MLPS6

Hao Ran Li, Yitao Hu, Susu Zhu, Feiwen Liang, Jui-Yu Lan

08/05/2020

Preprocess Data:

```
library(dplyr)
library(tidyr)
library(tm)
library(ggplot2)
library(data.table)
library(wordcloud)
library(glmnet)
vix_data<-as.data.frame(read.csv("vix_data.csv"))
news_data<-as.data.frame(read.csv("business_insider_text_data2.csv"))
names(news_data)[1]<-"Date"
names(vix_data)[1]<-"Date"
news_data$Date<-as.Date(news_data$Date)
vix_data$Date<-as.Date(vix_data$Date)

news_data=news_data %>%group_by(Date) %>%slice(seq_len(5))


for (j in 1:508){
  for (i in 1:5){
    news_data$Headline_name[5*(j-1)+i]<-paste0("Headline_",i)
  }
}

news_data<-spread(news_data, Headline_name, Headline)

# Combine headlines into one text blob for each day and add sentence separation token
news_data$all<- paste(news_data$Headline_1,news_data$Headline_2,news_data$Headline_3,news_data$Headline

# Get rid of the special characters you see if you inspect the raw data
news_data$all <- gsub("[^0-9A-Za-z///' ]","'" , news_data$all ,ignore.case = TRUE)

# Get rid of all punctuation except headline separators, alternative to cleaning done in tm-package
news_data$all <- gsub("([<>])|[[:punct:]]", "\\1", news_data$all)

news_data<-merge(news_data,vix_data,all.x=FALSE)

assignment_data_final <- as.data.frame(news_data[, c('Date', 'all', 'label')])
```

1

```
#load the Cropus
Cropus=Corpus(VectorSource(assignment_data_final$all))
```

2

```
#remove everthing that is not in letter
Cropus = tm_map(Cropus, content_transformer(gsub), pattern = "[^a-zA-Z0-9 ]", replacement = " ")
```

```
## Warning in tm_map.SimpleCorpus(Cropus, content_transformer(gsub), pattern =
## "[^a-zA-Z0-9 ]", : transformation drops documents
```

```
#remove numbers
Cropus <- tm_map(Cropus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(Cropus, removeNumbers): transformation drops
## documents
```

```
#make everything lower case
Cropus  <- tm_map(Cropus , tolower)
```

```
## Warning in tm_map.SimpleCorpus(Cropus, tolower): transformation drops documents
```

```
#remove stop words and double white space
Cropus  <- tm_map(Cropus , removeWords, c(stopwords(kind = "SMART"), "<s>"))
```

```
## Warning in tm_map.SimpleCorpus(Cropus, removeWords, c(stopwords(kind =
## "SMART"), : transformation drops documents
```

```
Cropus  <- tm_map(Cropus , stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(Cropus, stripWhitespace): transformation drops
## documents
```

```
#stem the words
Cropus=tm_map(x=Cropus, FUN=stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(x = Cropus, FUN = stemDocument): transformation
## drops documents
```

