Problem Set 6 Solutions

Denis Mokanov

May 20, 2019

Creating a sentiment index from text data

Before you start, please download the script "`CleanUpScript_PS6.R`" from CCLE. Run it on the data to do some pre-pre-processing so we're all starting from the same data set.

```
# Clean workspace
rm(list = ls())

# Load packages
library(data.table)
library(tm)
library(glmnet)
library(ggplot2)
library(pROC)
library(SnowballC)
library(dplyr)
library(wordcloud)
library(ROCR)
library(stargazer)
library(sandwich)
library(lmtest)

# Load data
data <- read.csv("DJIA_Headline_News.csv", stringsAsFactors = FALSE)



# Run pre-processing code from CleanUpScript_PS5.R Make 'Date' column a Date
# object to make train/test splitting easier
data$Date <- as.Date(data$Date)

## Combine headlines into one text blob for each day and add sentence
## separation token
data$all <- paste(data$Top1, data$Top2, data$Top3, data$Top4, data$Top5, data$Top6,
    data$Top7, data$Top8, data$Top9, data$Top10, data$Top11, data$Top12, data$Top13,
    data$Top14, data$Top15, data$Top16, data$Top17, data$Top18, data$Top19,
    data$Top20, data$Top21, data$Top22, data$Top23, data$Top24, data$Top25,
    sep = " <s> ")

## Get rid of those pesky b's and backslashes you see if you inspect the raw
## data
data$all <- gsub("b\"|b'|\\\\|\\\"", "", data$all)

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

```
## Reduce to only the three columns we need.
data <- data[, c("Date", "Label", "all")]
```

1. Use `Corpus(VectorSource(data))` to load the corpus. VectorSource makes each line a document, so now each document corresponds to a different date in the dataset.

```
Corpus <- Corpus(VectorSource(data$all))
```

2. Pre-process the data as in the lecture notes. Feel free to use the code from Code Snippets Topic 7 on CCLE. That is, remove numbers, make all lower case, remove stopwords, stemming, etc.

```
# Cleaning procedure
Corpus = tm_map(Corpus, removePunctuation)
Corpus = tm_map(Corpus, content_transformer(gsub), pattern = "\t", replacement = " ")
Corpus = tm_map(Corpus, content_transformer(gsub), pattern = "[^a-zA-Z0-9 ]",
    replacement = " ")
Corpus <- tm_map(Corpus, removeNumbers)
Corpus <- tm_map(Corpus, tolower)
Corpus <- tm_map(Corpus, removeWords, c(stopwords(kind = "SMART"), "<s>"))
Corpus <- tm_map(Corpus, stripWhitespace)
```

3. As in the lecture note, create a `DocumentTermMatrix`, call it `dtm`. Run the line `inspect(dtm[5:10, 801:810])`. Notice that the matrix is quite sparse (a lot of zeros).

```
control <- list(
    removeNumbers = TRUE,
    tolower = TRUE,
    stopwords = c(stopwords(kind = 'SMART'), '<s>'), # exclude stopwords and headline tokens
    stemming = TRUE
)

dtm <- Corpus %>% DocumentTermMatrix(control=control)
inspect(dtm[5:10, 801:810])
```

```
## <<DocumentTermMatrix (documents: 6, terms: 10)>>
## Non-/sparse entries: 0/60
## Sparsity           : 100%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs americanseuropean americansnn americansourc americanstyl
##   5                  0           0             0            0
##   6                  0           0             0            0
##   7                  0           0             0            0
##   8                  0           0             0            0
##   9                  0           0             0            0
##   10                 0           0             0            0
```

