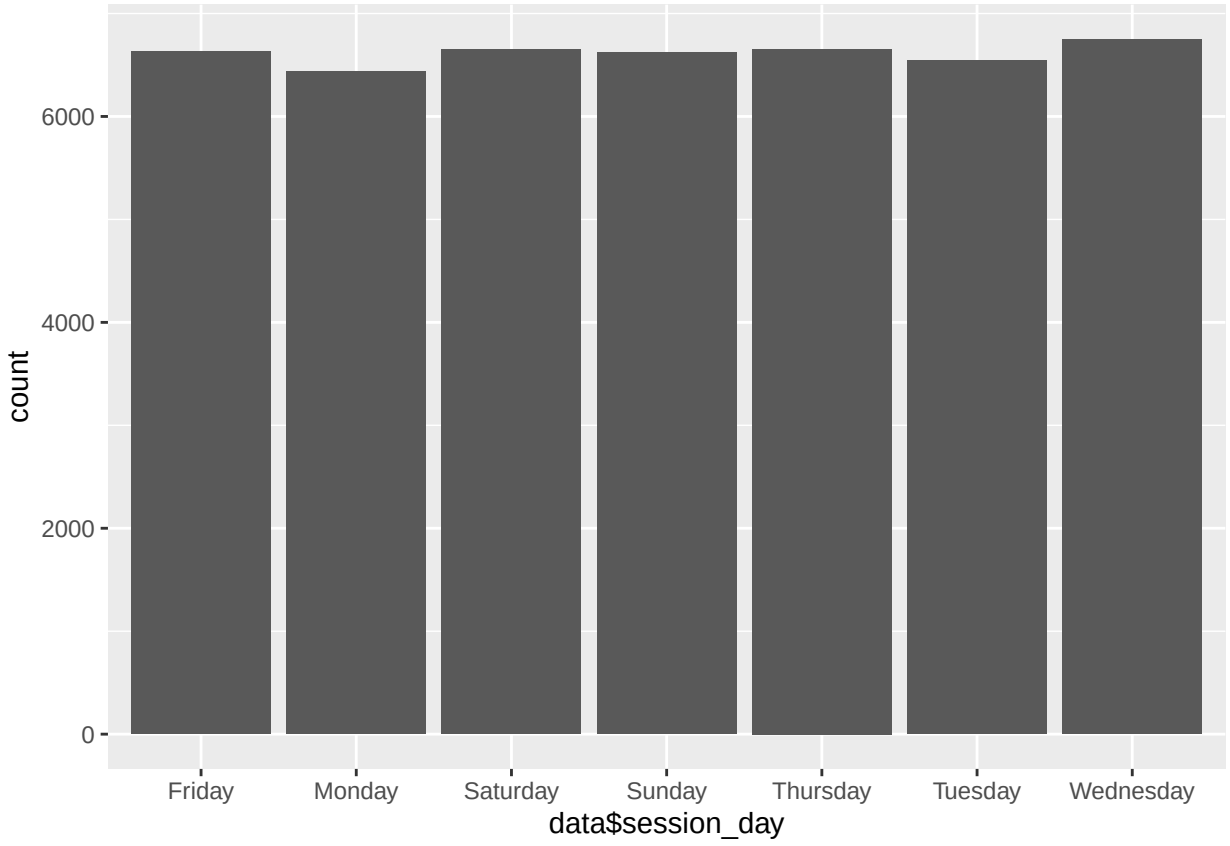
```
## Warning: package 'glmnet' was built under R version 4.1.3

## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.1.3