3

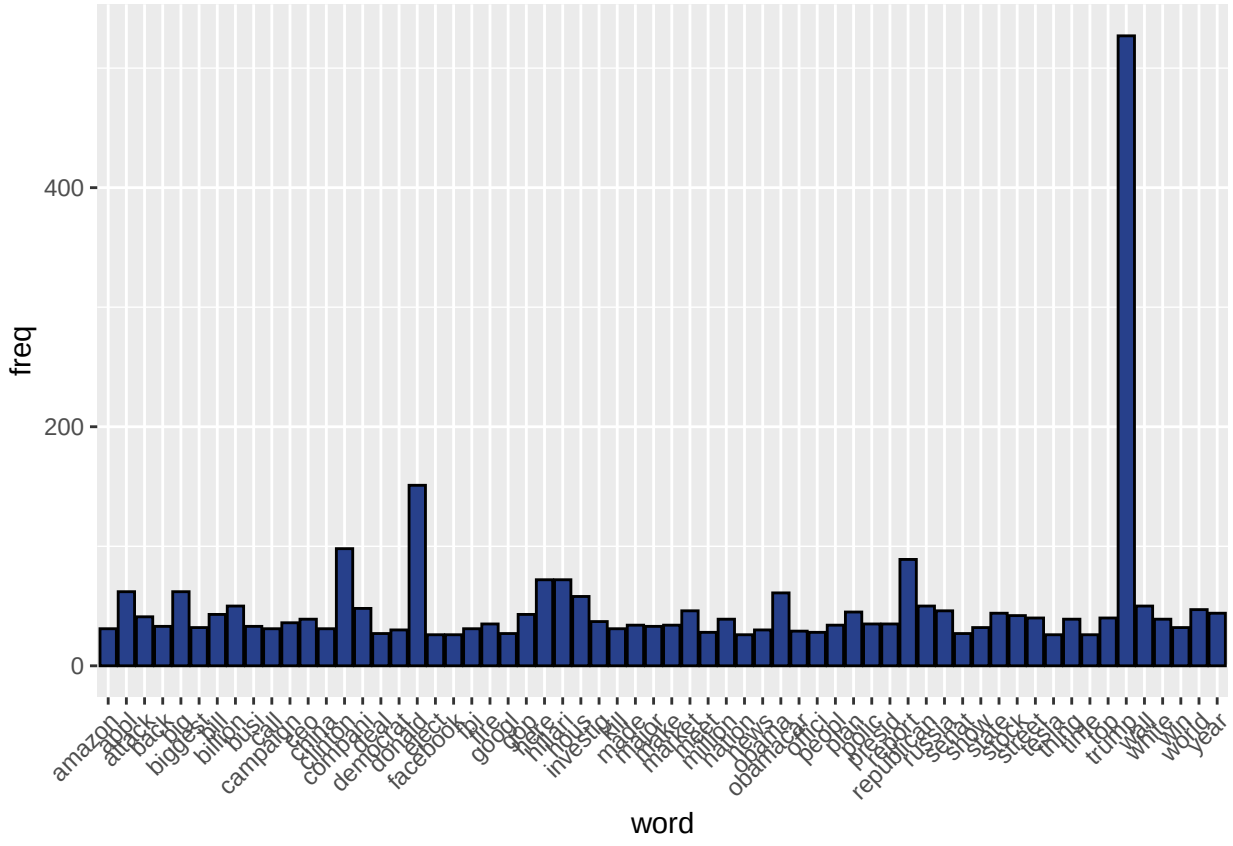
```
#create documentTermMatrix
dtm <- DocumentTermMatrix(Cropus)
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 347, terms: 3350)>>
## Non-/sparse entries: 12135/1150315
## Sparsity           : 99%
## Maximal term length: 19
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  appl big clinton donald here hillari hous obama report trump
##   104    0   0       1      0    0       0    0     0      1     1
##   223    0   0       0      1    1       0    0     0      0     2
##   257    0   0       0      0    0       0    0     0      0     5
```

```
##    258   0   0       0       0     1       0     1       0       0     4
##    282   0   0       0       0     0       0     0       0       0     1
##    283   0   0       0       0     0       0     1       0       0     3
##    287   0   0       0       0     0       0     0       0       0     3
##    