Economic Events

How BioTech stock abnormal returns response to market events and news sentiment

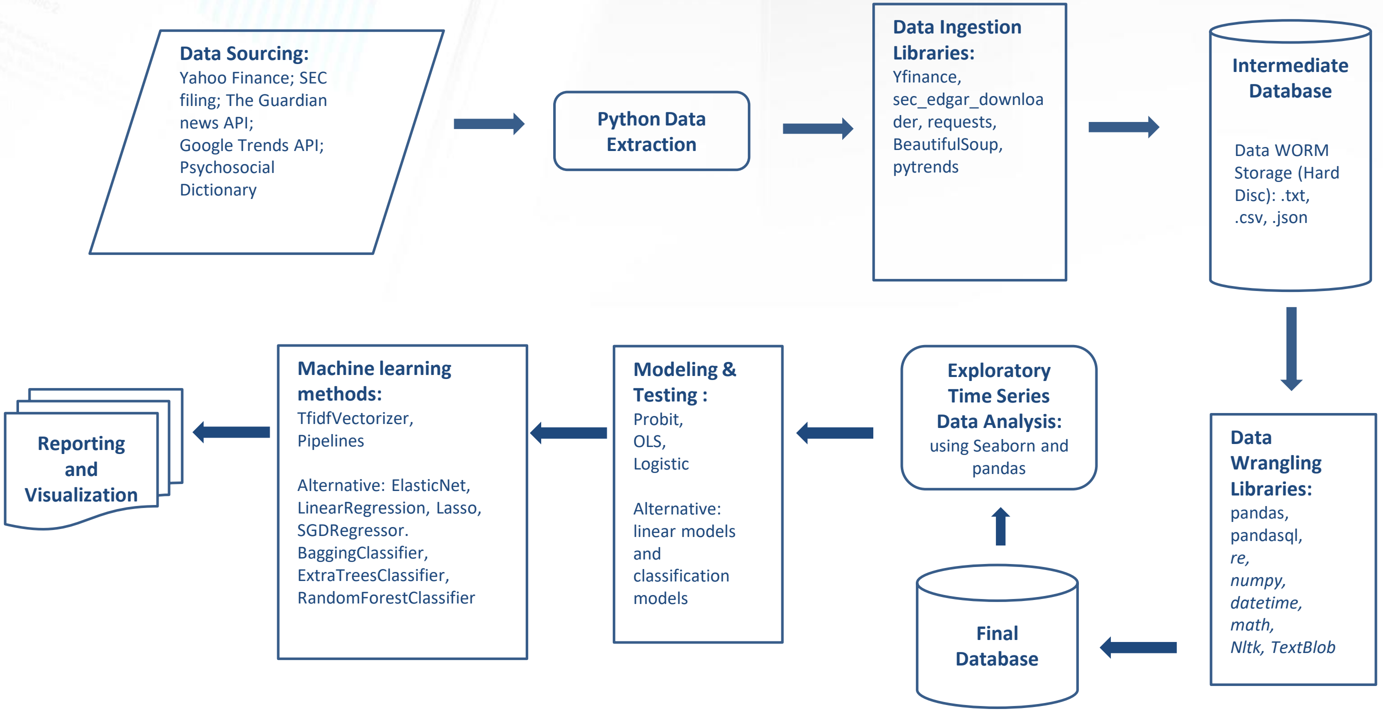
Jorge de Leon Miranda, Stephen Sigrist, Richard Zhai

Introduction

- Studies [1] have shown stock returns are affected by economic events and company specific news beyond the technical analysis which is based on market data of price and volume
- The project seeks to predict BioTech sector stock residual returns by words from economic events and company sentiment news using data science technology
- Residual Returns are defined as Actual Return minus Predicted Normal Return where predicted normal return for individual stock is modeled against broad market stock benchmark [2] return
- Seven BioTech companies are covered in this project: Alexion, Amgen, Biogen, Gilead, Incyte, Regeneron and Vertex

1. *See Research Reference in Appendix*
2. *S&P 500 Total Return Index is selected as market benchmark in the project*

Project Infrastructure



Data Sourcing: Yahoo Finance; SEC filing; The Guardian news API; Google Trends API; Psychosocial Dictionary
Python Data Extraction
Data Ingestion Libraries: Yfinance, sec_edgar_downloader, requests, BeautifulSoup, pytrends
Intermediate Database
Data WORM Storage (Hard Disc): .txt, .csv, .json
Data Wrangling Libraries: pandas, pandasql, *re, numpy, datetime, math, Nltk, TextBlob*
Final Database
Exploratory Time Series Data Analysis: using Seaborn and pandas
Modeling & Testing : Probit, OLS, Logistic
Alternative: linear models and classification models
Machine learning methods: TfidfVectorizer, Pipelines
Alternative: ElasticNet, LinearRegression, Lasso, SGDRegressor. BaggingClassifier, ExtraTreesClassifier, RandomForestClassifier
Reporting and Visualization

Data Ingestion

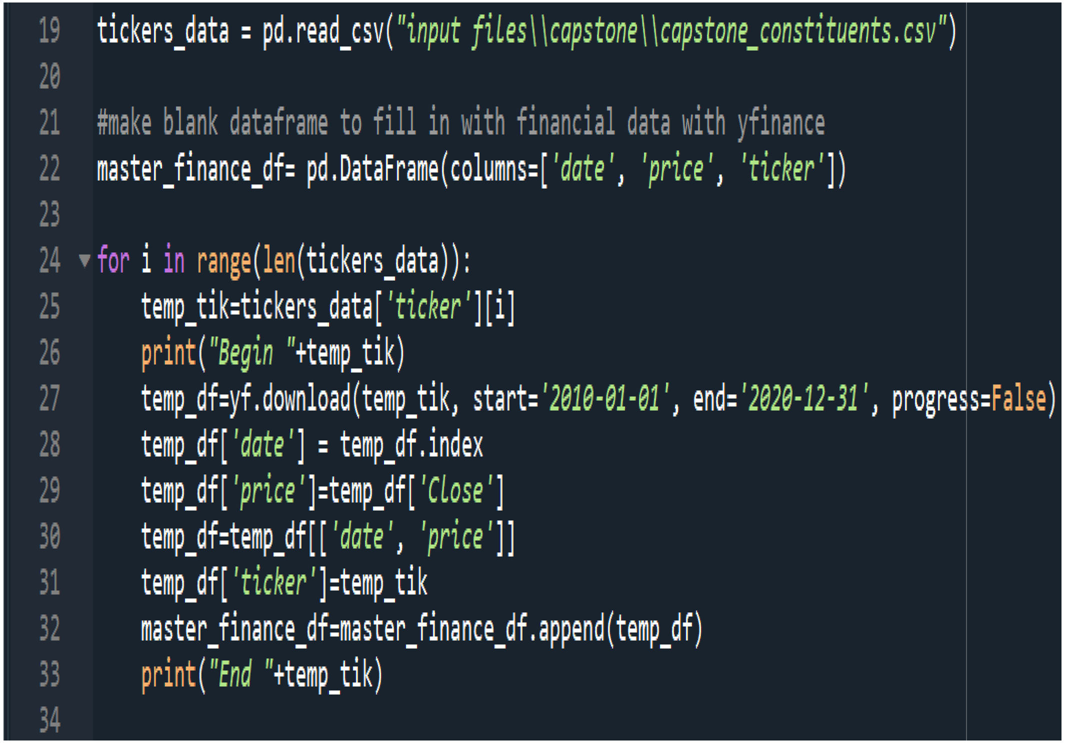
- Data Sources
- Ingestion Process
- Ingestion Results