##
## Attaching package: 'lubridate'

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

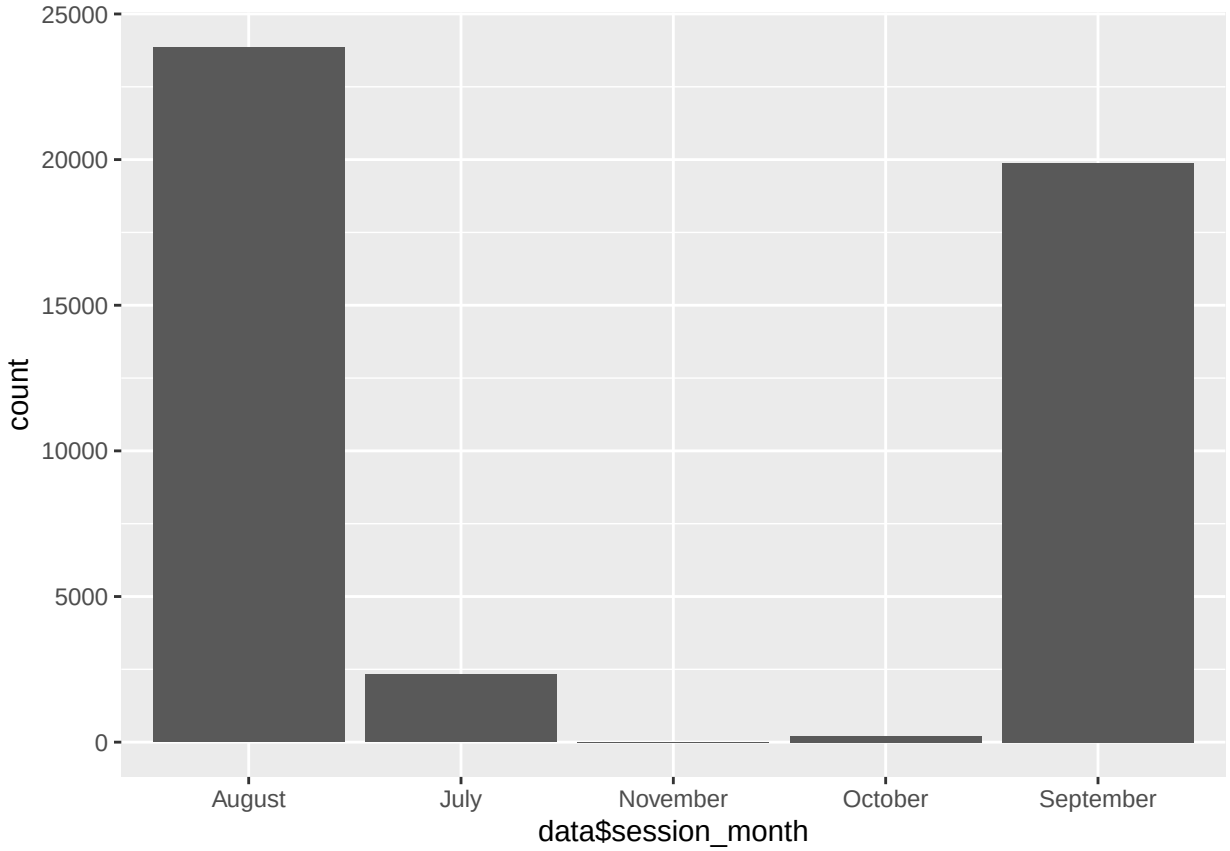
```
library(scatterplot3d)


