Brian Acquafredda
MDS564 – Spring 2023
Elmhurst University – Final Project

STOCK MARKET TEXT ANALYTICS

USING STOCK MARKET NEWSFEEDS TO PREDICT DAILY GAINS/LOSSES FOR THE DOW JONES INDUSTRIAL AVERAGE

In this project, the goal is to demonstrate the potential of human emotions and feelings through the written word regarding pre-market newsfeeds of the present and recent past. The data used in the project uses a historical dataset of the market open, close, high, low, a target of 1 for gain, and 0 for loss, and lastly daily sentiment analysis scores using the compounded polarity, to help predict the daily market outcome through recent daily and consistent pre-market newsfeeds which begins before the markets open for the day. The idea is that those who publish key topics of what the markets might consider important news, have keen insights and knowledge on the outcome of the markets for the day.

For our perspective, and assumption for this project is to assume that the financial markets represents chaos theory. The concept is that you can beat the market, but you cannot beat the market all the time. The financial markets run on numerous chaotic systems that influences many conditions, which make the market currently (*1) almost impossible to predict. Prediction relies on some form of mathematics in the traditional sense of how the market will perform. However, it is interesting, within Chaos Theory, there is a term which describes human emotion, and in mathematics, this human emotion is call the *strange attraction*. The goal of the project is to measure the accuracy of News Sentiment Scores from Sentiment Analysis in the Data Science world of Text Analytics, to see if this strange attraction, could be in conjunction with more linear and mathematical financial prediction to make something more accurate overall. However this project concludes on looking at this *Strange Attraction*.

The analysis uses a running study of the Daily News Sentiment Index, from a paper and dataset called “Measuring News Sentiment,” by Adam Hale Shapiro, Moritz Sudhof, and Daniel Wilson from the Federal Reserve Bank of San Francisco. (*2). This dataset will be the training dataset that will analyze recent pre-market newsfeeds. The dataset takes Sentiment Scores from several daily news articles, and as of April 2021, it takes from 24 different newspapers, which are the following, Atlanta Journal-Constitution, Boston Herald, Dallas Morning News, Detroit Free Press, Houston Chronicle, New York Post, New York Times, Palm Beach Post, Philadelphia Inquirer, Providence Journal, San Francisco Chronicle, San Jose Mercury News, Seattle Times, St. Louis Post-Dispatch, St. Paul Pioneer Press, Star Ledger, Star Tribune, The Tennessean, Arizona Republic, Arkansas Democrat Gazette, Denver Post, Times-Picayune, Wall Street Journal, and lastly the Washington Post. The entire dataset has a minimum corpus of 238,685 news articles.

First, our test will be taking “Top 5 Things to Know Before the Markets Open” from CNBC’s daily pre-market daily newsfeed, with the following assumptions, CNBC is the top business news network in television, and it is a well-known and main-stream news organization. Even though the Daily

News Sentiment Index uses several newspapers, our assumption is the CNBC's premarket newsfeed is a good general aggregation of the prior, and will assume that it is similar for now.

The idea, we know that Sentiment analysis of Twitter feeds can predict up and down changes in the Dow Jones Industrial Average (DJIA) closing values with an accuracy of 87.6%. However, can we use Premarket Newsfeeds to predict the Dow Jones Industrial Average closing values for the day? If this was possible, and you could get a good accuracy rating with Machine Learning, this will be feasible to make a living, only if it were so easy. This project uses NTLK's Vader Lexicon package that provides Sentiment Analysis Scores for the Premarket Newsfeeds that we are using.

BUSINESS/DATA UNDERSTANDING

There are 3 main datasets used in this project. 1. CNBC '5 things to know before the Stock Market Opens today', newsfeed that was collected over the past 3 weeks and created specifically for this project that was done independently. 2. Stock Market News Headline Sentiment Analysis, by a dataset provided by the Federal Reserve Bank of San Francisco called the "Daily News Sentiment Index,", 3. Historical Dow Jones Industrial Average for Open, Close, High, Low, Change, and target of 1 if overall gain, or target 0, if overall loss.

Daily New Sentiment Index, "Measuring News Sentiment" ***'Federal Reserve Bank of San Francisco'**

The Daily News Sentiment Index, which is from "Measuring News Sentiment" from the article from the Federal Reserve Bank of San Francisco, using an average since 1980 of several newspapers and judges around 25 of them consistently through this period of time.

	$s_a^i = f_{t(a)}^i + f_{p(a),j(a)}^i$	$+ \varepsilon_a^i$
Where		
o is the positivity score for article	s_a^i	
o is a sample-day (t) fixed effect	$f_{t(a)}^i$	
o is a newspaper*type fixed effect	$f_{p(a),j(a)}^i$	

What this model does is take lexicons and use its own model that looks at the sentiment in economic news, not just daily newsfeeds during pre-market or intraday market. It uses the Loughran and McDonald dictionary and Harvard General Inquirer dictionary. For NTLK, VADER package, we will use the default, however it is understood that these two are different. This model does score with weighted averages that we do not use in VADER for the TEST dataset. It would be interesting to see if we can improve the accuracy of the prediction scores and values if we used a weight on the VADER score that we later use in the TEST dataset. However, please note that this project does not yet do that.