Google

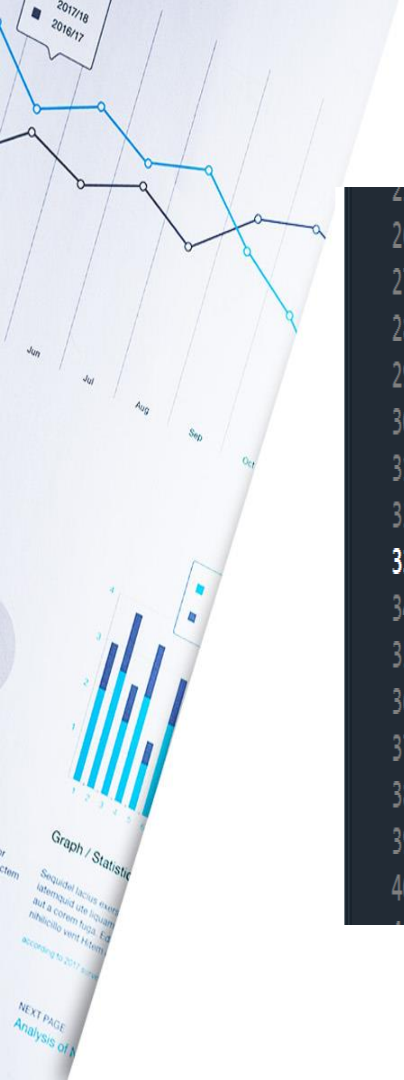
YAHOO! FINANCE

theguardian

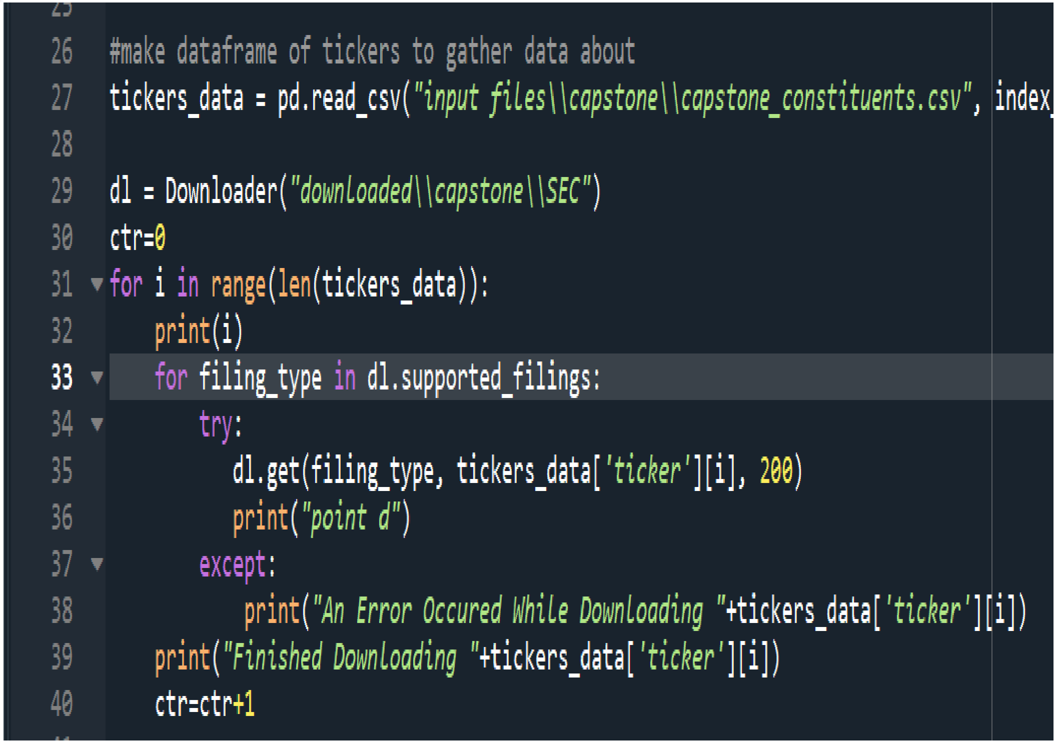
Data Ingestion: Stocks



```
tickers_data = pd.read_csv("input files\\capstone\\capstone_constituents.csv")

#make blank dataframe to fill in with financial data with yfinance
master_finance_df= pd.DataFrame(columns=['date', 'price', 'ticker'])

for i in range(len(tickers_data)):
    temp_tik=tickers_data['ticker'][i]
    print("Begin "+temp_tik)
    temp_df=yf.download(temp_tik, start='2010-01-01', end='2020-12-31', progress=False)
    temp_df['date'] = temp_df.index
    temp_df['price']=temp_df['Close']
    temp_df=temp_df[['date', 'price']]
    temp_df['ticker']=temp_tik
    master_finance_df=master_finance_df.append(temp_df)
    print("End "+temp_tik)

```

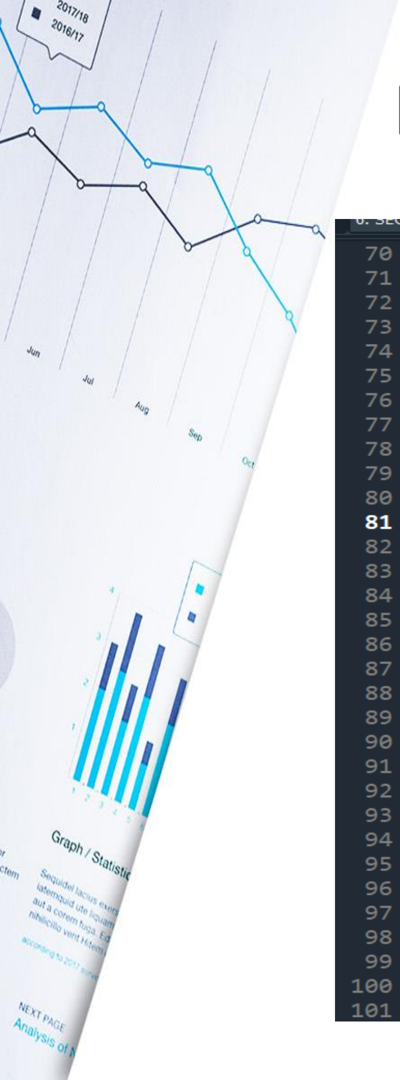


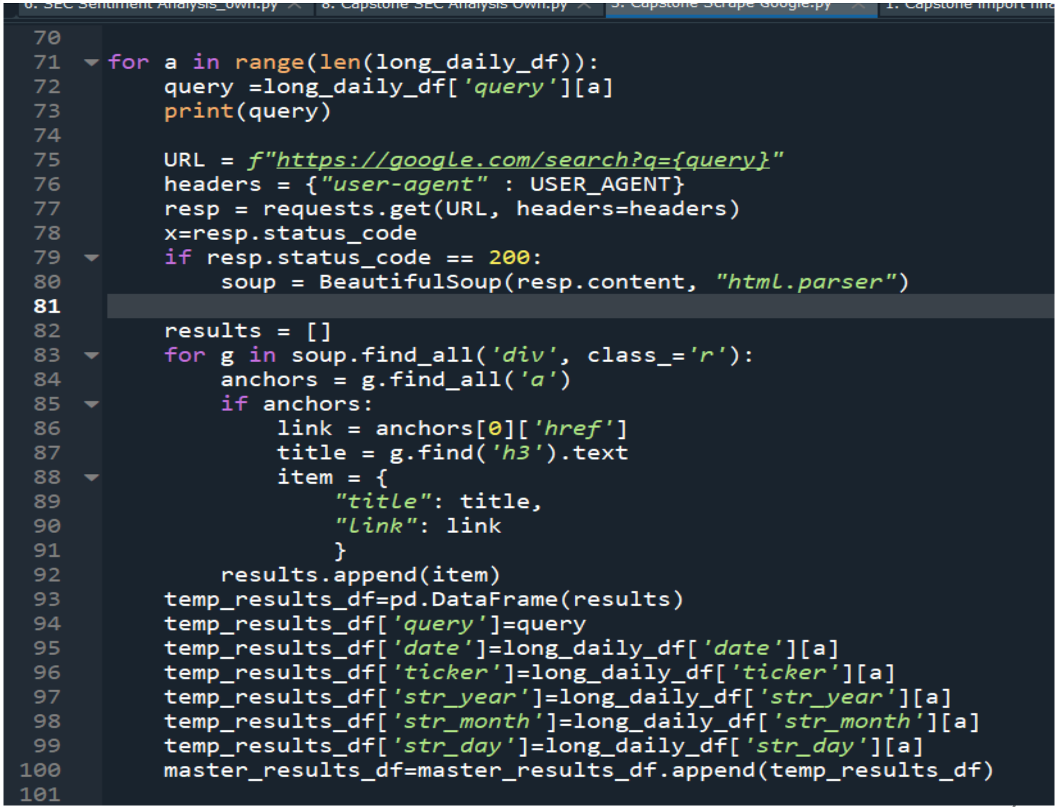
Data Ingestion: SEC Filings



```
#make dataframe of tickers to gather data about
tickers_data = pd.read_csv("input files\\capstone\\capstone_constituents.csv", index

dl = Downloader("downloaded\\capstone\\SEC")
ctr=0
for i in range(len(tickers_data)):
    print(i)
    for filing_type in dl.supported_filings:
        try:
            dl.get(filing_type, tickers_data['ticker'][i], 200)
            print("point d")
        except:
             print("An Error Occured While Downloading "+tickers_data['ticker'][i])
    print("Finished Downloading "+tickers_data['ticker'][i])
    ctr=ctr+1
```