### LOAD THE DATA IN AND MERGE

session = read.csv('sessions.csv', sep = ',')

transactions = read.csv('transactions.csv', sep = ',')

data= merge(session,transactions,by=c("session_id"))

### USE YMD TO CONVERT DATE TO DATE TYPE

data$session_dt = ymd(data$session_dt)


### ADD TWO FEATURE FOR DATE WHICH TELLS WHAT DAY AND WHAT MONTH

data = data %>% mutate(session_day = weekdays(as.Date(data$session_dt)),session_month = months(as.Date(

data$session_day = as.factor(data$session_day)

data$session_month = as.factor(data$session_month)

### EDA

ggplot(data = data, aes(data$session_day)) +
  geom_bar()
```

```
## Warning: Use of ‘data$session_day‘ is discouraged. Use ‘session_day‘ instead.
```



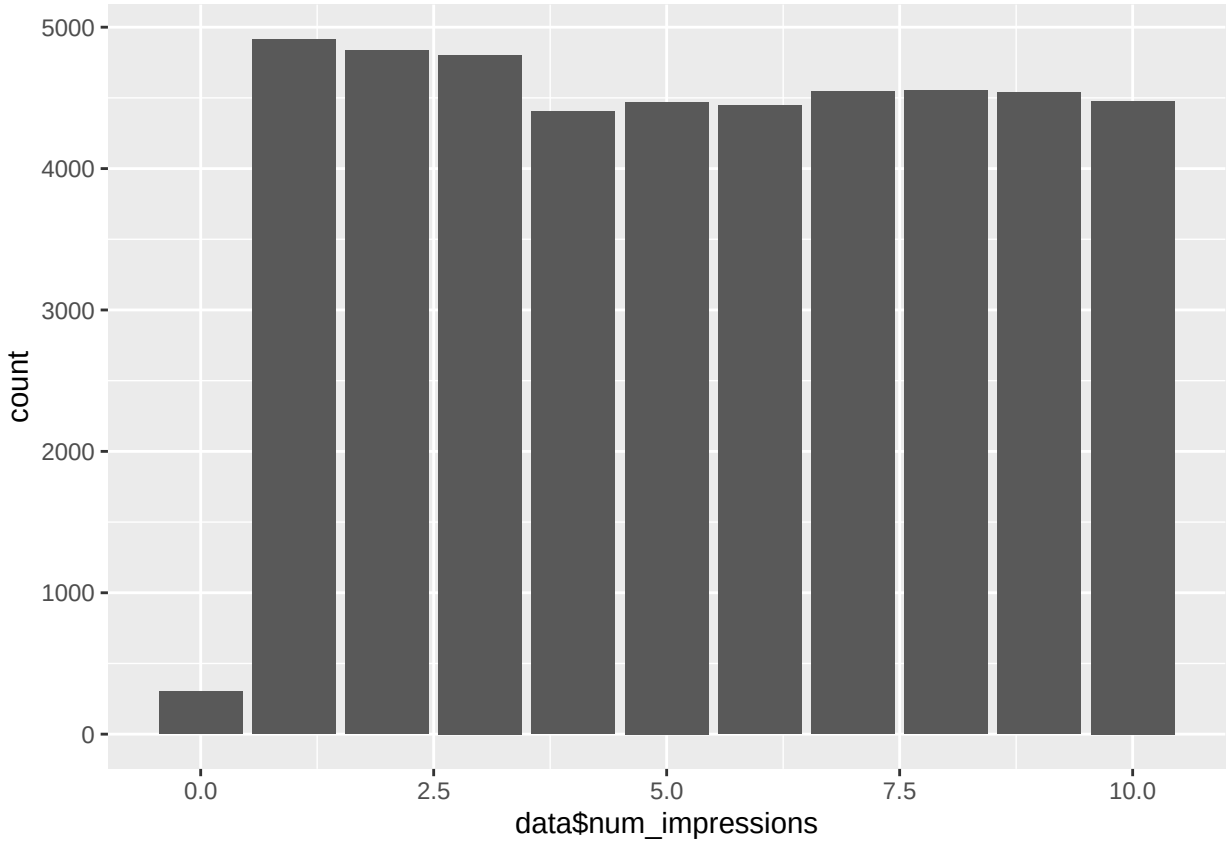