290   0   0       0       0     0       0     0       0       0     2
##    327   0   0       0       0     0       0     0       0       1     4
##    330   0   0       0       0     0       0     0       0       2     2
```

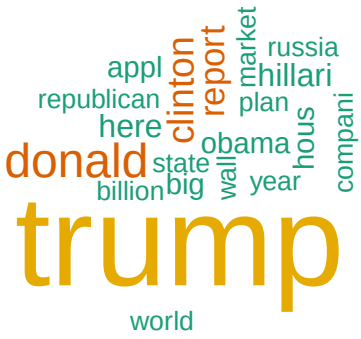
4

```
#create a freq matrix
freq_mat=colSums(as.matrix(dtm))
#make freqmat a datatable
words_freq=data.table(word=names(freq_mat),freq=freq_mat)
# Plot words that occur more than 25 times
ggplot(words_freq[freq > 25], aes(word, freq)) + geom_bar(fill = "royalblue4", color = "black", stat =
```



5

```
#create the words cloud of most frequent 20 words
wordcloud(names(freq_mat), freq_mat, max.words = 20, rot.per = 0.3, colors = brewer.pal(6,"Dark2"))
```



Based on the wordcloud above, we can conclude that the most frequent words in business insiders are politics-related, which potentially may be used to predict stock returns or volitility.

