CS410 Final Project (Fall 2022)

Sam Kirby (slkirby2@illinois.edu)

2022-11-25

```
knitr::opts_chunk$set(echo = TRUE)
library(knitr)
library(dplyr)
library(tidytext)
library(SentimentAnalysis)
library(rvest)
library(yahoofinancer)
library(xml2)
library(pdftools)
library(stringr)
library(caret)
library(expss)
library(pander)
```

Source file acquisition and text processing

The first step is to locate and download mutual fund quarterly commentary files using targeted Google search queries. An example Google search term would be:

"mutual fund" AND ("September 30, 2022" | "Sept. 30, 2022" | "9/30/2022" | "3Q 2022" | "3Q2022" | "September 30 2022") AND "quarter" AND "performance" AND "contributors" AND "detractors" AND "outlook" AND "before investing" AND "past performance" filetype:pdf

Because this step was only completed once for this project, I used a quasi-manual process where individual pages of Google search results are combined into a single HTML document, which is then parsed and a list of URLs extracted with the script below. I also supplemented the files obtained autonomously with additional commentary files for a sample of fund managers whose commentaries do not appear within Google search results.

Reference: https://stackoverflow.com/questions/3746256/extract-links-from-webpage-using-r

Extract and filter links from source HTML

```
pg = read_html("sample.htm")
links = html_attr(html_nodes(pg, "a"), "href")
results = as.data.frame(links)

# remove duplicates
results = as.data.frame(results[!duplicated(results),])
```

```
# remove rows beginning with: https://www.google.com
results = results[!grepl("https://www.google.com", results[,1]),]

results = as.data.frame(results)
colnames(results) = c("URL")

# remove rows that do not contain "PDF" or "pdf"
pdf_results = dplyr::filter(results, grepl("PDF|pdf",URL))
```

Download files

Using the URL list generated in the step above, this code will:

- extract filenames
- loop over URLs
- download and save files
- handle errors

```
path = "files/3Q22/"

for (f in 1:nrow(pdf_results)){

  es = paste("Downloading file #: ",f,sep="")
  print(es)

  # extract filename as the text following final "\" character in URL
  fn = strsplit(pdf_results[f,1], "/", perl=TRUE)[[1]]
  fn = fn[length(fn)]
  fp = paste(path,fn,sep="")

  tryCatch(download.file(pdf_results[f,1], fp, mode="wb"),
           error = function(e) print("Download error....skipping file."))

  Sys.sleep(5)
}
```

Extract text

Some of the files obtained via the automated query did not represent the target filetype; e.g. annual or semi-annual reports or other literature. Others represented asset allocation funds with blended benchmarks. I removed these files manually.

The next step is to extract text from the directory of PDF files. The code below:

- obtains a list of filenames from the directory
- Loops over the files and extracts elements from the PDF
- extract and combine the text elements into a single vector of words
- Check to see if a ticker symbol exists
  - If so, use this as the name of the vector.
  - If not, use XXN as the ticker (where N is the current iteration); will be manually added later (some managers do not include tickers)

- Finally, save the results as an R data object for later use.

```
# initialize results list
mat = matrix(ncol=0, nrow=0)
raw_results = list()

# get list of files within directory

path=getwd()

files = list.files(path = "files/3Q22")

# loop over files
for (f in 1:length(files)) {
  this_file = file.path("files/3Q22",files[f], fsep="/")
  print(this_file)
  file_contents = pdf_data(this_file)

  # file_contents is a nested list of X elements, where the 6th sub-element "text" is a column of words
  # loop over list and rbind these into a single column

  word_vec = data.frame(mat)
  file_words = data.frame(mat) # initialize for new file

  for (l in 1:length(file_contents)){
    word_vec = as.data.frame(file_contents[[l]][[6]])
    file_words = rbind(file_words, word_vec)
  }

  # check to see if a ticker symbol exists in this column (5 uppercase letters, ending in X);
  # if so, make it the colname
  # if not, create a temporary ticker for later manual replacement

  ticker = str_extract(file_words, "[A-Z]{4}X+")
  print(ticker)

  if (is.na(ticker)==TRUE){
    ticker = paste("XX",f,sep="")
  }

  colnames(file_words) = ticker
  raw_results = append(raw_results, file_words)
}

# Save results list as R data object
saveRDS(raw_results, "text_extract_raw_2022-11-06.Rda")
```