```
ggplot(data = data, aes(data$session_month)) +
  geom_bar()
```

```
## Warning: Use of ‘data$session_month‘ is discouraged. Use ‘session_month‘
## instead.
```



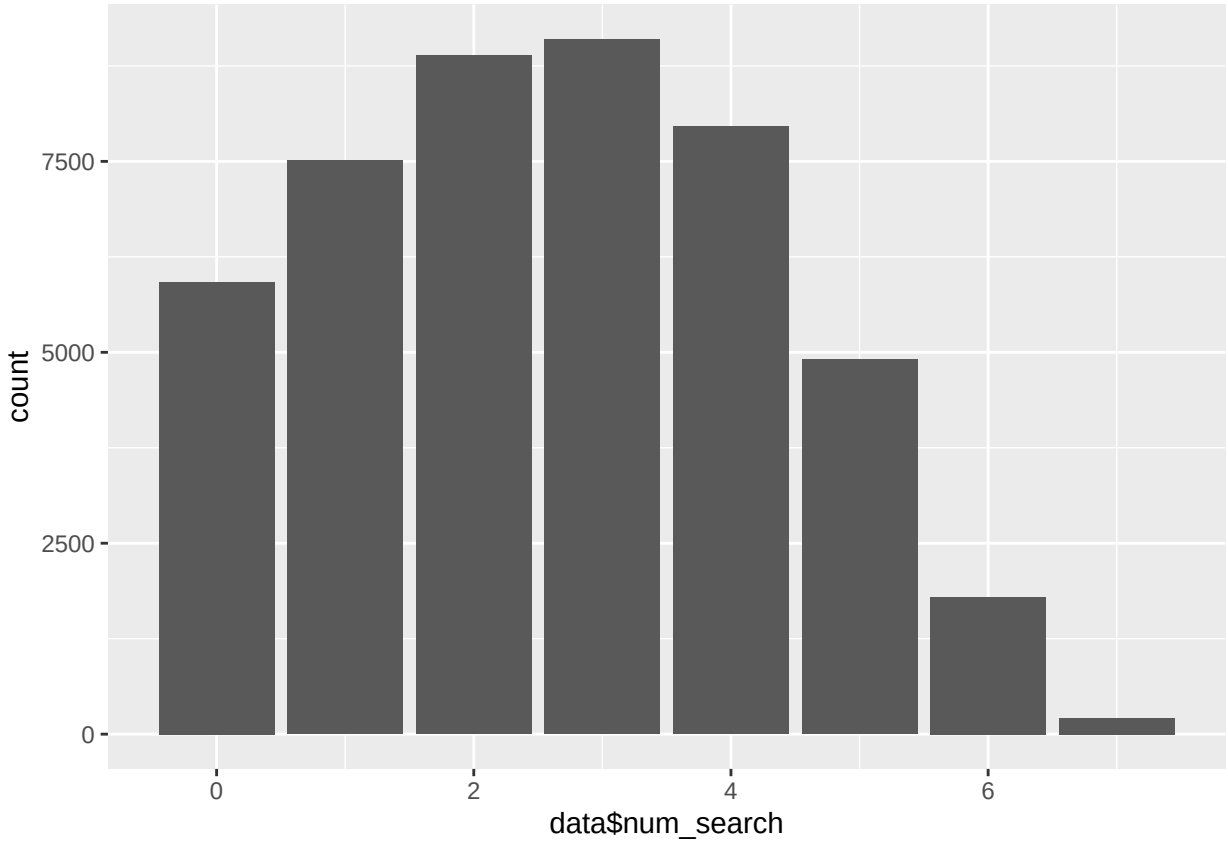

```
ggplot(data = data, aes(data$num_impressions)) +
  geom_bar()
```

```
## Warning: Use of ‘data$num_impressions‘ is discouraged. Use ‘num_impressions‘
## instead.
```




```
ggplot(data = data, aes(data$num_search)) +
  geom_bar()
```

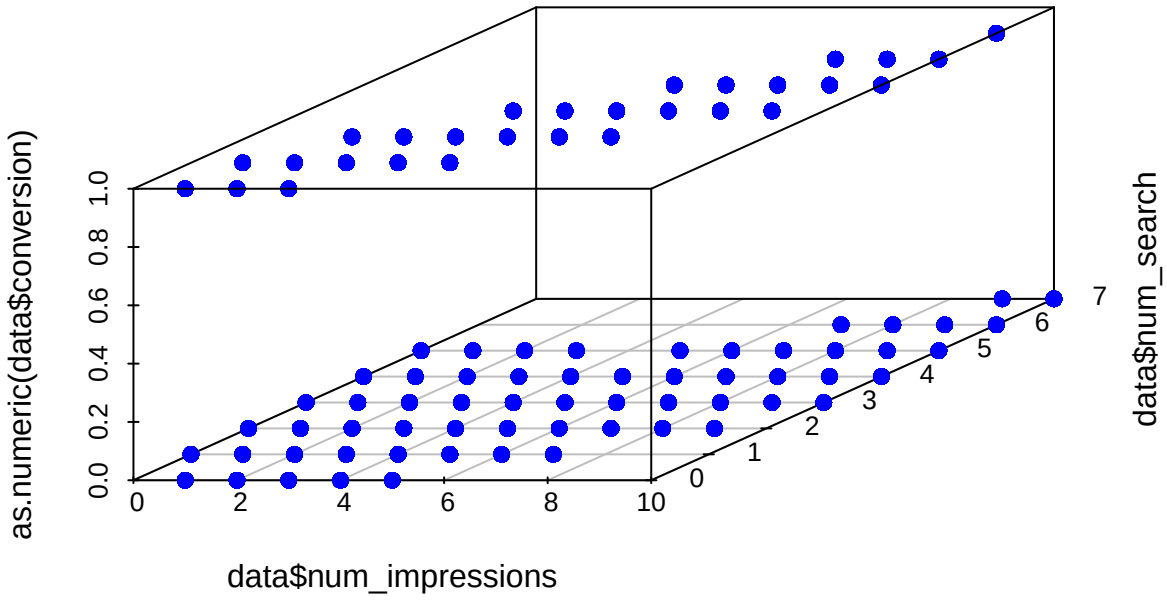
```
## Warning: Use of ‘data$num_search‘ is discouraged. Use ‘num_search‘ instead.
```