6

```
#construct the predictors and labels
y_data <- as.factor(assignment_data_final$label)
x_data <- as.matrix(dtm)
```

7

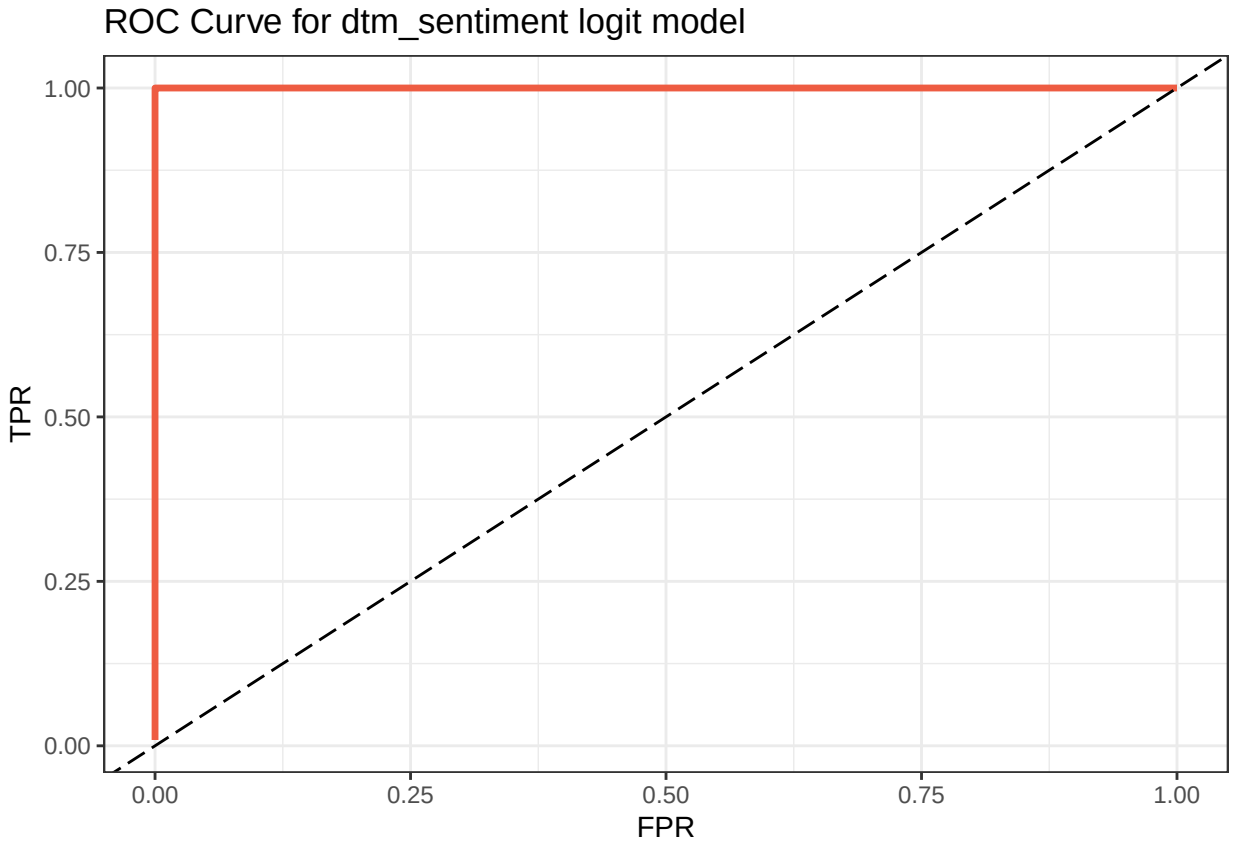
```
#training and test split
breakpoint=last(which(assignment_data_final$Date<='2016-12-31'))
Xtrain=x_data[1:breakpoint,]
ytrain=y_data[1:breakpoint]
Xtest=x_data[breakpoint:nrow(x_data),]
ytest=y_data[breakpoint:nrow(x_data)]
```