```
##     Terms
## Docs americantrain americanturkish americath amerium ami amia
##   5              0                0         0       0   0    0
##   6              0                0         0       0   0    0
##   7              0                0         0       0   0    0
##   8              0                0         0       0   0    0
##   9              0                0         0       0   0    0
##   10             0                0         0       0   0    0
```

4. As in the lecture note, create a freq matrix as the column sums of dtm. Show in a bar plot the frequency of words that occur more than 1000 times.

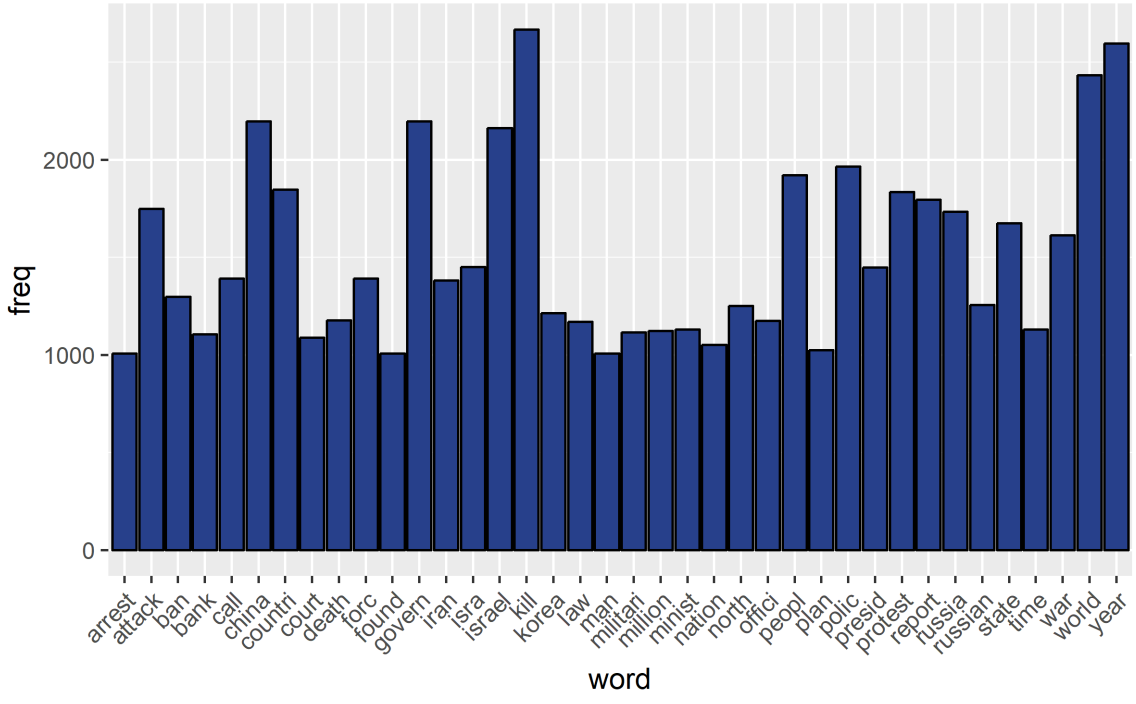
```
# Organize words by frequency
freq <- colSums(as.matrix(dtm))
ord_corpus <- order(freq)

# See most common words
freq[tail(ord_corpus)]
```

```
## israel  china govern  world   year   kill
##   2161   2195   2197   2432   2594   2666
```

```
# Identify words that appear frequently
word_freq <- data.table(word = names(freq), freq = freq)

# Plot most frequent words along with frequency
ggplot(word_freq[freq > 1000], aes(word, freq)) + geom_bar(fill = "royalblue4",
    color = "black", stat = "identity") + theme(axis.text.x = element_text(angle = 45,
    hjust = 1))
```



5. Create a wordcloud of the 100 most frequent words. Based on this (and 4.), how would you characterize the typical headline in terms of the news subject (economic, entertainment, domestic affairs, foreign affairs, etc.)? Are there words that, intuitively, can matter for the stock market returns that day?

```
# Plot 100 most frequent words, and add some color
wordcloud(names(freq), freq, max.words = 100, rot.per = 0.3, colors = brewer.pal(6,
    "Dark2"))
```

From the most frequent words (`police, governmnet, israel, people, china,...`) we can assume that the typical headline is mostlikely about domestic and foreign affairs. The words that could matter for the stock market could be: `growth, invest, debt, war,oil, risk, etc`

6. Create the data `y_data <- as.factor(data$Label)` and `x_data <- as.matrix(dtm)` .You will try to construct an index based on the words in dtm that predicts the direction of stock returns.

```
# Create full data set
y_data <- as.factor(data$Label)
x_data <- as.matrix(dtm)
data_full <- data
```

7. Split the data into a training data-set, based on data up to and including `2014-12-31`. The remaining data should be used for actual out-of-sample testing.

```
# Create training data set
split_index <- data$Date <= as.Date("2014-12-31")
data_train <- data_full[split_index, ]
```

```
y_train <- as.factor(data$Label[split_index])

# Create out of sample test data set
data_test <- data_full[!split_index, ]
y_test <- as.factor(data$Label[!split_index])
```

8. The text data is very noisy. You will create a sentiment index based on words you think are likely to matter for the stock market. So that we all are considering the same words, define a new dtm-variable as follows:

```
sentiment_words <- c("invest", "growth", "grow", "high", "strong", "lead", "bankrupt",
    "good", "bull", "bear", "interest", "market", "hous", "rate", "oil", "loss",
    "weak", "low", "fear", "poor", "risk", "stock", "debt", "financi", "fiscal",
    "reserv", "crash", "war", "recess")
dtm_sentiment <- dtm[, sentiment_words]
x_data_sent <- as.matrix(dtm_sentiment)
```

Note how I express words in their stemmed form. What is the overall frequency of these words in the database? How do these frequency numbers compare to the frequencies of the words you plotted in sub-question 4.?