Text manipulation

In this step, I will convert word vectors into a single "document" (row of unigram words) for each fund. Because some fund commentaries do not provide forward outlook commentary, I will create a flag to indicate whether the document contains the word "outlook."

```
files = list.files(path = "files/3Q22")

# load data from previous step, if necessary
raw_results = readRDS("text_extract_raw_2022-11-06.Rda")

# Iterate over list and create dataframe with 3 columns: docID, Ticker and Text

documents = data.frame()

for (d in 1:length(raw_results)){

  new_row = data.frame(docID = d, Ticker = names(raw_results[d]), filename = files[d], fullText = combi
  documents = rbind(documents, new_row)

}

# Add binary column: 1 if contains "outlook", 0 if not.
documents$outlookFlag = grepl("outlook",documents$fullText, ignore.case=TRUE)

# Number of files containing outlook
sum(documents$outlookFlag)
```

```
## [1] 173
```

The word "outlook" appeared in 173 of 189 documents (92%). Next, I remove fund results that do not contain this word, and perform additional text standardization and manipulation.

```
# Remove the rows that do not contain "outlook", and remove the outlookFlag column
documents = documents[!(documents$outlookFlag==FALSE),-5]

#change case of fullText to lower
documents$fullText = tolower(documents$fullText)

# trim text from fullText prior to occurance of word "outlook"
documents$trimmed = gsub(".*outlook","",documents$fullText)

# remove fullText column to reduce dataframe size
documents = documents[,-4]


# write ticker and filename to CSV, so that I can manually add missing tickers
docTickerFN = documents[,c(1:3)]
write.csv(docTickerFN, "docTickerFN.csv")

saveRDS(documents, "documents.Rda")
```

There were 31 files with missing tickers; after manually adding them, re-import and merge with the documents dataframe.

```
# import manually added tickers
revised = read.csv("docTickerFN_revised.csv")
colnames(revised) = c("X", "docID", "newTicker", "filename")
```

```
documents = merge(documents, revised, by="filename")
documents = documents[,c(2,7,4)]
colnames(documents) = c("docID", "Ticker", "Trimmed")
```

Obtain benchmark data

Next, obtain prospectus benchmarks for each fund using the rvest package to extract values from the Ycharts website.

```
for (u in 1:nrow(documents)) {

  url = paste("https://ycharts.com/mutual_funds/M:",documents[u,2], sep="")
  page = read_html(url)
  Sys.sleep(2)

  bmraw = html_nodes(page, "body > main > div > div:nth-child(4) > div > div > div > div > div.col-md-4
                     > div.panel-content > div > div:nth-child(2) > table > tbody > tr:nth-child(2) > t

  bmtext = as.character(bmraw[[1]])
  bm = gsub( " *<.*?> *", "", bmtext)

  documents$BM[u] = bm

  print(u)
  Sys.sleep(3)
}

# manually adjust one incorrect ticker/benchmark

documents$Ticker[which(documents$Ticker == "MSIFT")] = "MUIIX"
documents$BM[which(documents$Ticker == "MUIIX")] = "ICE BofA US 3M Trsy Bill TR USD"

# replace &amp;amp; with &
documents$BM = gsub("&amp;amp;", "&", documents$BM)
```

Perform sentiment analysis

The 'analyzeSentiment' package includes several domain-optimized dictionaries for sentiment analysis. I will use all of them to generate sentiment scores and for the purpose of this project, determine which to use via qualitative spot-checks. For a more robust examination of dictionary performance, other quantitative approaches (such as correlation with benchmark-relative performance) could be used. However, this would require a larger dataset to avoid optimizing the model for the data that will later be tested. In other words, a "train/test split" approach would be required.

The development of a custom dictionary for the purpose of investment manager sentiment analysis would provide the best results, but is outside the scope of this project.

Dictionary evaluation

Dictionaries available in the 'analyzeSentiment' package include:

- GI - dictionary from the Harvard-IV dictionary as used in the General Inquirer software
- HE - dictionaryfrom Henry's Financial dictionary
- LM - dictionary from the Loughran-McDonald Financial dictionary

Please refer to project submission document for further discussion of these dictionaries.