```
totconversion = data %>% group_by(conversion) %>% summarise(count = n())

howmanysessions = data %>% group_by(user_id) %>% summarise(count = n())

unique(howmanysessions$count)
```

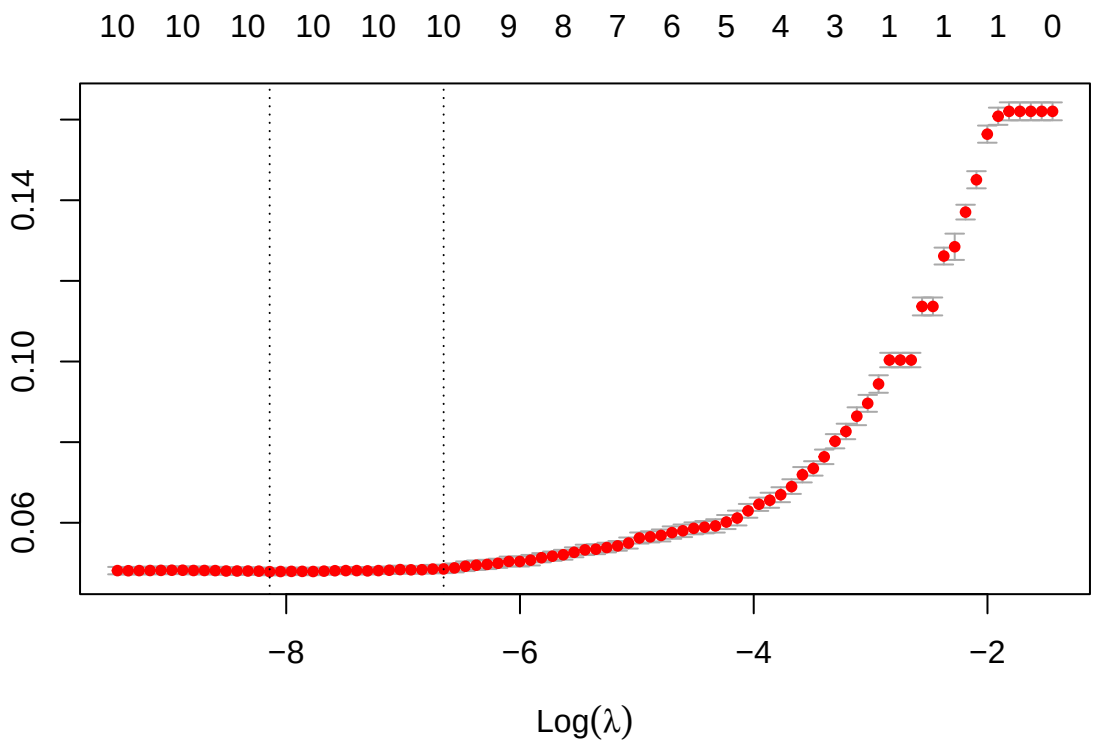
```
## [1] 3 4 5 6
```