```
setkey(word_freq, freq)
word_freq[word %in% sentiment_words]
```

```
##         word freq
##  1:     bull   12
##  2:   fiscal   12
##  3: bankrupt   23
##  4:     weak   30
##  5:   recess   78
##  6:   growth   88
##  7:     bear   93
##  8:    stock  111
##  9:     loss  113
## 10:   strong  113
## 11:      low  118
## 12:   reserv  120
## 13:   invest  128
## 14: interest  193
## 15:     poor  205
## 16:     debt  213
## 17:     risk  232
## 18:     good  238
## 19:    crash  252
## 20:  financi  254
## 21:   market  257
## 22:     rate  298
## 23:     lead  351
## 24:     grow  353
## 25:     hous  366
## 26:     fear  423
```

```
## 27:     high  436
## 28:      oil  857
## 29:      war 1613
##         word freq
```

The only word with frequency > 1000 (that also appeard in question 4) is `war`. The other most frequent word is `oil` (freq $= 857$), `high` (freq $= 436$), `fear` (freq $= 423$), ...

9. Fit a standard logistic regression to the `y_data` using the data in `dtm_sentiment`. Report the results. Are any words signficiant? Which seem to be the most important?

```
glm.fit <- glm(y_data ~ x_data_sent, family = "binomial")
summary(glm.fit)
```

```
##
## Call:
## glm(formula = y_data ~ x_data_sent, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6573  -1.2365   0.9617   1.0925   1.8276
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.229378   0.089433   2.565 0.010323 *
## x_data_sentinvest    -0.180426   0.172342  -1.047 0.295143
## x_data_sentgrowth     0.256280   0.211696   1.211 0.226046
## x_data_sentgrow      -0.032557   0.109809  -0.296 0.766855
## x_data_senthigh       0.099427   0.097311   1.022 0.306905
## x_data_sentstrong    -0.240696   0.190047  -1.267 0.205331
## x_data_sentlead       0.022544   0.105989   0.213 0.831561
## x_data_sentbankrupt -0.561593   0.437678  -1.283 0.199451
## x_data_sentgood      -0.023833   0.125785  -0.189 0.849724
## x_data_sentbull      -0.172888   0.588551  -0.294 0.768948
## x_data_sentbear       0.301502   0.197119   1.530 0.126129
## x_data_sentinterest -0.223533   0.146046  -1.531 0.125877
## x_data_sentmarket     0.026550   0.129133   0.206 0.837100
## x_data_senthous      -0.026278   0.103165  -0.255 0.798942
## x_data_sentrate       0.112261   0.109119   1.029 0.303577
## x_data_sentoil       -0.105986   0.057302  -1.850 0.064372 .
## x_data_sentloss      -0.036842   0.184135  -0.200 0.841418
## x_data_sentweak      -0.093859   0.364760  -0.257 0.796935
## x_data_sentlow       -0.633616   0.190324  -3.329 0.000871 ***
## x_data_sentfear       0.142049   0.099881   1.422 0.154973
## x_data_sentpoor      -0.066665   0.135019  -0.494 0.621488
## x_data_sentrisk       0.007246   0.128298   0.056 0.954963
## x_data_sentstock     -0.467565   0.185209  -2.525 0.011585 *
## x_data_sentdebt       0.195661   0.125995   1.553 0.120442
## x_data_sentfinanci    0.040153   0.121906   0.329 0.741871
## x_data_sentfiscal     0.055839   0.552284   0.101 0.919467
## x_data_sentreserv    -0.124275   0.175667  -0.707 0.479290
## x_data_sentcrash     -0.095836   0.109599  -0.874 0.381885
## x_data_sentwar       -0.011991   0.043675  -0.275 0.783659
```

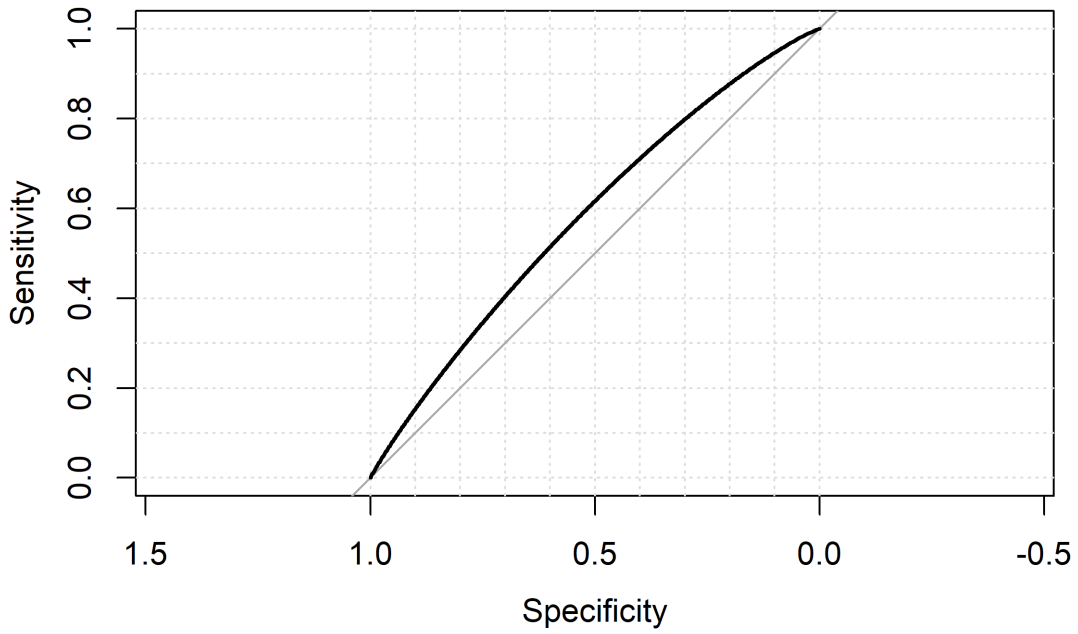
```
## x_data_sentrecess    0.042055   0.217861   0.193 0.846932
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2747.3  on 1988  degrees of freedom
## Residual deviance: 2704.6  on 1959  degrees of freedom
## AIC: 2764.6
##
## Number of Fisher Scoring iterations: 4
```