8

```
# Run logistic regression
LogModel = glm(ytrain ~ Xtrain, family='binomial')
```

```
## Warning: glm.fit: algorithm did not converge
```

```
# Compute a ROC curve
simple_roc <- function(labels, scores) {
labels <- labels[order(scores, decreasing = TRUE)]
data.frame(TPR = cumsum(labels)/sum(labels), FPR = cumsum(!labels)/sum(!labels),labels) }
#plot the in-sample roc
glm_roc <- simple_roc(ytrain == "1", LogModel$fitted.values)
TPR1 <- glm_roc$TPR
FPR1 <- glm_roc$FPR
data1 <- data.table(TPR = TPR1, FPR = FPR1)
# Plot the corresponding ROC curve
ggplot(data1, aes(x = FPR, y = TPR)) + geom_line(color = "tomato2", size = 1.2) + ggtitle("ROC Curve fo
```



From the in-sample roc curve we can see that this is clearly overfitting. Simple logistical Regression does not work here because the dimension of the features about 3000 is clearly more than the observation number which is about 300. When we have more features than observations, the model is so flexible so that it tends to overfit in sample but perform poorly out-of-sample.

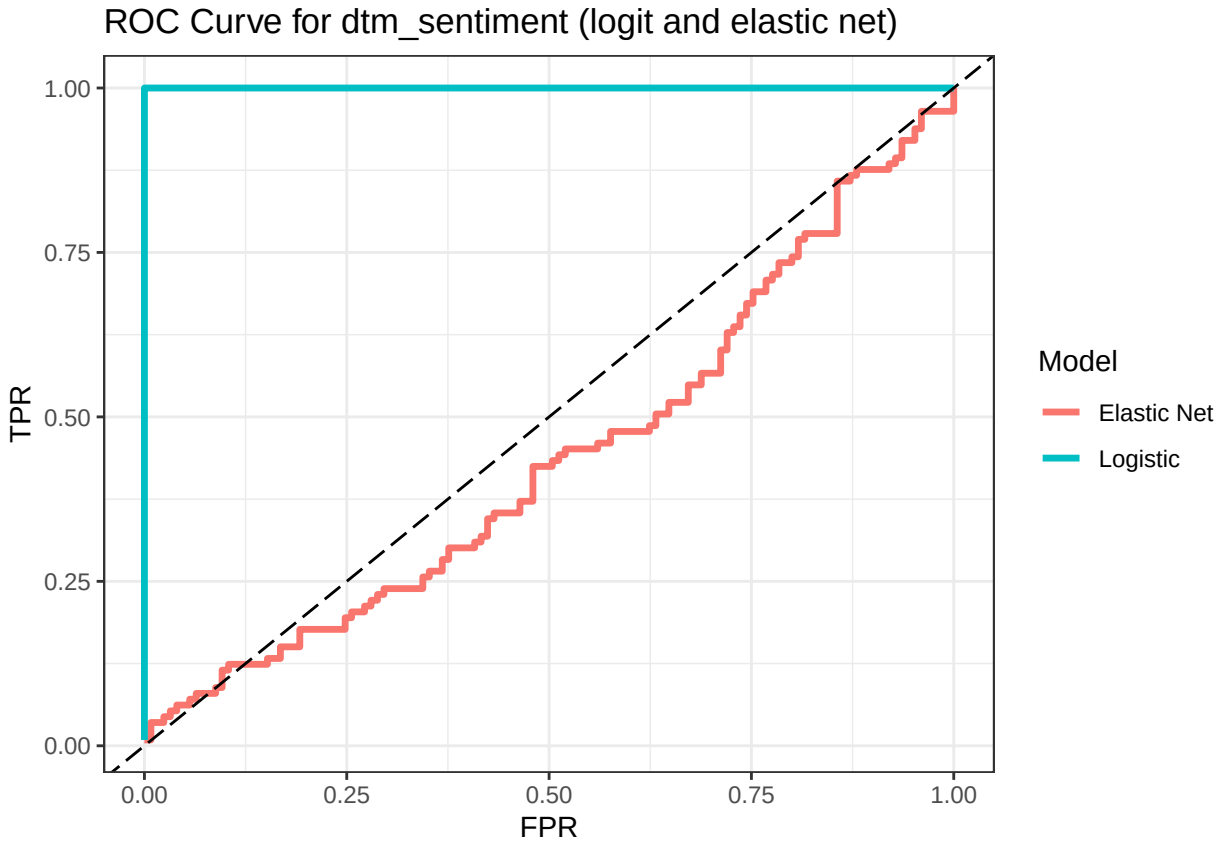
9

```
#CV the model
Regularized_Model <- cv.glmnet(x = Xtrain, y = ytrain, family = "binomial", alpha = 0.5)

# get fitted value
regfitteds <- as.numeric(predict(Regularized_Model, newx = Xtrain, type = "response", s = "lambda.min"))

glmnet_roc <- simple_roc(ytrain == "1", regfitteds)

TPR2 <- glmnet_roc$TPR
FPR2 <- glmnet_roc$FPR
data2 <- data.table(TPR = TPR2, FPR = FPR2)
data1[, `:=`(Model, "Logistic")]
data2[, `:=`(Model, "Elastic Net")]
data12 <- rbind(data1, data2)
# Plot the corresponding ROC curve
ggplot(data12, aes(x = FPR, y = TPR, color = Model)) + geom_line(size = 1.2) + ggtitle("ROC Curve for d
```