Data Ingestion: Google Scrapes




6. SEC Sentiment Analysis_own.py
8. Capstone SEC Analysis Own.py
5. Capstone Scrape Google.py
1. Capstone import fina

```python
for a in range(len(long_daily_df)):
    query =long_daily_df['query'][a]
    print(query)

    URL = f"https://google.com/search?q={query}"
    headers = {"user-agent" : USER_AGENT}
    resp = requests.get(URL, headers=headers)
    x=resp.status_code
    if resp.status_code == 200:
        soup = BeautifulSoup(resp.content, "html.parser")

    results = []
    for g in soup.find_all('div', class_='r'):
        anchors = g.find_all('a')
        if anchors:
            link = anchors[0]['href']
            title = g.find('h3').text
            item = {
                "title": title,
                "link": link
                }
        results.append(item)
    temp_results_df=pd.DataFrame(results)
    temp_results_df['query']=query
    temp_results_df['date']=long_daily_df['date'][a]
    temp_results_df['ticker']=long_daily_df['ticker'][a]
    temp_results_df['str_year']=long_daily_df['str_year'][a]
    temp_results_df['str_month']=long_daily_df['str_month'][a]
    temp_results_df['str_day']=long_daily_df['str_day'][a]
    master_results_df=master_results_df.append(temp_results_df)
```

Google Scrapes

- The web was queried for each stock on each trading day during the past 10 years

- Results matching the following were stored:
  - Articles selected from known high-impact disseminators: New York Times, Wall Street Journal, Bloomberg, Financial Times, Seeking Alpha

  - Matching month and year in the web address

Data Ingestion: Sentiment Words

- http://www.wjh.harvard.edu/~inquirer/homecat.htm
- Roughly 1,600 and 2,300 positive and negative sentiment words were selected and stored in a csv

Data Ingestion: Guardian Articles

- The Guardian free newspaper was used for this project using the Guardian API and the requests library

- Issue: limits on the number of articles that can be downloaded per request, and limitation on the number of requests per day

- Solution: A code was created to search and request all articles in the Guardian that contained the name of the company and industry relevant words for each day since January 1, 2000

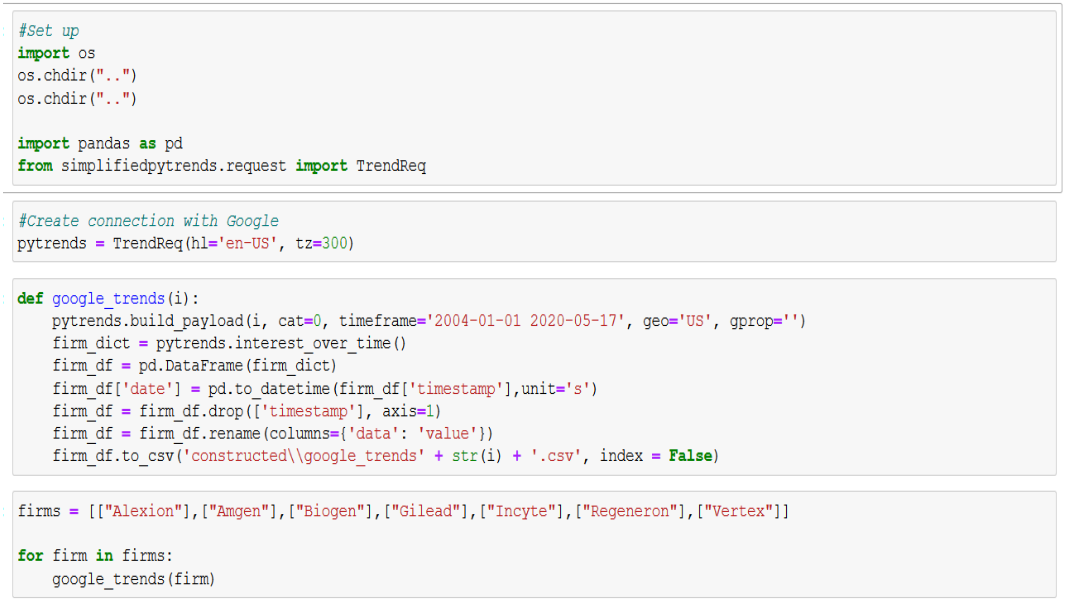
- The following terms were searched for:
  - “Alexion”, “Amgen”, “Gilead”, “Incyte”, Regeneron”, “Vertex”, “Biotechnology” and “Pharmaceutical”

- All articles that matched the search query for each day for each term were stored as a json file and later converted to time series data frames

- Other news data sources were explored: Factiva, New York Times, Wall Street Journal, Financial Times and News API – not free or not suitable for this project.

Data Ingestion: Google Trends Data

- ‘PyTrends’ was used to request data from the Google Trends API
- Names for the companies are searched as Term Frequency
- Data available since 2004 on a monthly basis



```
#Set up
import os
os.chdir("..")
os.chdir("..")

import pandas as pd
from simplifiedpytrends.request import TrendReq
```

```
#Create connection with Google
pytrends = TrendReq(hl='en-US', tz=300)
```

```
def google_trends(i):
    pytrends.build_payload(i, cat=0, timeframe='2004-01-01 2020-05-17', geo='US', gprop='')
    firm_dict = pytrends.interest_over_time()
    firm_df = pd.DataFrame(firm_dict)
    firm_df['date'] = pd.to_datetime(firm_df['timestamp'],unit='s')
    firm_df = firm_df.drop(['timestamp'], axis=1)
    firm_df = firm_df.rename(columns={'data': 'value'})
    firm_df.to_csv('constructed\\google_trends' + str(i) + '.csv', index = False)
```

```
firms = [["Alexion"],["Amgen"],["Biogen"],["Gilead"],["Incyte"],["Regeneron"],["Vertex"]]

for firm in firms:
    google_trends(firm)
```