```
preds_logit <- as.numeric(predict(glm.fit, type = "response"))
```

Only the intercept and the coefficients on `low` and `stock` have t-stats > 2.

10. Create the ROC curve for this model. Is it better than random? You likely want to use the predict function to get the model prediction.

```
ROC_enet <- roc(response = y_data, predictor = preds_logit, smooth = TRUE, plot = TRUE,
    grid = TRUE)
```

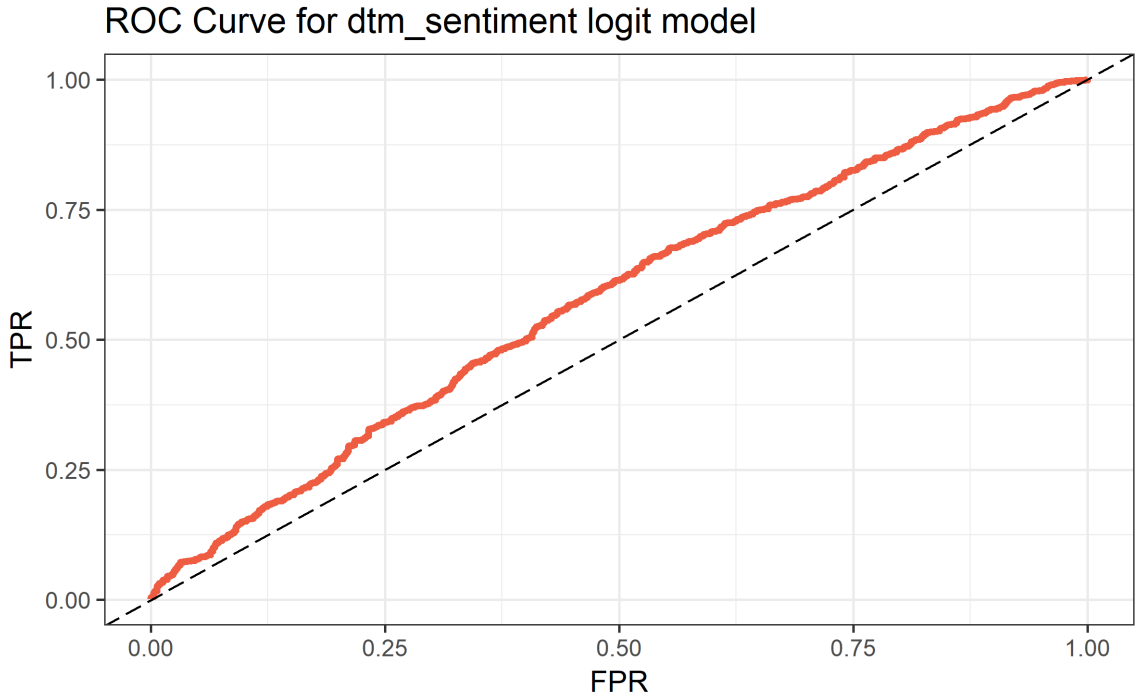