```
scatterplot3d(data$num_impressions, data$num_search, as.numeric(data$conversion), pch = 19, color = "blu
```




```
### ADD RANKING FEATURE FOR SEVERAL FEATURES

nexttr = data %>% group_by(user_id) %>% arrange(session_dt) %>% mutate(rank = rank(session_dt,ties.metho

nexttr = nexttr %>% group_by(user_id) %>% mutate(prevrel = ifelse(rank == 1,0,ifelse(rank == 2, avg_relev

nexttr = nexttr %>% group_by(user_id) %>% mutate(prevsearch = ifelse(rank == 1,0,ifelse(rank == 2, num_s

nexttr = nexttr %>% group_by(user_id) %>% mutate(previmp = ifelse(rank == 1,0,ifelse(rank == 2, num_impr

nexttr = nexttr %>% mutate(b4thisses = (imp2sess - num_impressions))

nexttr = nexttr %>% mutate(searchb4 = (num2ses - num_search))

nexttr$prevrel = as.numeric(nexttr$prevrel)

nexttr$prevsearch = as.numeric(nexttr$prevsearch)

nexttr$previmp = as.numeric(nexttr$previmp)

### DROP FEATURES THAT ARE OF NO VALUE AND NOT BEING USED
```

```
use = nexttr[,!(colnames(nexttr)%in% c("session_id","session_dt","user_id","avgrelacross","totsearch","a

### Split into train, validation and test

train = use[use$train == TRUE,]

train = train[,!(colnames(train)%in% c("train","score","test"))]

validation = use[use$score == TRUE,]

validation = validation[,!(colnames(validation)%in% c("train","score","test"))]

test = use[use$test == TRUE,]

test = test[,!(colnames(test)%in% c("train","score","test"))]

### CREATE MATRIX FOR FEATURES AND PULL TARGET VARIABLE INTO ITS OWN DATAFRAME

x = model.matrix(conversion ~ ., train)[,-1]

y = as.factor(train$conversion)


### FIT MODEL USING CV AND L1 PENALTY

fit_ridge_cv = cv.glmnet(x, y, alpha = 1,family = "binomial",type.measure = "class")
plot(fit_ridge_cv)
```




```
coef(fit_ridge_cv,s = "lambda.min")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)           -24.6484577
## num_impressions         0.3088236
## avg_relevance           6.6932226
## num_search             -0.2201854
## session_monthJuly      -2.8521413
## session_monthNovember  -8.2045718
## session_monthOctober  -11.2655291
## session_monthSeptember  -0.1214183
## rank                    2.6773452
## imp2sess                0.8305224
## num2ses                -0.3804287
```

```
coef(fit_ridge_cv,s = "lambda.1se")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)           -22.14387870
## num_impressions         0.26603141
## avg_relevance           5.99574054
## num_search             -0.19271328
```

```
## session_monthJuly       -1.13262187
## session_monthNovember   -6.29910993
## session_monthOctober    -8.55426181
## session_monthSeptember  -0.05386694
## rank                     2.32764634
## imp2sess                 0.72063170
## num2ses                 -0.26257964
```

```
### PREDICT AND CHECK FOR ACCURACY TRAIN DATA

xtrain = model.matrix(conversion ~ ., train)[,-1]

ne = data.frame(predict(fit_ridge_cv,xtrain,type = "class", s="lambda.min"))

see = data.frame(train$conversion)

see$pred = ne$lambda.min

table(see$pred,see$train.conversion)
```

```
##
##         FALSE  TRUE
##   FALSE 25574   858
##   TRUE    643  4212
```

```
### CALCULATE PREDICTION STATISTICS

cm <- table(see$pred, see$train.conversion)

accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity * precision))/(sensitivity + precision)
specificity <- cm[1] / sum(cm[1], cm[2])

accuracy
```

```
## [1] 0.9520248
```

```
precision
```

```
## [1] 0.8675592
```

```
sensitivity
```

```
## [1] 0.8307692
```

```
fscore
```

```
## [1] 0.8487657
```

```
specificity
```

```
## [1] 0.9754739
```

```
### PREDICT AND CHECK ACCURACY

xval = model.matrix(conversion ~ ., validation)[,-1]