From the in-sample roc curve, we can see that the regularized model performs better than random guess within certain area of FPR, but worse than a random guess more most of the part. A elastic net kind of works because during the cross validation process, the Lasso shrinkage forces to model to drop a lot of redundant coefficients and thus avoid overfitting. However, it could also lead to the issue of under-fitting because reduce complexity of our model.

10

```
Opt.lambda=Regularized_Model$lambda.min
#refit the model
RegModel=glmnet(x = Xtrain,y = ytrain,family = 'binomial',alpha = 0.5,lambda = Opt.lambda)
idx=which(as.vector(RegModel$beta)>0)
#grab the word that has a non-negative coeff
c(colnames(Xtrain)[idx],RegModel$beta[idx])
```

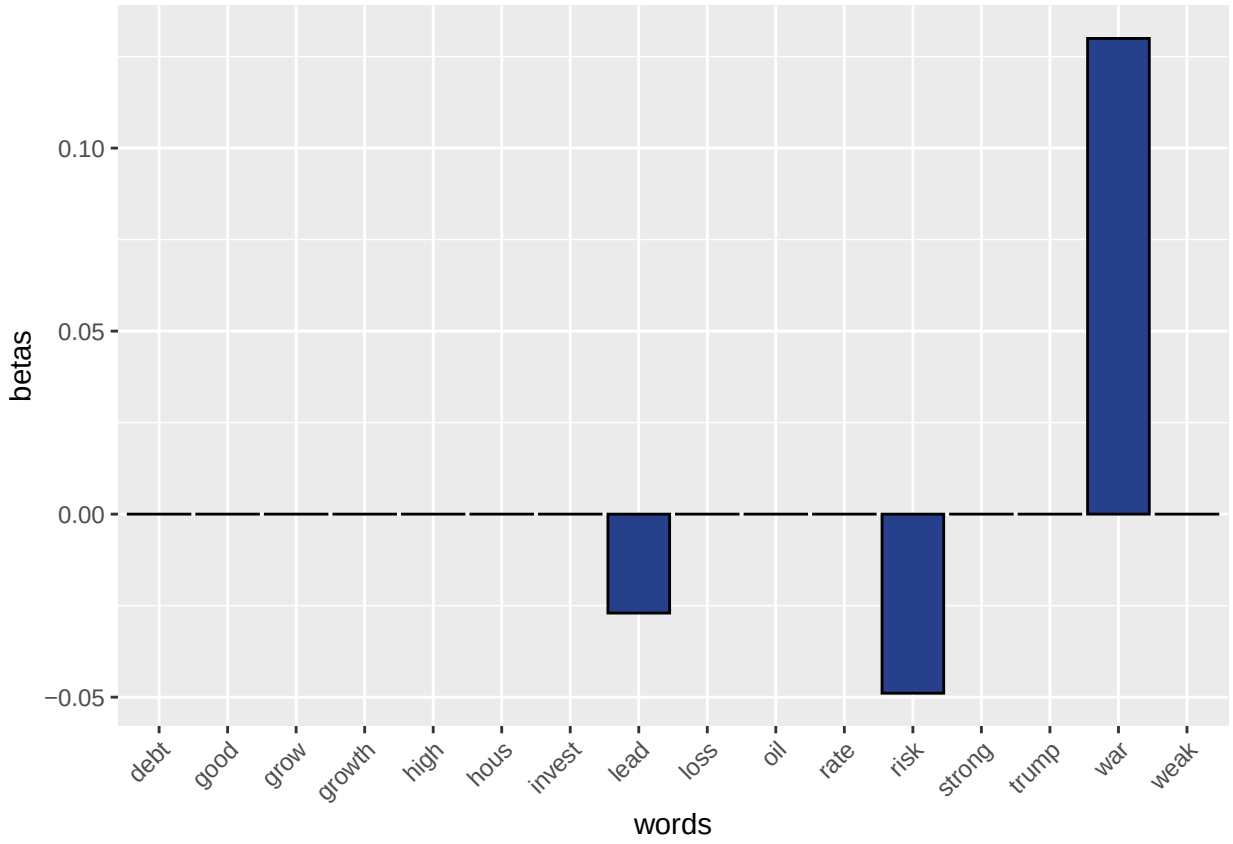
```
## [1] "cook"                  "1.91605978364568e-16"
```

The output above shows that the only word that has an non-zero coefficient is 'cook' with a coefficient of 0.178. This result is surprising, as the word itself not even been a politics-related word as we predicted before. Therefore, I would argue it might be better to choose pre-defined sentiment words instead of choosing words within the text material itself.

11

```
#redefine the features from predefined dictionary
dtm_sentiment <- dtm[,c("trump","invest","growth","grow","high","strong","lead","good","risk","debt","o
x_data_pre <- as.matrix(dtm_sentiment)
```

```
X_train=x_data_pre[1:breakpoint,]
X_test=x_data_pre[breakpoint:nrow(x_data_pre),]
#CV the model
PreCV_Model <- cv.glmnet(x = X_train, y = ytrain, family = "binomial", alpha = 0.5)
Opt.lambda=PreCV_Model$lambda.min
#refit the model
Opt_Model=glmnet(x = X_train,y = ytrain,family = 'binomial',alpha = 0.5,lambda = Opt.lambda)
coefftb=data.table(words=colnames(X_train),betas=as.vector(Opt_Model$beta))
#plot the coeff
ggplot(coefftb, aes(words, betas)) + geom_bar(fill = "royalblue4", color = "black", stat = "identity") +
```