```
# Compute a ROC curve
simple_roc <- function(labels, scores) {
    labels <- labels[order(scores, decreasing = TRUE)]
    data.frame(TPR = cumsum(labels)/sum(labels), FPR = cumsum(!labels)/sum(!labels),
        labels)
}

glm_roc <- simple_roc(y_data == "1", preds_logit)
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 for dtm_sentiment logit model") + geom_abline(slope = 1,
    intercept = 0, linetype = "longdash") + theme_bw()
```



As we can see the model is better than random (the 45-degree line).

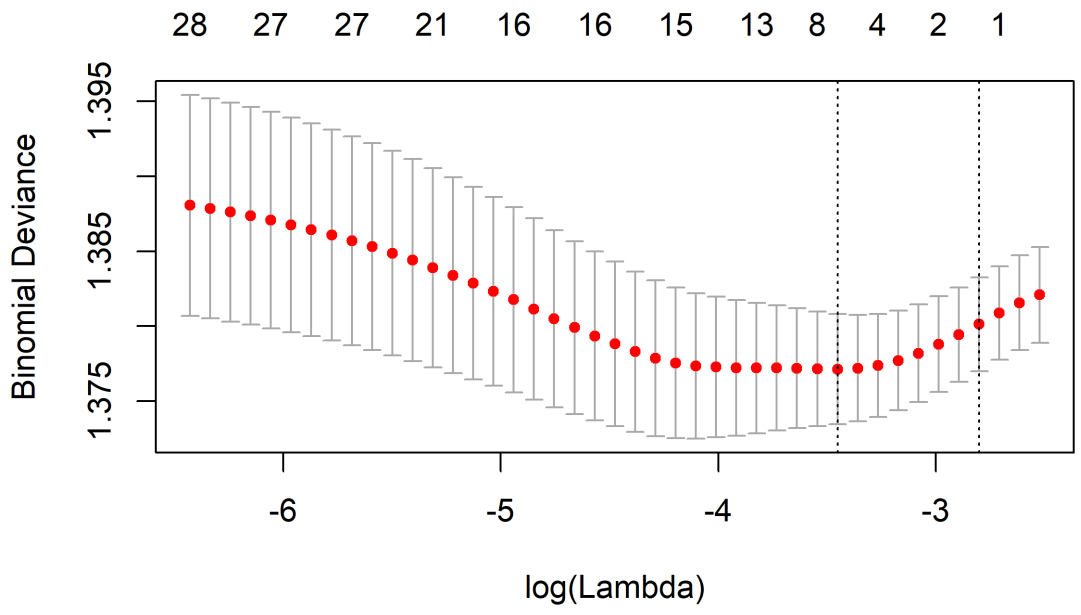
11. Now, fit an elastic net model with `alpha = 0.5` using cross-validation. First, give the objective function of such a model (using the notes). Report the results by plotting the fitted coefficients and the cross-validation curve as in the lecture notes. Also, which words are the most important?

From lecture notes, the objective function for the Elastic net estimator is the mean-squared error (MSE) which we try to minimize:

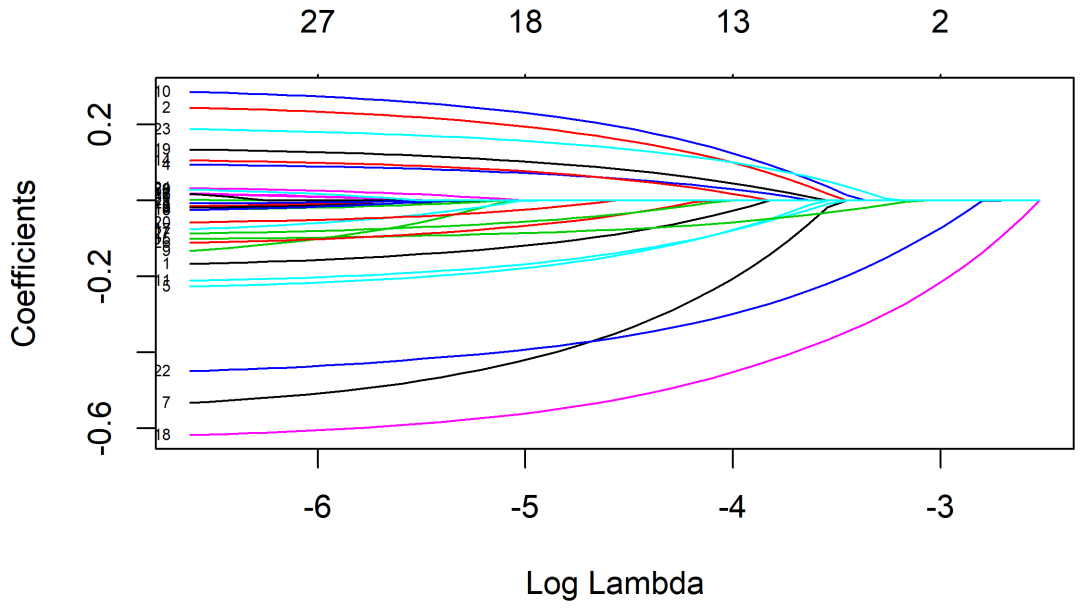
$$\min_{\beta} \sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{K} \beta_k X_{ki} \right)^2 \quad \text{s.t.} \quad \sum_{k=1}^{K} (1-\alpha)\beta_k^2 + \alpha|\beta_k| \leq B$$

where K is number of X variables on the right-hand side and N is the number of data points.