Data Wrangling

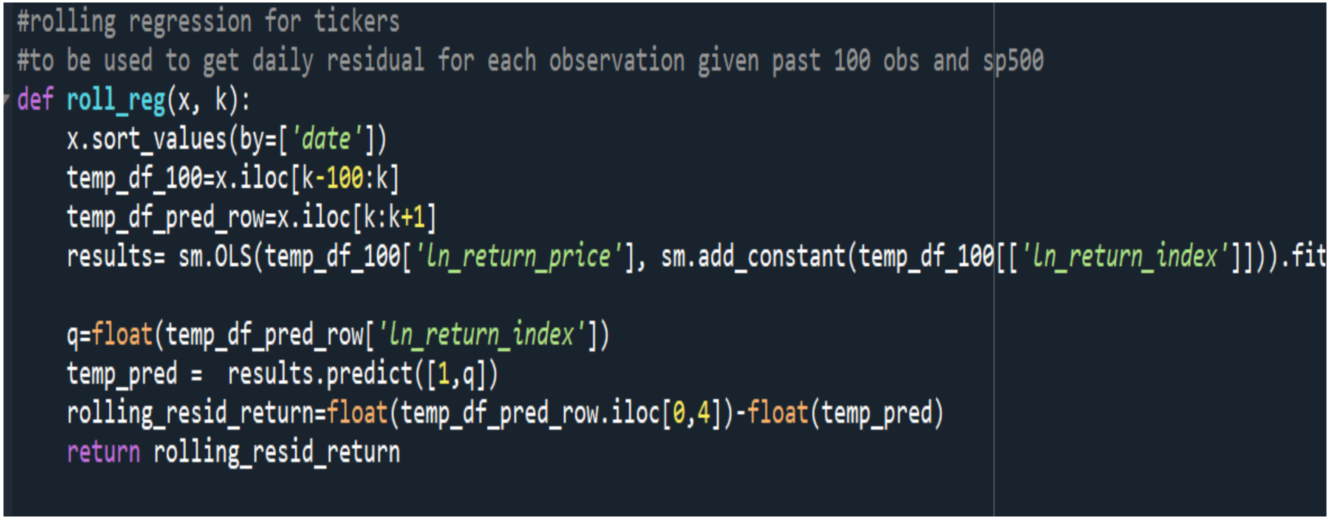
- Date and time were extracted from SEC Documents with regular expressions

- The daily stock log return and S&P 500 Index log return were calculated

- Google web results and SEC Documents were scanned for sentiment words per the psychosocial dictionary:
  - words_df= pd.read_csv("input files\\capstone\\capstone_sentiment.csv") for k in range(len(words_df)):
  - stemp_df[f"w_{k}"]=temp_text.upper().count(words_df['word'][k])

Data Wrangling

- Abnormal/Residual return was calculated on a daily basis. A rolling regression estimated a market model for each day based on the prior 100 days observations, using the estimated equation to calculated a predicted return.
  - Log Abnormal Return=Log Actual Return –Log Predicted
  - Arithmetic Residual Return=EXP(Log Abnormal Return)-1



```
#rolling regression for tickers
#to be used to get daily residual for each observation given past 100 obs and sp500
def roll_reg(x, k):
    x.sort_values(by=['date'])
    temp_df_100=x.iloc[k-100:k]
    temp_df_pred_row=x.iloc[k:k+1]
    results= sm.OLS(temp_df_100['ln_return_price'], sm.add_constant(temp_df_100[['ln_return_index']])).fit

    q=float(temp_df_pred_row['ln_return_index'])
    temp_pred =  results.predict([1,q])
    rolling_resid_return=float(temp_df_pred_row.iloc[0,4])-float(temp_pred)
    return rolling_resid_return
```

Data Wrangling Issues

- Issues:
  - Limited data was picked up by the google scrape; For the 7 biotech stocks, monthly observations are fewer than 500 after inner join
  - SEC documents are large text files, taking a lot of space on computer:
    - Solution: Documents for each stock were downloaded 1 at a time, then analyzed and deleted algorithmically:

```
shutil.rmtree("downloaded\\big capstone\\SEC",
ignore_errors=True)
os.mkdir("downloaded\\big      capstone\\SEC")
```

  - Text data downloaded from the internet and SEC documents often has html formatting code that complicates bag-of-words feature extraction.
    - Solution: Clean documents of punctuation and keep only English words:

```
import nltk
words = set(nltk.corpus.words.words())
 temp_text_c2=re.sub('[^abcdefghijklmnopqrstuvwxyz\s]', '', temp_text_c1)
temp_text_c3=" ".join(w for w in nltk.wordpunct_tokenize(temp_text_c2) \
if w.lower() in words or not w.isalpha())
```

Data Wrangling

- Monthly average sentiment polarity and subjectivity indicators were calculated for all Guardian News for a given month; Using NLTK to clean text data and TextBlob to generate these indicators
- Next, the monthly average indicators were organized in a time series since 2000-01-01.
- The process above was repeated for each term and articles containing words of “biotechnology” and “pharmaceutical”. Other indicators created the number of articles on a given month, and the average number of words in the article.
- The Google Trends data was cleaned to produce a monthly indicator since 2004-01-01 for the relative frequency of the number of searches of the name of the companies in Google.

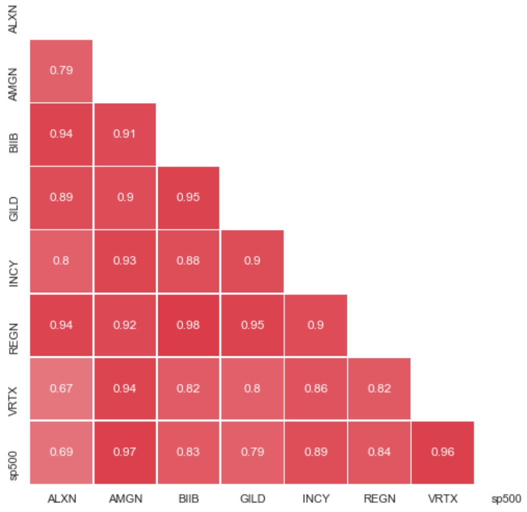
Data Wrangling Issues

- Issues:
  - All Guardian News that match the search query for each term were downloaded on a different json file for each day. An alternative to analyze text data is to generate indicators such as sentiment polarity and subjectivity indicators, number of words per article, number of articles per day, frequency of specific words. The issue is that one needs these data to analyze a time series 'target'

    - Solution: Clean data and generate time series information from text data:
      - Convert json files to dataframes
      - Eliminate non-article data such as blogs, video scripts, etc.
      - Clean text data using nltk (eliminate urls, lower case, punctuation, remove stop words)
      - Generate sentiment polarity and subjectivity for each article using TextBlob
      - Use pandas to generate daily and monthly average and sums for each indicator.

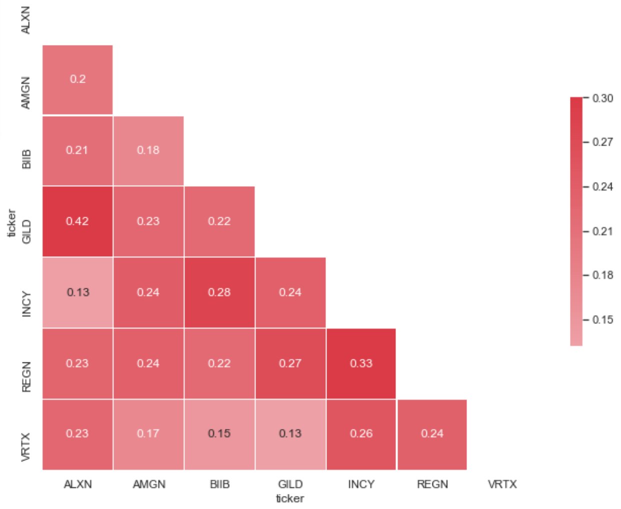
Exploratory Data Analysis

Correlations between stock prices



	ALXN	AMGN	BIIB	GILD	INCY	REGN	VRTX	sp500
ALXN								
AMGN	0.79							
BIIB	0.94	0.91						
GILD	0.89	0.9	0.95					
INCY	0.8	0.93	0.88	0.9				
REGN	0.94	0.92	0.98	0.95	0.9			
VRTX	0.67	0.94	0.82	0.8	0.86	0.82		
sp500	0.69	0.97	0.83	0.79	0.89	0.84	0.96	