Reference: https://cran.r-project.org/web/packages/SentimentAnalysis/SentimentAnalysis.pdf

```
dicts = c("GI", "HE", "LM")

for (d in 1:nrow(documents)){

  text = documents$Trimmed[d]

  for (z in 1:3){

    es = paste("documents$",dicts[z],"[",d,"] = analyzeSentiment(text)$Sentiment",dicts[z],sep="")
    eval(parse(text=es))

  }
  #print(d)
}

saveRDS(documents, "documents_full.Rda") # intermediate step: save results thus far
```

```
# reload documents file, if necessary
documents = readRDS("documents_full.Rda")

results = documents[,c(1,2,4,5,6,7)]
results$GI = as.numeric(results$GI)
results$HE = as.numeric(results$HE)
results$LM = as.numeric(results$LM)

# convert the raw, continuous sentiment scores into a binary (positive/negative) score.

results$GI_binary = convertToBinaryResponse(results$GI)
results$HE_binary = convertToBinaryResponse(results$HE)
results$LM_binary = convertToBinaryResponse(results$LM)

# examine the balance of positive and negative sentiment obtained via each method

print(paste("GI positive: ", round((sum(results$GI_binary == "positive")/nrow(results))*100,0),"%",sep=
```

```
## [1] "GI positive: 100%"
```

```
print(paste("HE positive: ", round((sum(results$HE_binary == "positive")/nrow(results))*100,0),"%",sep=
```

```
## [1] "HE positive: 74%"
```

```
print(paste("LM positive: ", round((sum(results$LM_binary == "positive")/nrow(results))*100,0),"%",sep=
```

```
## [1] "LM positive: 24%"
```

From the results above, we see that:

- The GI dictionary produced 100% positive sentiment, which does not seem useful (particularly given the highly uncertain nature of the current economic and market environment).
- The HE dictionary produces sentiment scores skewed toward the positive. While this may seem improbable given the current economic environment, Portfolio Managers at asset management firms are often an optimistic bunch - particularly within their own investment category.
- The LM dictionary proved skewed toward the negative to a similar degree.

After some spot-checking of a sample of higher and lower scores generated by the HE and LM dictionaries (a sample of which will be included in the final report), I chose the HE dictionary. I will examine both the binary and continuous results from this dictionary.

```
# trim results to remove extraneous columns. Keep HE score and original HE binary scores.

results = results[,c(1:3,5,8)]

# remove funds with missing benchmark
results = results[!grepl("Bloomberg Short-term Gov/Corp TR USD", results$BM),]

# remove funds with sentiment scores of 0 (these represent errors introduced by my simplistic approach
results = results[!results$HE==0,]
```

Collect fund and benchmark performance data

The next step was to find data for either the benchmark index used by each fund, or a passively managed ETF that tracks the indices used by the mutual funds included in the analysis. There were 37 unique indices. I performed this step manually, using a free trial subscription to the YCharts service to obtain access to some of the less-common indices. Even so, this was a manual process because the free trial did not provide access to the YCharts API or Excel plug-in.

My approach was two-fold:

- Use the Yahoo! Finance API and 'yahoofinancer' R library to obtain start and end date prices (10/1/2022 through 11/23/2022) for the mutual funds and ETFs used as index proxies
- Manually obtain historical price data for the remaining indices where no ETF was available using the Ycharts Web interface

```
# import manually-created list of ETFs/Funds available as index proxies
proxies = read.csv("etfs.csv")

# combine into a single vector of tickers

proxy_tickers = proxies$Fund
result_tickers = results$Ticker
all_tickers = as.data.frame(append(result_tickers, proxy_tickers))
colnames(all_tickers) = c("Ticker")

# add placeholder column for holding period returns
all_tickers$HPR = NA
```

```
# call Yahoo! Finance API to obtain beginning and end prices, calculate holding period return, and add

for (t in 1:nrow(all_tickers)){

  # set ticker value
  es = paste(all_tickers[t,1]," = Ticker$new(\"",all_tickers[t,1],"\")",sep="")
  eval(parse(text=es))

  # call yahoo API to get data table
  es = paste("y_data = ",all_tickers[t,1],"$get_history(start=\"2022-10-1\", end=\"2022-11-23\", interva
  eval(parse(text=es))

  # get beginning and ending adjusted price (to account for income distributions), and calculate hpr
  begin = y_data[1,7]
  end = y_data[nrow(y_data),7]
  hpr = (end-begin)/begin

  # add hpr to all_tickers dataframe
  all_tickers$HPR[t] = hpr
  print(t)
}

# save to CSV so I can add the manual entries
write.csv(all_tickers, "all_tickers.csv")
```