```
glmnet.fit <- cv.glmnet(x = x_data_sent, y = y_data, family = "binomial", alpha = 0.5)
plot.cv.glmnet(glmnet.fit)
```



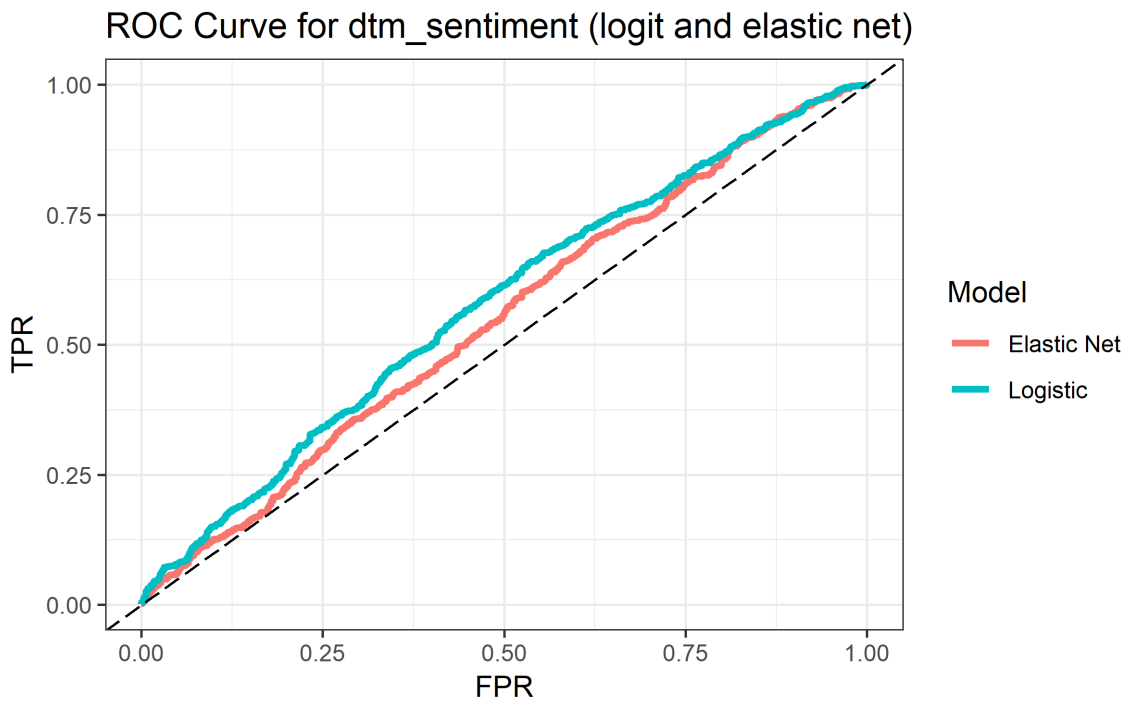

```
plot.glmnet(glmnet.fit$glmnet.fit, "lambda", label = TRUE)
```



12. Create the ROC curve for this model. Is it better than random? Is it better than that in 10.?

```r
preds_enet <- as.numeric(predict(glmnet.fit, newx = x_data_sent, type = "response",
    s = "lambda.min"))
glmnet_roc <- simple_roc(y_data == "1", preds_enet)
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 dtm_sentiment (logit and elastic net)") + geom_abline(slope = 1,
    intercept = 0, linetype = "longdash") + theme_bw()
```



From the area under the ROC curve we see that the elastic net model performs slightly worse than the logistic model in 10 but better than the random model (45-degree line).

13. Now, using the *test sample* and your two fitted models, what is the proportion of days your model would have made the right prediction in this new sample? Report this for both models. Are they better than random (50/50)? Which model is best?