Correlations between residual returns

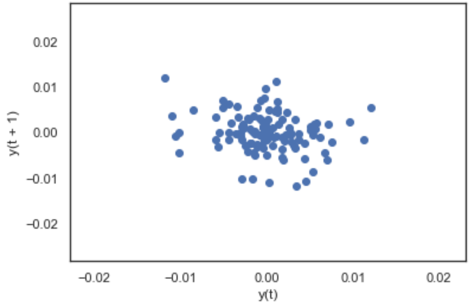


ticker	ALXN	AMGN	BIIB	GILD	INCY	REGN	VRTX
ALXN							
AMGN	0.2						
BIIB	0.21	0.18					
GILD	0.42	0.23	0.22				
INCY	0.13	0.24	0.28	0.24			
REGN	0.23	0.24	0.22	0.27	0.33		
VRTX	0.23	0.17	0.15	0.13	0.26	0.24	

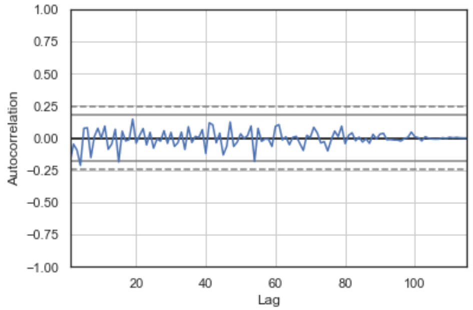
Exploratory Data Analysis

ALXN - Autocorrelation

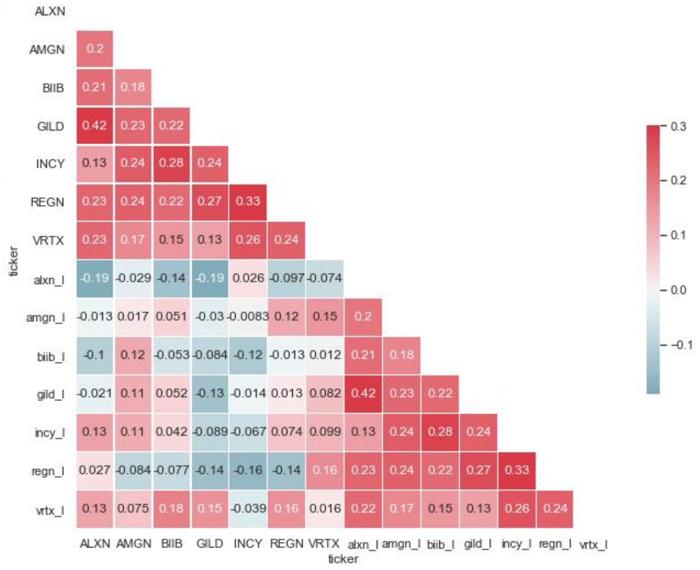
Lag plot



Autocorrelation plot



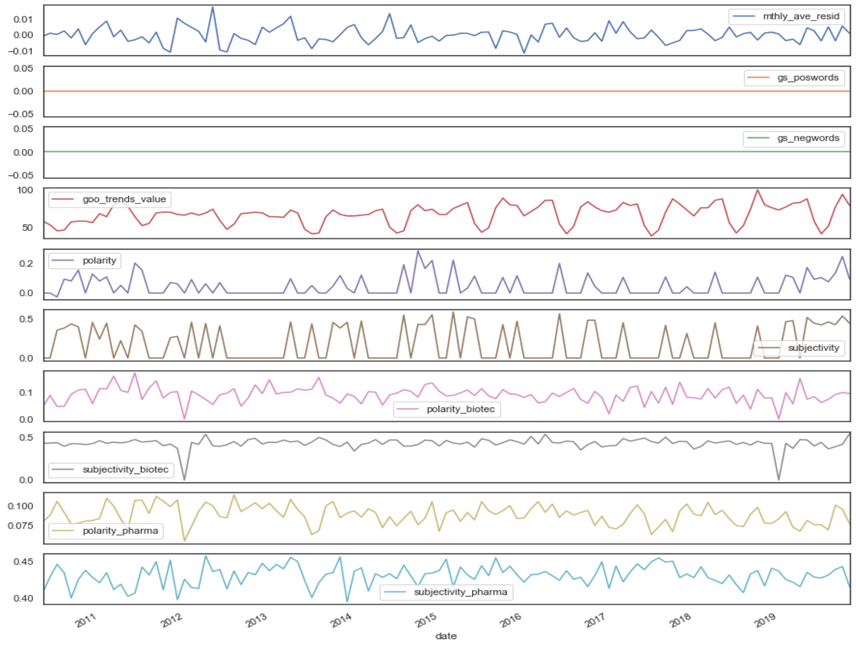
Correlation with one-month lag residual returns of the other companies



ticker	ALXN	AMGN	BIIB	GILD	INCY	REGN	VRTX	alxn_l	amgn_l	biib_l	gild_l	incy_l	regn_l	vrtx_l
ALXN														
AMGN	0.2													
BIIB	0.21	0.18												
GILD	0.42	0.23	0.22											
INCY	0.13	0.24	0.28	0.24										
REGN	0.23	0.24	0.22	0.27	0.33									
VRTX	0.23	0.17	0.15	0.13	0.26	0.24								
alxn_l	-0.19	-0.029	-0.14	-0.19	0.026	-0.097	-0.074							
amgn_l	-0.013	0.017	0.051	-0.03	-0.0083	0.12	0.15	0.2						
biib_l	-0.1	0.12	-0.053	-0.084	-0.12	-0.013	0.012	0.21	0.18					
gild_l	-0.021	0.11	0.052	-0.13	-0.014	0.013	0.082	0.42	0.23	0.22				
incy_l	0.13	0.11	0.042	-0.089	-0.067	0.074	0.099	0.13	0.24	0.28	0.24			
regn_l	0.027	-0.084	-0.077	-0.14	-0.16	-0.14	0.16	0.23	0.24	0.22	0.27	0.33		
vrtx_l	0.13	0.075	0.18	0.15	-0.039	0.16	0.016	0.22	0.17	0.15	0.13	0.26	0.24	

Exploratory Data Analysis

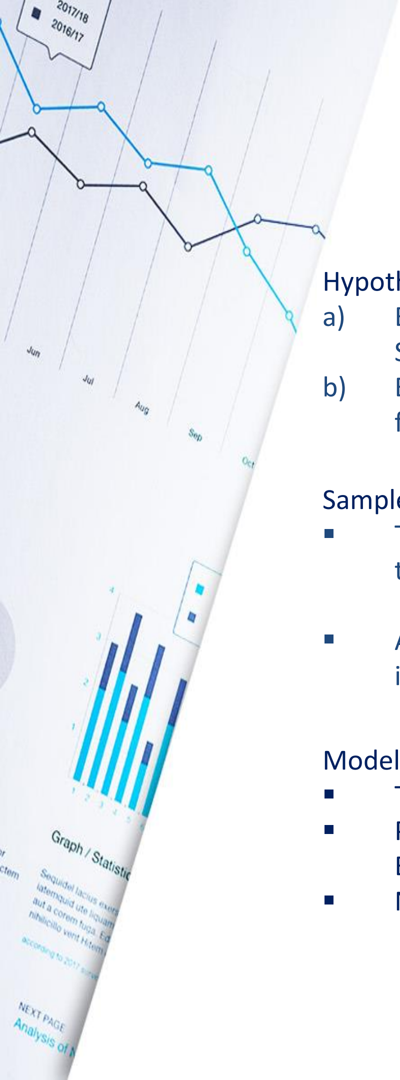
Vertex – Feature plot



Machine Learning

- Many documents and articles don't have words corresponding to Psychosocial dictionary words:
  - Solution: Create features with TfidVectorizer, unsupervised learning.