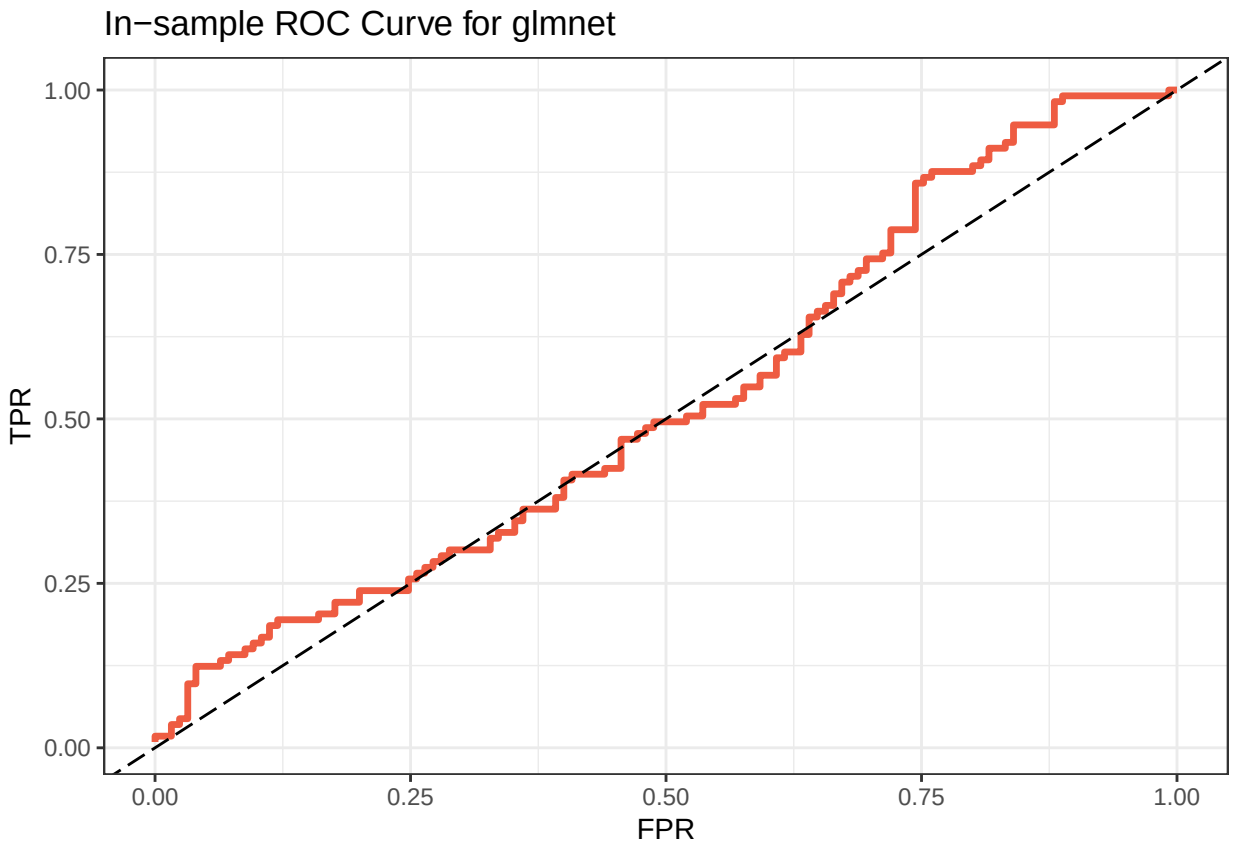
When using pre-defined dictionaries, we have five non-zero coefficients after cross-validation process, which indicates cross-validation process cannot replace a experieneced model specification.

12

```
# get fitted value
Preoptfitteds <- as.numeric(predict(PreCV_Model, newx = X_train, type = "response", s = "lambda.min"))

glmnet_roc <- simple_roc(ytrain == "1", Preoptfitteds)

TPR2 <- glmnet_roc$TPR
FPR2 <- glmnet_roc$FPR
dataopt <- data.table(TPR = TPR2, FPR = FPR2)
# Plot the corresponding ROC curve
ggplot(dataopt, aes(x = FPR, y = TPR)) + geom_line(color = "tomato2", size = 1.2) + ggtitle("In-sample
```