```r
# Split data
x_train <- x_data_sent[1:length(y_train), ]
x_test <- x_data_sent[(length(y_train) + 1):length(data_full$Date), ]

# Test logisitic model
logit.fit_train <- glm(y_train ~ x_train, family = "binomial")
preds_logit_test <- predict(logit.fit_train, type = "response", new = data.frame(x_test))
```

```
preds_logit_test <- preds_logit_test[1:length(y_test)]
prop_correct_logit <- sum(round(preds_logit_test) == y_test)/length(y_test)  # Proportion correct of ou
prop_correct_logit
```

```
## [1] 0.5291005
```

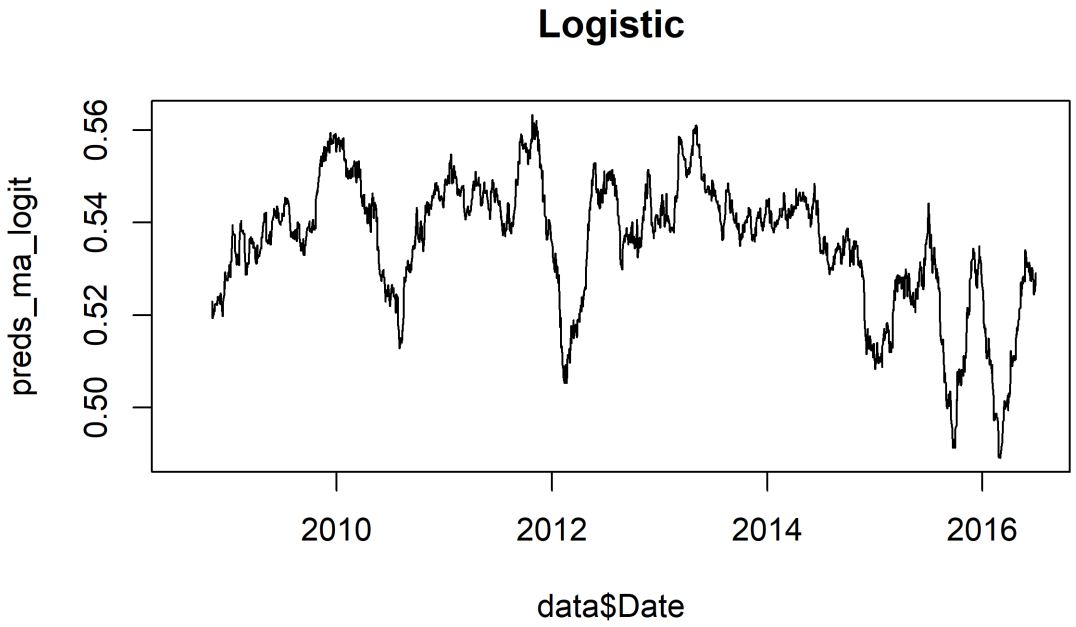
```
# Test elastic net model
enet.fit_train <- cv.glmnet(x = x_train, y = y_train, family = "binomial", alpha = 0.5)
preds_enet_test <- predict(enet.fit_train, newx = x_test, type = "response",
    s = "lambda.1se")
prop_correct_enet <- sum(round(preds_enet_test) == y_test)/length(y_test)  # Proportion correct of out-
prop_correct_enet
```

```
## [1] 0.5079365
```

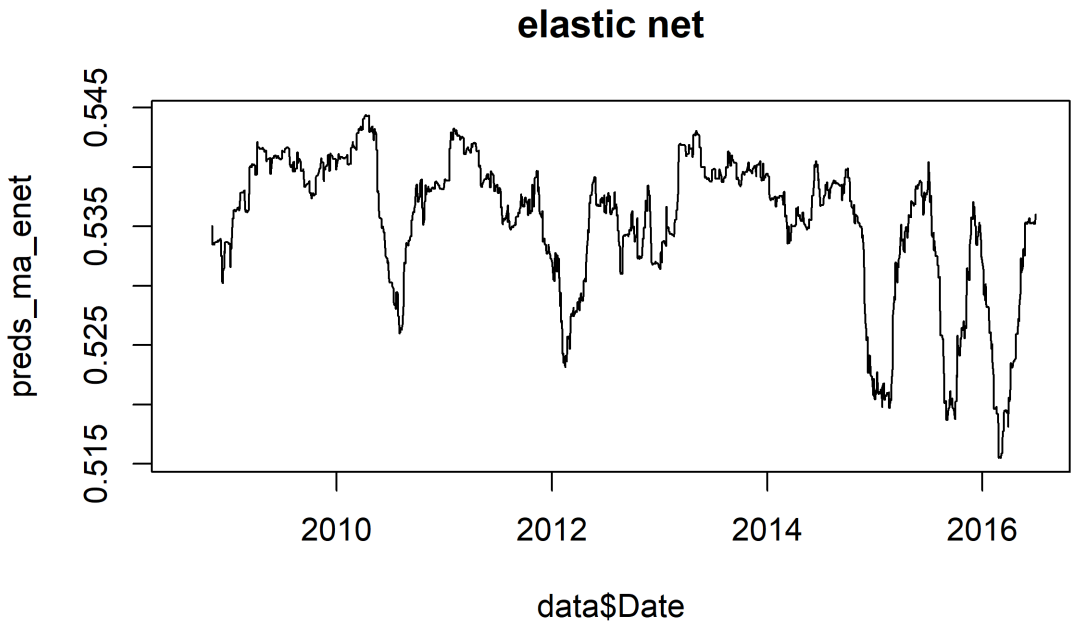
The logistic model performs better than the elastic net model out-of-sample. Both perform better than random a random model (the proportions of correct predictions are above 50% for both models).

14. Finally, create, in the training sample, 63 day moving average of the prediction (`pred_ma`) from both models, as well as the y-variable (`y_ma`). Plot `pred_me` versus date. Does the sentiment index behave reasonably? Run a standard ols of `y_ma` on `pred_ma`. What is the R^2 of each model? Are the sentiment scores and stock returns significantly positively related, as the model would predict at this lower frequency?