ne = data.frame(predict(fit_ridge_cv,xval,type = "class", s="lambda.min"))

see = data.frame(validation$conversion)

see$pred = ne$lambda.min

table(see$pred,see$validation.conversion)
```

```
##
##         FALSE  TRUE
##   FALSE 12370   416
##   TRUE    200  2014
```

```
### CALCULATE PREDICTION STATISTICS

cm <- table(see$pred, see$validation.conversion)

accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity * precision))/(sensitivity + precision)
specificity <- cm[1] / sum(cm[1], cm[2])

accuracy
```

```
## [1] 0.9589333
```

```
precision
```

```
## [1] 0.9096658
```

```
sensitivity
```

```
## [1] 0.8288066
```

```
fscore
```

```
## [1] 0.8673557
```

```
specificity
```

```
## [1] 0.9840891
```

```
### PREDICT AND CHECK ACCURACY ON TEST DATA

xtest = model.matrix(conversion ~ ., test)[,-1]

ne = data.frame(predict(fit_ridge_cv,xtest,type = "class", s="lambda.min"))

see = data.frame(test$conversion)

see$pred = ne$lambda.min

table(see$pred,see$test.conversion)
```

```
##
##         FALSE TRUE
##   FALSE  2461  416
##   TRUE    109 2014
```

```
### CALCULATE PREDICTION STATISTICS

cm <- table(see$pred, see$test.conversion)

accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity * precision))/(sensitivity + precision)
specificity <- cm[1] / sum(cm[1], cm[2])

accuracy
```

```
## [1] 0.895
```

```
precision
```

```
## [1] 0.9486576
```

```
sensitivity
```

```
## [1] 0.8288066
```

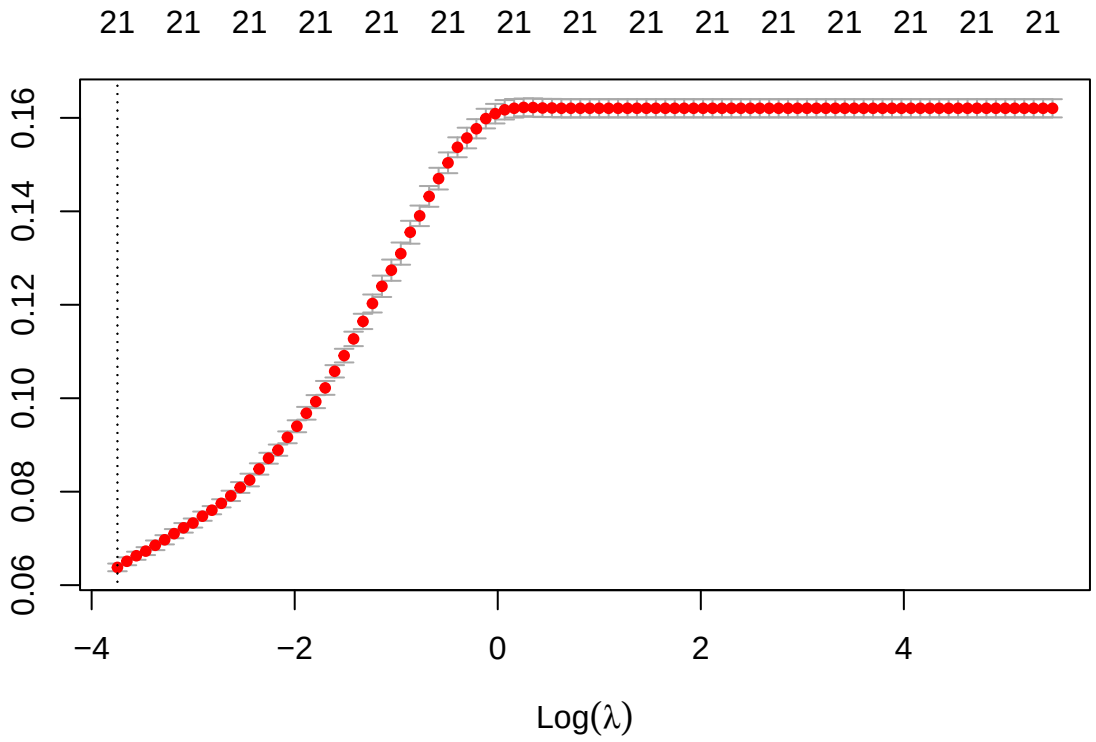
```
fscore
```