After adding holding period return values for the indices for which I could not locate a proxy ETF, I re-imported the files with tickers/returns and benchmarks/returns, merged values into the results table, and calculated benchmark-relative excess returns for each fund.

NOTE: this analysis does not reflect the lowest-cost available share class of funds, resulting in an inconsistent performance impact of fees. This represents another area of improvement of this analysis.

```
# reimport tickers and HPRs, and Benchmarks and HPRs (including those added manually)
fundHPRs = read.csv("fundHPRs.csv")
BMHPRs = read.csv("BMHPRs.csv")


# add column for fund returns, merge values, and rename column
results2 = merge(results, fundHPRs, by="Ticker", all.x = TRUE)
names(results2)[names(results2) == 'HPR'] = 'fundHPR'

# add column for benchmark returns, merge values, and rename column
results3 = merge(results2, BMHPRs, by="BM", all.x=TRUE)
names(results3)[names(results3) == 'HPR'] = 'BMHPR'

# calculate and store benchmark-relative excess return
results3$ExcessReturn = results3$fundHPR-results3$BMHPR

# calculate and store up-market capture
results3$upCapture = results3$fundHPR/results3$BMHPR

# reorder columns
results3 = results3[,c(3,2,1,4:9)]
```

```
# add binary column for true results
results3$Truth = ifelse(results3$BMHPR >= 0,"positive","negative")

# save intermediate results
saveRDS(results3, "results3.Rda")
```

Analysis of model performance

The first observation is that for the limited time period examined, all of the benchmarks represented show a positive holding period return. This is unusual, but consistent with market conditions this year where asset classes that generally show differing return patterns have become synchronized. However, this complicates my analysis somewhat because it is difficult to assess whether a manager is directionally "correct" in their sentiment in terms of positive and negative returns.

The use of benchmark-relative excess returns (Return of fund - Return of benchmark) is also problematic because the wide variation of typical magnitude of returns across asset classes makes them incomparable. I will therefore use up-market capture to provide better cross-asset comparability: Return of fund / Return of Benchmark. This ratio describes the percentage of benchmark returns that the fund was able to capture. A negative up-market capture ratio indicates that the fund generated a negative return, while the benchmark was positive.

Binary assessment: Confusion Matrix

The first test is to see if the fund managers perform better than a "coin flip" at predicting the future investment environment.

```
# read results
results3 = readRDS("results3.Rda")

# create predicted and truth vectors for confusion matrix
confMtx = results3[,c(5,10)]
confMtx = apply_labels(confMtx, HE_binary="Predicted Outcome", Truth="True Outcome")

pandoc.table(cross_cases(confMtx, HE_binary, Truth))
```

```
##
## ----------------------------------------------------------
##            row_labels             True Outcome|positive
## ------------------------------- ------------------------
##    Predicted Outcome|negative              44
##
##    Predicted Outcome|positive              121
##
##  Predicted Outcome|#Total cases            165
## ----------------------------------------------------------
```

As shown above, all of the actual benchmark returns were positive, while 121 of the predicted sentiment scores were positive and 44 were negative. This results in:

- Sensitivity (Recall): 73% - measures the model's ability to detect true positives

- Specificity: N/A - measures the model's ability to avoid false negatives; however in this case, no true negatives exist

An analysis over longer time periods, representing a range of different market environments, would be more illustrative and is a clear area for further refinement. The challenge over longer time frames will be the collection of source documents (quarterly commentary PDF files), likely requiring the use of other approaches than the Google query technique.