- <u>Alternative:</u> modeling analysis uses time series data derived from text data



Regression Model Methodology

Hypothesis

a) BioTech Stock residual returns are predictable by n-gram features that come from SEC filings

b) BioTech Stock residual returns are predictable by company news sentiments and frequency of searches

Samples

- The main sample is the merged daily stock observations of the 7 biotech stocks and the dates on which SEC documents were released; Merged days are kept

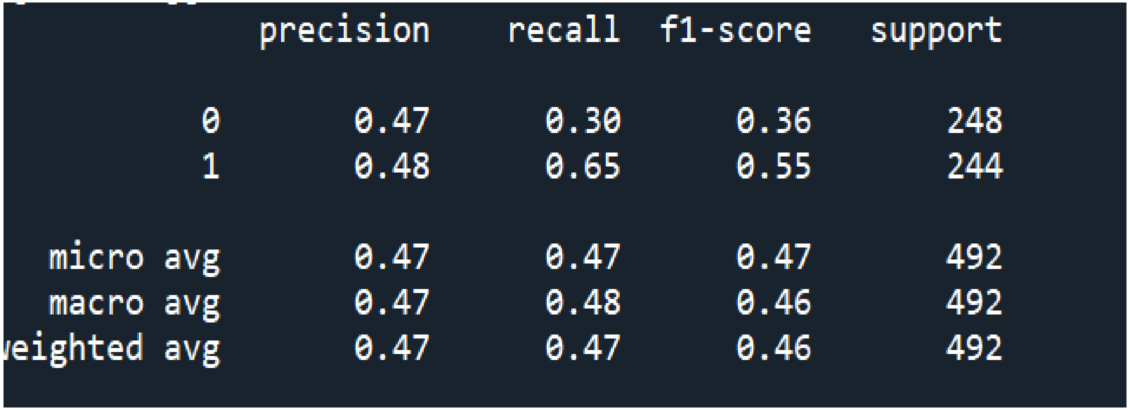
- A larger sample of random stocks and merged SEC filings was also evaluated. Task is binary classification: 1 is arithmetic residual increased; 0 otherwise.

Model Selection

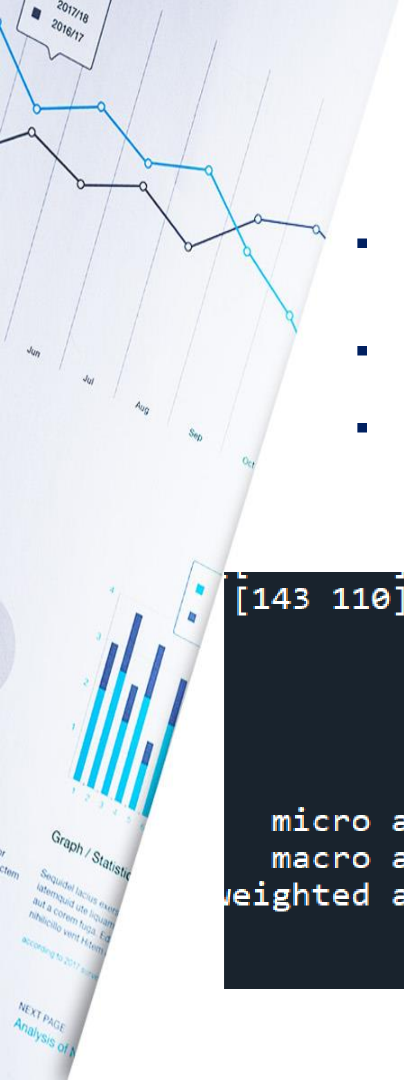
- TfidVEctorizerCV
- PipeLines: SVC, SGDClassifier, KNeighborsClassifier, LogisticRegressionCV, BaggingClassifier()
- Nothing performed better than LogisticRegressionCV

Model Performance Analysis

- 7 Biotech Stock Stock Sample: How do features extracted from SEC documents with TfidfVectorizer Predict Daily Abnormal Stock Returns:
- Total Observations: 984, 50% TTS split:
- Using LogisticRegression():
  Test Score: 48%

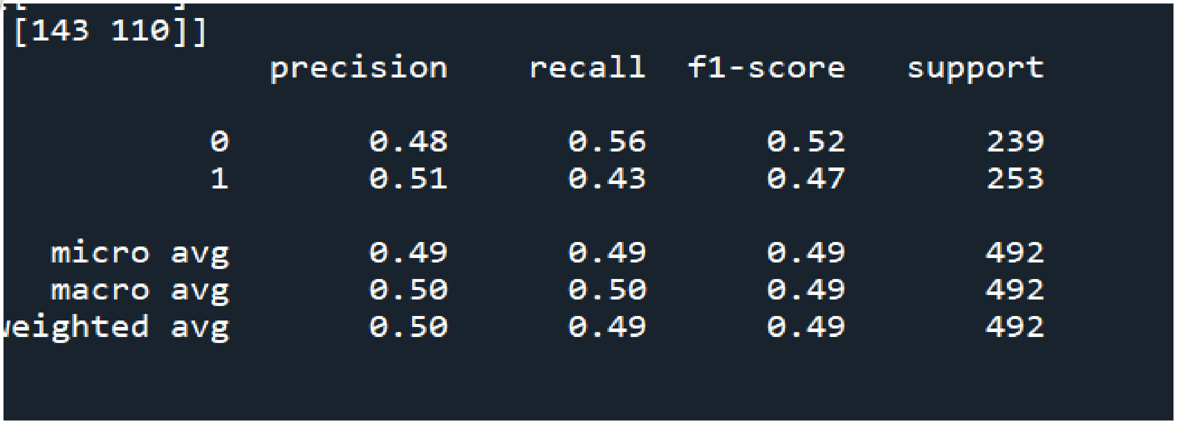


```
              precision    recall  f1-score   support

           0       0.47      0.30      0.36       248
           1       0.48      0.65      0.55       244

   micro avg       0.47      0.47      0.47       492
   macro avg       0.47      0.48      0.46       492
eighted avg       0.47      0.47      0.46       492
```



Model Performance Analysis

- 7 Biotech Stock Stock Sample: How do features extracted from SEC documents with TfidfVectorizer Predict Daily Abnormal Stock Returns:
- Total Observations: 984, 50% TTS split:
- Using LogisticRegressionCV() (With Cross Validation):
  Test Score: 50%



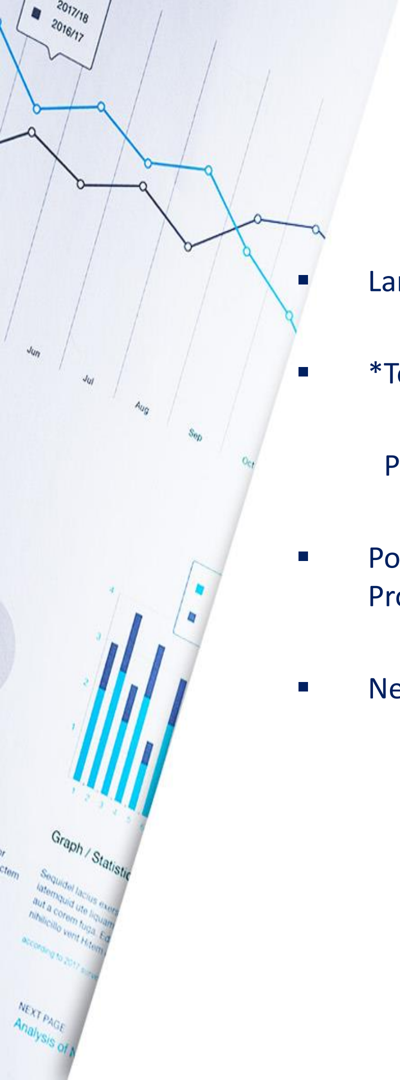
```
[143 110]]
              precision    recall  f1-score   support

           0       0.48      0.56      0.52       239
           1       0.51      0.43      0.47       253

   micro avg       0.49      0.49      0.49       492
   macro avg       0.50      0.50      0.49       492
eighted avg       0.50      0.49      0.49       492
```

Model Performance Analysis

- 7 Biotech Stock Stock Sample: How do features extracted from SEC documents with TfidfVectorizer Predict Daily Abnormal Stock Returns:

- Total Observations: 984:

  Pipeline Model Selection: Max Overall Score Consistently at 67%



Model Performance Analysis

- Large Random Sample of Many More stocks
- *Total Observations: ~50k obs, 50% TTS Splt

  Pipeline Model Selection: Max Score Consistently at 68%

- Positive N-Grams: “technology” “innovation” “Growth” “Share Repurchase Program”
- Negative N-Grams: “deferred” “cash restrictions” “resignation”

Alternative: Model Performance Analysis

- Alternative model performance analysis using time series data derived from text analysis. Time period 2010-2019.