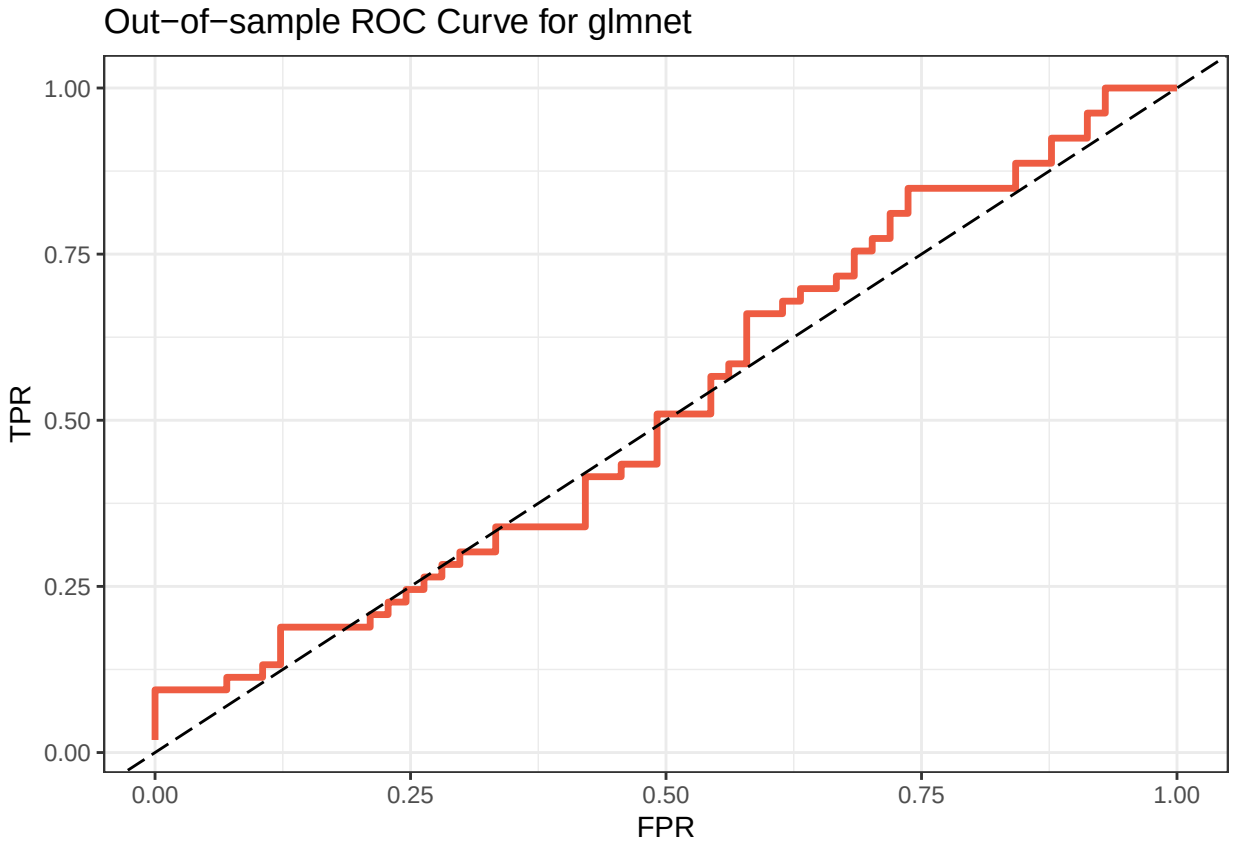
From the ROC curve above, we can conclue that the regularized model with predefined dictionary performs slightly better than random guess in sample, but the increment is not significant.

13

```
# get fitted value
Preoptfitteds <- as.numeric(predict(PreCV_Model, newx = X_test, type = "response", s = "lambda.min"))

glmnet_roc <- simple_roc(ytest == "1", Preoptfitteds)

TPR2 <- glmnet_roc$TPR
FPR2 <- glmnet_roc$FPR
dataopt <- data.table(TPR = TPR2, FPR = FPR2)
# Plot the corresponding ROC curve
ggplot(dataopt, aes(x = FPR, y = TPR)) + geom_line(color = "tomato2", size = 1.2) + ggtitle("Out-of-samp
```




```
#choose 0.5 as the cutoff prob
Accuracy=sum(ifelse(Preoptfitteds>0.5,1,0)==ytest)/length(ytest)
Accuracy
```

```
## [1] 0.5545455
```

When perform prediction on the test set, our model has an accuracy of 0.57, which is not significantly better than a 50/50 guess. This fact can also be verified by the out of sample ROC curve. I would argue this result is due to small sample and simlple model structure of logistical regression. If we can collect a large sample and train a Neural Network or Recurrent Model, the performance might be much better.