Degree of outperformance of "correct" managers using binary sentiment

The next test will examine whether the 121 managers whose sentiment was "correct" outperformed the 44 that were incorrect, using a simple mean of Up Market Capture of the two cohorts.

```
# create flag if fund manager's sentiment is correct
results3$Correct = ifelse(results3$HE_binary=="positive",1,0)

cor_mean = results3 %>% filter(Correct == 1) %>% summarise(Correct = mean(upCapture))
incor_mean = results3 %>% filter(Correct != 1) %>% summarise(Incorrect = mean(upCapture))
print(paste("Upside Capture of correct managers: ",round(cor_mean*100,0),"%",sep=""))
```

```
## [1] "Upside Capture of correct managers: 83%"
```

```
print(paste("Upside Capture of incorrect managers: ",round(incor_mean*100,0),"%",sep=""))
```

```
## [1] "Upside Capture of incorrect managers: 66%"
```

As shown above, the managers who correctly predicted the future environment (via binary, positive/negative sentiment) demonstrated a higher up-market capture than those who were incorrect. However additional analysis would be necessary to determine whether this difference is statistically significant.

Regression: Degree of outperformance using continous sentiment scores

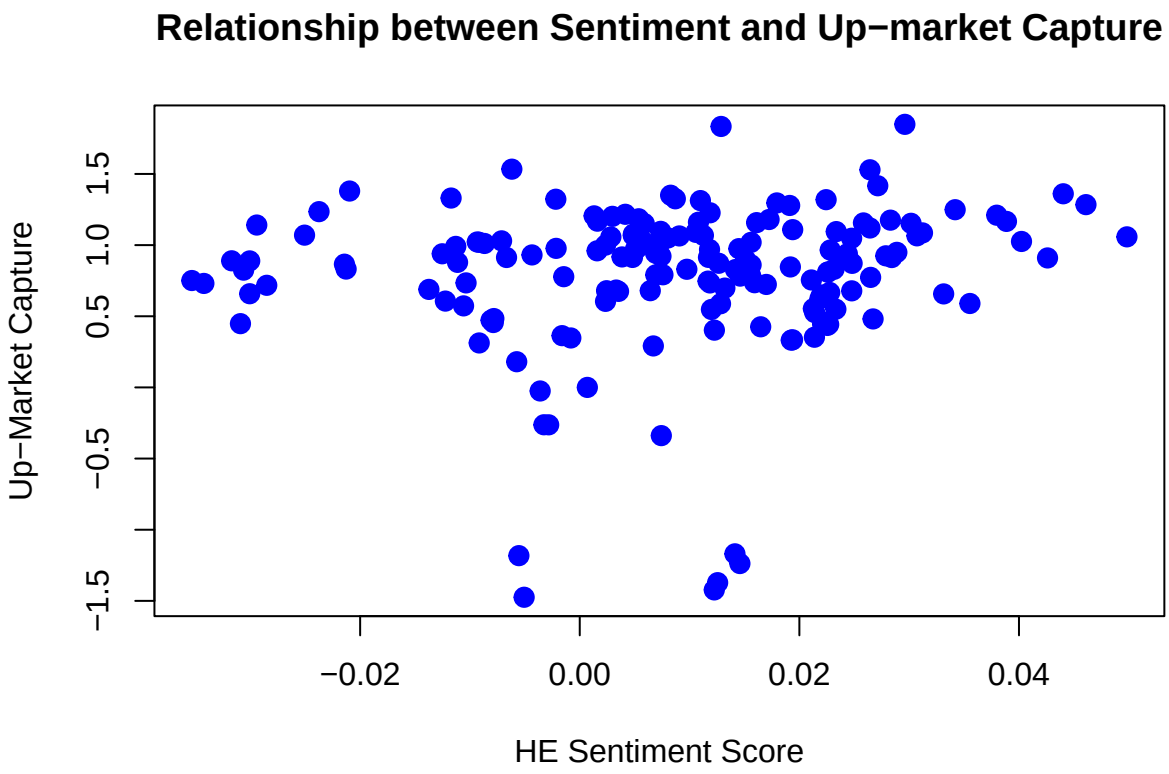
Finally, I will evaluate the continuous sentiment score against the degree of out(under)performance using linear regression to examine the explanatory power of sentiment relative to the degree of up-market capture.

```
# Linear regression analysis

# setup response (y) and predictor (x) variables
y = results3$upCapture
x = results3$HE #continuous sentiment scores

reg_model = data.frame(x,y)

plot(reg_model, col='blue', pch=20, cex=2, main="Relationship between Sentiment and Up-market Capture",
    xlab="HE Sentiment Score", ylab="Up-Market Capture")
```