- **Linear models:**

```
features = ["goo_trends_value", "polarity",
        "no_articles","polarity_biotec","subjectivity_biotec","no_articles_biotec",
        "polarity_pharma","subjectivity_pharma","no_articles_pharma"]

X = data[features]
y = data["mthly_ave_resid"]
```

- Issue: standardize target variable. Pipeline() and transformerTargetRegressor() use to normalize using MinMaxScaler()
- Models: LinearRegression(), Lasso(alpha = 0.1), ElasticNet(random_state = 0)

- **Scores**: **R^2 = 0.02**

- **Classification models:** if monthly average residual return is +/- 1 std away from the mean = 1, if not = 0.   Group by ‘ticker’.

- Models: SVC(gamma='auto'), LinearSVC(), SGDClassifier(max_iter=100, tol=1e-3), KNeighborsClassifier(),
- LogisticRegression(solver='lbfgs'), BaggingClassifier(), ExtraTreesClassifier(n_estimators=100),
- RandomForestClassifier(n_estimators=10), GaussianNB()

- **Scores:** BaggingClassifier: 0.81
- ExtraTreesClassifier: 0.87
- RandomForestClassifier: 0.83

- **Cross-validation:** done manually using expanding window CV.

- **Sample increased:** 2000-2019.

- **Security specific (pure Time Series)**: Done for each security by itself, not enough data.

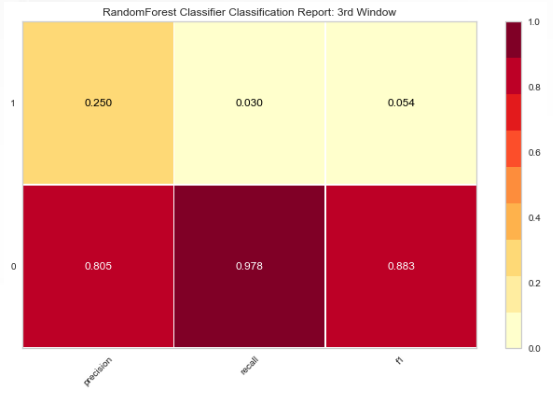
Visualization

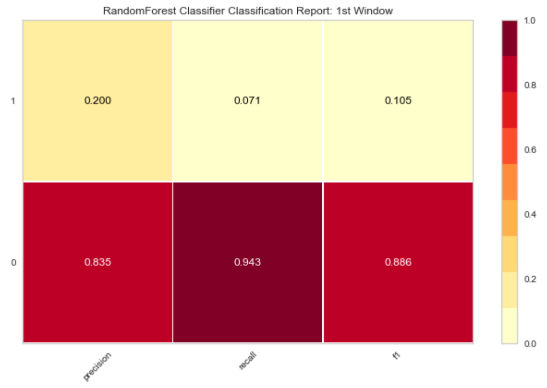
Best performance: Classification for extended sample using the RandomForest() Classifier. However, classification scores better for less important movements. The classification model has issues to better predict "larger" movements in the stock residual returns.

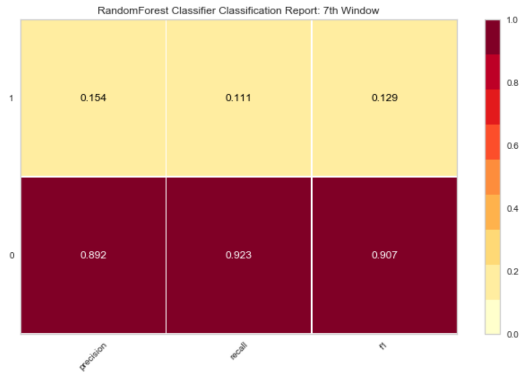
1st window – train in data 2000-2011. Test in data for 2012-2013.

3rd window – train in data 2000-2013. Test in data for 2014-2015.

7th window – train in data 2000-2017. Test in data for 2018-2019.

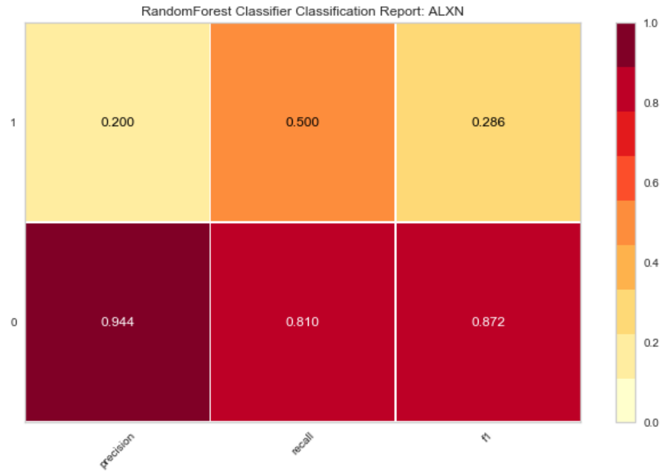


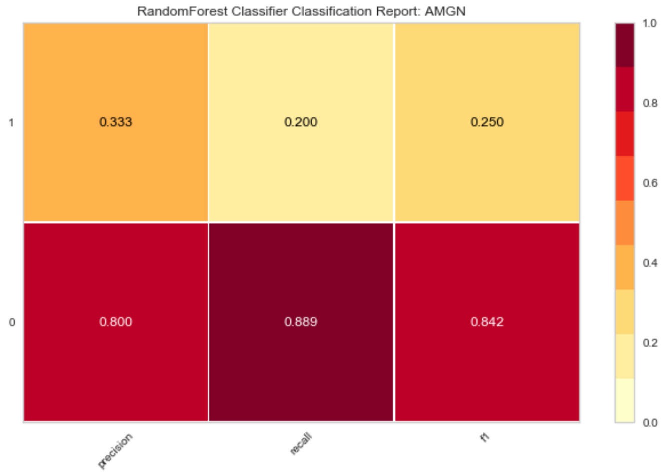




Visualization

For each security, results are poor are consistent as when using pooled time series (panel data).
7th window – train in data 2000-2017. Test in data for 2018-2019.





Key Findings

- *Regardless of sample size, maximum scores for most models based on n-grams from SEC filings max out at under 70%*
- *F1 scores are low, like in the published literature*
- *Abnormal returns exhibit greater variation on the days of SEC filings*
- *Findings are robust to cross-validation*
- *Classification seems to perform better – however, when looking at per-class model performance, models do a poor job in predicting "large" (important) movements*
- *Greater targeted and granular features more likely to improve accuracy than more instances*
- *Data is not cheap, but it is necessary to increase features with more events data*

Appendix

References:

1) Michael Hagenau, Michael Liebmann, Dirk Neumann, “Automated news reading: Stock price prediction based on financial news using context-capturing features”, Decision Support Systems, 20 February 2013

2) Adam Atkins, Mahesan Niranjan, Enrico Gerd, “Financial news predicts stock market volatility better than close price”, Science Direct, 8 February 2018