```
## [1] 0.8846914
```

```
specificity
```

```
## [1] 0.9575875
```

```
### COMPARE TO RIDGE REGRESSION
### FIT MODEL USING CV AND L2 PENALTY

use = nexttr[,!(colnames(nexttr)%in% c("session_id","session_dt","user_id"))]


### Split into train, validation and test

train = use[use$train == TRUE,]

train = train[,!(colnames(train)%in% c("train","score","test"))]

validation = use[use$score == TRUE,]

validation = validation[,!(colnames(validation)%in% c("train","score","test"))]

test = use[use$test == TRUE,]

test = test[,!(colnames(test)%in% c("train","score","test"))]


### CREATE MATRIX FOR FEATURES AND PULL TARGET VARIABLE INTO ITS OWN DATAFRAME

x = model.matrix(conversion ~ ., train)[,-1]

y = as.factor(train$conversion)

fit_ridge_cv = cv.glmnet(x, y, alpha = 0,family = "binomial",type.measure = "class")
plot(fit_ridge_cv)
```




```
coef(fit_ridge_cv,s = "lambda.min")
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)             -10.697326056
## num_impressions           0.214285500
## avg_relevance             2.708405158
## num_search               -0.027367169
## session_dayMonday        -0.004076398
## session_daySaturday       0.039117269
## session_daySunday        -0.047904140
## session_dayThursday       0.059749229
## session_dayTuesday       -0.014975387
## session_dayWednesday     -0.007331888
## session_monthJuly        -0.674876645
## session_monthNovember    -4.783354609
## session_monthOctober     -3.675890103
## session_monthSeptember   -0.042727021
## rank                      0.724582171
## imp2sess                  0.147198688
## num2ses                   0.081795072
## prevrel                   0.278245844
## prevsearch               -0.059160561
## previmp                  -0.005087706
## b4thisses                 0.125025090
## searchb4                  0.114112663
```

```
coef(fit_ridge_cv,s = "lambda.1se")
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)             -10.697326056
## num_impressions           0.214285500
## avg_relevance             2.708405158
## num_search               -0.027367169
## session_dayMonday        -0.004076398
## session_daySaturday       0.039117269
## session_daySunday        -0.047904140
## session_dayThursday       0.059749229
## session_dayTuesday       -0.014975387
## session_dayWednesday     -0.007331888
## session_monthJuly        -0.674876645
## session_monthNovember    -4.783354609
## session_monthOctober     -3.675890103
## session_monthSeptember   -0.042727021
## rank                      0.724582171
## imp2sess                  0.147198688
## num2ses                   0.081795072
## prevrel                   0.278245844
## prevsearch               -0.059160561
## previmp                  -0.005087706
## b4thisses                 0.125025090
## searchb4                  0.114112663
```

```
### PREDICT AND CHECK FOR ACCURACY TEST DATA

xtest = model.matrix(conversion ~ ., test)[,-1]

ne = data.frame(predict(fit_ridge_cv,xtest,type = "class", s="lambda.min"))

see = data.frame(test$conversion)

see$pred = ne$lambda.min

table(see$pred,see$test.conversion)
```

```
##
##         FALSE TRUE
##   FALSE  2515  726
##   TRUE     55 1704
```

```
### CALCULATE PREDICTION STATISTICS

cm <- table(see$pred, see$test.conversion)

accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity * precision))/(sensitivity + precision)
```

```r
specificity <- cm[1] / sum(cm[1], cm[2])

accuracy
```

```
## [1] 0.8438
```

```r
precision
```

```
## [1] 0.9687322
```

```r
sensitivity
```

```
## [1] 0.7012346
```

```r
fscore
```

```
## [1] 0.8135593
```

```r
specificity
```

```
## [1] 0.9785992
```