```
# Prepare data and plot
f63 <- rep(1/63, 63)
preds_ma_logit <- stats::filter(preds_logit, f63, sides = 1)
preds_ma_enet <- stats::filter(preds_enet, f63, sides = 1)
y_ma <- stats::filter(y_data, f63, sides = 1)
plot(data$Date, preds_ma_logit, type = "l", main = "Logistic")
```



```r
plot(data$Date, preds_ma_enet, type = "l", main = "elastic net")
```




```r
# Define Newey-West lag
NW_lag = 90

# Comparison for logistic
reg1 <- lm(as.numeric(y_data[-(1:62)]) ~ preds_logit[-(1:62)])
reg2 <- lm(as.numeric(y_ma) ~ preds_ma_logit)
stargazer(reg1, reg2, coeftest(reg2, NeweyWest(reg2, lag = NW_lag)), type = "text",
    column.labels = c("OLS", "MA", "MA with NW SEs"), dep.var.labels.include = F)
```

```
##
## ===================================================================
##                                       Dependent variable:
##                                  ---------------------------------
##                                     OLS       OLS      coefficient
##                                                            test
##                                     OLS        MA     MA with NW SEs
##                                     (1)       (2)          (3)
## -------------------------------------------------------------------
## preds_logit[-(1:62)]              1.008***
##                                    (0.155)
##
## preds_ma_logit                               1.487***     1.487**
##                                              (0.094)      (0.710)
##
## Constant                          0.998***   0.740***      0.740*
##                                    (0.084)   (0.051)      (0.378)
##
## -------------------------------------------------------------------
## Observations                        1,927     1,927
```

```
## R2                                0.021     0.114
## Adjusted R2                       0.021     0.114
## Residual Std. Error (df = 1925)   0.493     0.058
## F Statistic (df = 1; 1925)      42.118*** 248.877***
## ============================================================
## Note:                                  *p<0.1; **p<0.05; ***p<0.01
```

```
# Comparison for elastic net
reg3 <- lm(as.numeric(y_data[-(1:62)]) ~ preds_enet[-(1:62)])
reg4 <- lm(as.numeric(y_ma) ~ preds_ma_enet)
stargazer(reg3, reg4, coeftest(reg4, NeweyWest(reg4, lag = NW_lag)), type = "text",
    column.labels = c("OLS", "MA", "MA with NW SEs"), dep.var.labels.include = F)
```

```
##
## ============================================================
##                                      Dependent variable:
##                                 ---------------------------
##                                    OLS       OLS      coefficient
##                                                           test
##                                    OLS        MA     MA with NW SEs
##                                    (1)       (2)          (3)
## ------------------------------------------------------------
## preds_enet[-(1:62)]              2.012***
##                                   (0.416)
##
## preds_ma_enet                                3.731***      3.731
##                                              (0.222)      (2.306)
##
## Constant                          0.461**  -0.461***      -0.461
##                                   (0.223)   (0.119)      (1.235)
##
## ------------------------------------------------------------
## Observations                       1,927     1,927
## R2                                 0.012     0.127
## Adjusted R2                        0.012     0.127
## Residual Std. Error (df = 1925)    0.496     0.058
## F Statistic (df = 1; 1925)       23.415*** 281.220***
## ============================================================
## Note:                                  *p<0.1; **p<0.05; ***p<0.01
```

Taking a moving average reveals more low-frequent fluctuations. For both models, the R2 's are much higher when looking at lower frequency co-movement. Thus, even though the daily sentiment index has low forecasting power, there seems to be a clear relation between sentiment and stock returns. Note that when using the 63-day (1 quarter) moving average, we are really looking mainly at a contemporaneous relation, so this is not about forecasting. Note that the t-statistics on the moving average predictions are really high (>15). Partly this is due to the fact that we have not controlling for the positive autocorrelation in the error terms of the regressions, induced by the moving average specification (63 day data, regression uses daily time increments so there are 62 days overlap). When we address this by using Newey-West standard errors, we see that the relationship becomes weaker.