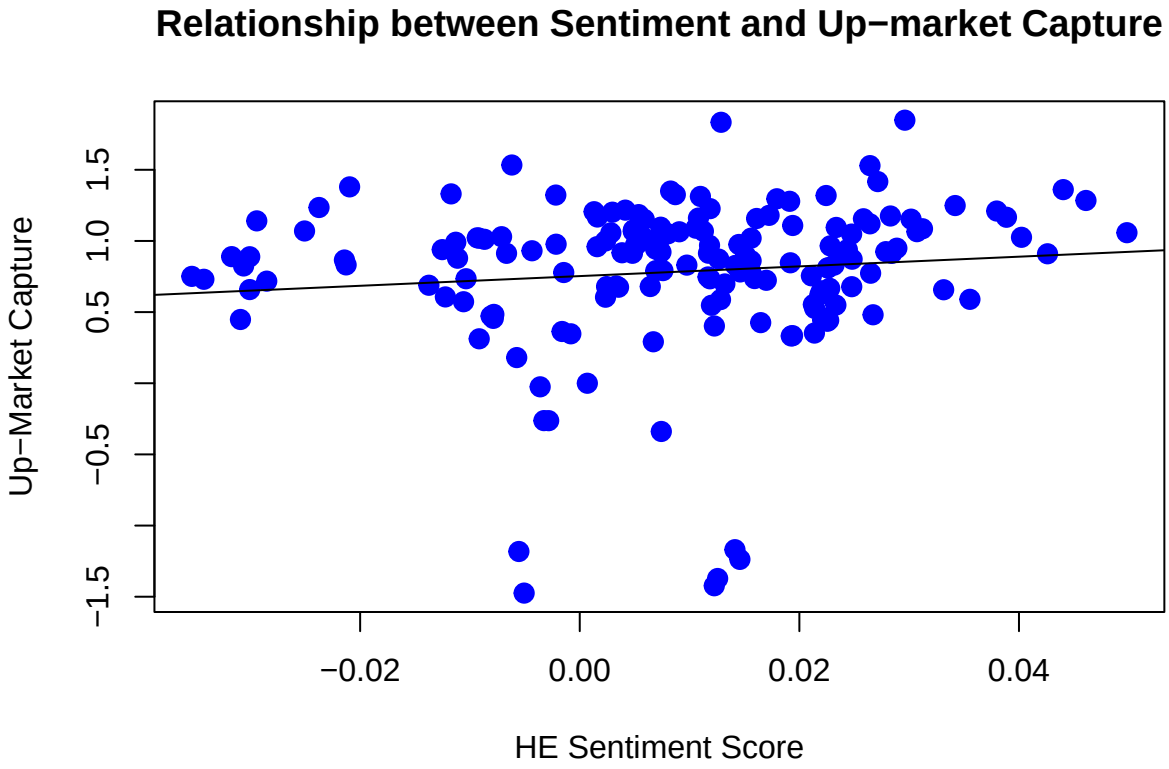


A linear relationship is not evident from the scatterplot above; however I will perform regression to confirm this quantitatively.

```
relation = lm(y~x)
print(summary(relation))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2179 -0.1634  0.1029  0.2944  1.0368
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.75308    0.04683  16.081   <2e-16 ***
## x            3.40959    2.39715   1.422    0.157
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5347 on 163 degrees of freedom
## Multiple R-squared:  0.01226,    Adjusted R-squared:  0.0062
## F-statistic: 2.023 on 1 and 163 DF,  p-value: 0.1568
```

```
plot(reg_model, col='blue', pch=20, cex=2, main="Relationship between Sentiment and Up-market Capture",
    xlab="HE Sentiment Score", ylab="Up-Market Capture")
abline(relation)
```



For this regression analysis, the null hypothesis is that continuous sentiment scores do not have a relationship with degree of up-market capture. With a t-value of 1.422 and a p-value of 15.7%, I fail to reject this null hypothesis.

Given the limited number of funds examined and the short performance window for benchmark-relative valuation, I cannot conclude that there is a relationship between continuous sentiment scores and degree of up-market capture.