Appendix

<Add a tag line for this table>

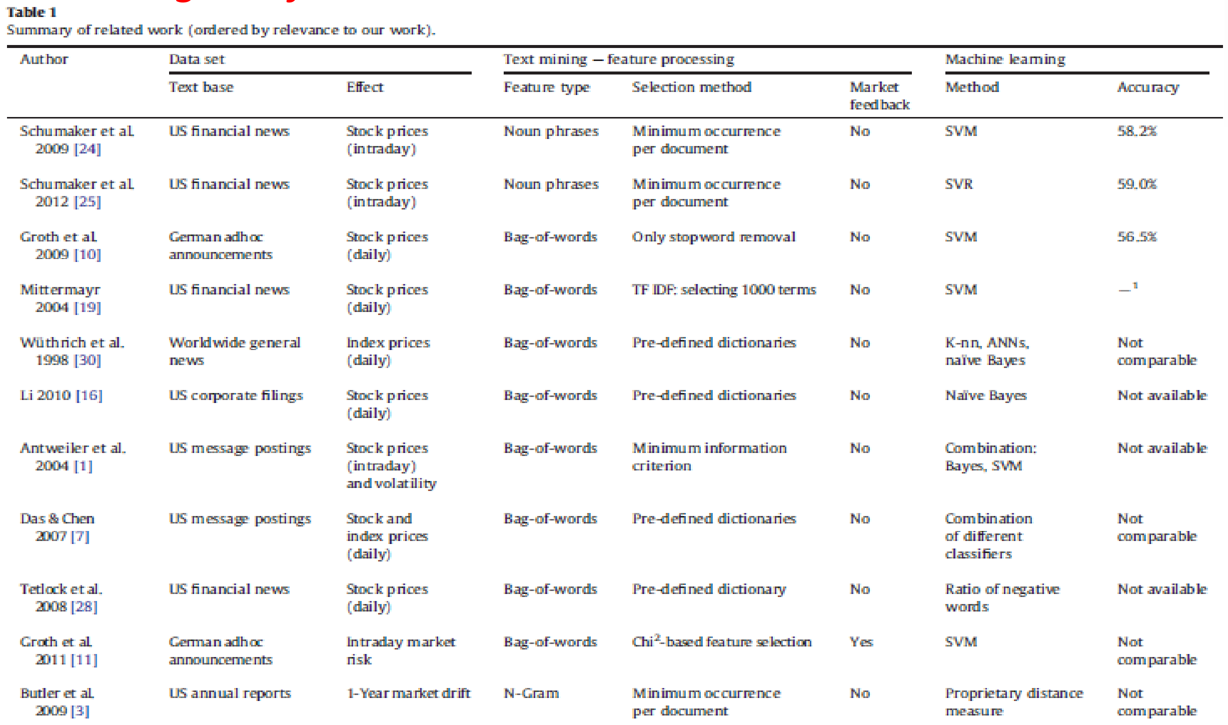


Table 1
Summary of related work (ordered by relevance to our work).

Author	Data set		Text mining — feature processing			Machine learning	
	Text base	Effect	Feature type	Selection method	Market feedback	Method	Accuracy
Schumaker et al. 2009 [24]	US financial news	Stock prices (intraday)	Noun phrases	Minimum occurrence per document	No	SVM	58.2%
Schumaker et al. 2012 [25]	US financial news	Stock prices (intraday)	Noun phrases	Minimum occurrence per document	No	SVR	59.0%
Groth et al. 2009 [10]	German adhoc announcements	Stock prices (daily)	Bag-of-words	Only stopword removal	No	SVM	56.5%
Mittermayr 2004 [19]	US financial news	Stock prices (daily)	Bag-of-words	TF IDF: selecting 1000 terms	No	SVM	—[1]
Wüthrich et al. 1998 [30]	Worldwide general news	Index prices (daily)	Bag-of-words	Pre-defined dictionaries	No	K-nn, ANNs, naïve Bayes	Not comparable
Li 2010 [16]	US corporate filings	Stock prices (daily)	Bag-of-words	Pre-defined dictionaries	No	Naïve Bayes	Not available
Antweiler et al. 2004 [1]	US message postings	Stock prices (intraday) and volatility	Bag-of-words	Minimum information criterion	No	Combination: Bayes, SVM	Not available
Das & Chen 2007 [7]	US message postings	Stock and index prices (daily)	Bag-of-words	Pre-defined dictionaries	No	Combination of different classifiers	Not comparable
Tetlock et al. 2008 [28]	US financial news	Stock prices (daily)	Bag-of-words	Pre-defined dictionary	No	Ratio of negative words	Not available
Groth et al. 2011 [11]	German adhoc announcements	Intraday market risk	Bag-of-words	Chi^2-based feature selection	Yes	SVM	Not comparable
Butler et al. 2009 [3]	US annual reports	1-Year market drift	N-Gram	Minimum occurrence per document	No	Proprietary distance measure	Not comparable

Appendix

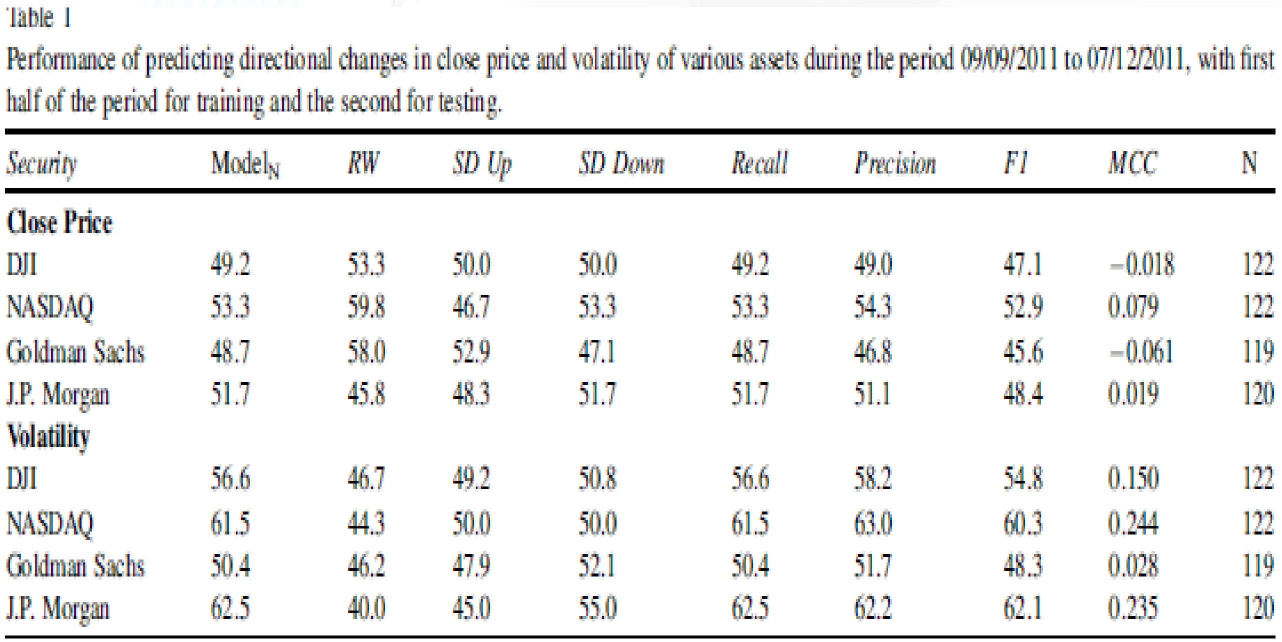


Table 1

Performance of predicting directional changes in close price and volatility of various assets during the period 09/09/2011 to 07/12/2011, with first half of the period for training and the second for testing.

Security	Model$_N$	*RW*	*SD Up*	*SD Down*	*Recall*	*Precision*	*F1*	*MCC*	N
Close Price									
DJI	49.2	53.3	50.0	50.0	49.2	49.0	47.1	−0.018	122
NASDAQ	53.3	59.8	46.7	53.3	53.3	54.3	52.9	0.079	122
Goldman Sachs	48.7	58.0	52.9	47.1	48.7	46.8	45.6	−0.061	119
J.P. Morgan	51.7	45.8	48.3	51.7	51.7	51.1	48.4	0.019	120
Volatility									
DJI	56.6	46.7	49.2	50.8	56.6	58.2	54.8	0.150	122
NASDAQ	61.5	44.3	50.0	50.0	61.5	63.0	60.3	0.244	122
Goldman Sachs	50.4	46.2	47.9	52.1	50.4	51.7	48.3	0.028	119
J.P. Morgan	62.5	40.0	45.0	55.0	62.5	62.2	62.1	0.235	120

From Adam Atkins, Mahesan Niranjan, Enrico Gerd, “Financial news predicts stock market volatility better than close price”, Science Direct, 8 February 2018

Questions?