Below is one of the dictionaries and lexicon scores and rules for the Sentiment Analysis Scoring used for the Daily News Sentiment Index.

Word	Sequence	Word Cou	Word Proportion	Average Proportion	Std Dev	Doc Count	Negative	Positive	Uncertainty	Litigious	Constraini	Superfluou	Interesting	Modal	Irr_Verb	Harvard_IV	Syllables	Source
AARDVARK	1	275	1.60E-08	1.31E-08	3.67E-06	82	0	0	0	0	0	0	0	0	0	0	2	12of12inf
AARDVARKS	2	3	1.75E-10	1.03E-11	1.01E-08	1	0	0	0	0	0	0	0	0	0	0	2	12of12inf
ABACI	3	8	4.66E-10	1.47E-10	6.40E-08	7	0	0	0	0	0	0	0	0	0	0	3	12of12inf
ABACK	4	6	3.50E-10	1.76E-10	7.21E-08	6	0	0	0	0	0	0	0	0	0	0	2	12of12inf
ABACUS	5	6,729	3.92E-07	3.75E-07	3.45E-05	845	0	0	0	0	0	0	0	0	0	0	3	12of12inf
ABACUSES	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	12of12inf

Example of the DNSI Code below with the Loughran and McDonald Dictionary:

```
def load_lm_lexicon():
    # Loughran McDonald
    fn = os.path.join(lexicon_dir, "LoughranMcDonald_2016.csv")
    reader = csv.DictReader(open(fn))
    words2weights = {}
    for r in reader:
        pos_score = 1. if r['Positive'] != "0" else 0.
        neg_score = 1. if r['Negative'] != "0" else 0.
        sentiment_score = pos_score - neg_score
        w = r['\ufeffWord'].lower() # weird header in this file
        #w = r['Word'].lower()
        if sentiment_score:
            words2weights[w] = sentiment_score
    return words2weights
```

Vader Lexicon using NTLK in Python with CNBC '5 things to know before the Stock Market Opens today',

VADER Formulation for the Compounded Polarity Score.

I collected 15 days' worth of the Daily Premarket Newsfeed. It would be much more interesting if more were collected, however the automated idea of web scrapping, which I provide as interesting code that I did and tried with this project is provided. However, the result was not conclusive, or simply did not work with the type of web pages were are trying to get data from.

$$\frac{x}{\sqrt{x^2 + \alpha}}$$

DATA PREPERATION

Data preparation phase with our good old friend CRISP, I went through the motions of tokenization, look at a corpus vs. document, looking at bigrams, a bag of words, reviewing the term frequency, removing stopwords, punctuation, numbers, lemmanation, conversion of text to lower case. One area that was planned, and was done, but was not necessary, after completion, and was a surprising result was that VADER and NTLK does a nice job of preprocessing the text, without doing the necessary steps one-by-one.

Merging of the 2 datasets and preparing the project dataset as the TEST dataset. A massive V-LOOKUP in Excel was used to merge the two historical datasets which will be used as the training dataset for the machine learning model that we will use for the prediction. However, we are using the text Sentiment Scores provided from the 'News Sentiment Dataset,' which is the Daily News Sentiment Index from the Federal Reserve Bank of San Francisco. Removal of Dates on Stock Market Holidays from the News Sentiment Dataset (Daily News Sentiment Index). The dataset came with everyday of the week since 1980. This was trimmed and merged other datasets to make it complete. Data was removed for weekends and the following years with the following Holidays, 2012 - 2023 Stock Market Holidays

Data preparation phase with our good old friend CRISP, I went through the motions of tokenization, look at a corpus vs. document, looking at bigrams, a bag of words, reviewing the term frequency, removing stopwords, punctuation, numbers, lemmanation, conversion of text to lower case. Surprisingly, which was overlooked, was that NTLK and the Vader package and its Sentiment Polarity Score automatically does a nice job of preprocessing the data, which makes preprocessing not necessary with NTLK and Vader. I also added Open Close, High Low, Change, and a target to the DNSI. Also created my own dataset with CNBCs premarket newsfeeds.

News Sentiment Dataset, created => 07/01/2012 – 02/24/2023 Dataset

This was created by Merging the Daily News Sentiment Index, which only had a date and Sentiment Polarity Score. Used Historical Datasets from MarketWatch, which only lets you pull 1 year's worth of Dow Jones Industrial Average data.

date	News Sentiment	Open	Close	High	Low	Change	target
07/02/12	-0.219829142	12879.71	12871.39	12902.12	12795.48	-8.32	0
07/03/12	-0.228132448	12868.06	12943.82	12946.2	12845.28	75.76	1
07/05/12	-0.207128756	12941.85	12896.67	12961.3	12852.24	-45.18	0
07/06/12	-0.208013396	12889.4	12772.47	12889.4	12702.99	-116.93	0
07/09/12	-0.195998693	12772.02	12736.29	12772.02	12686.57	-35.73	0
07/10/12	-0.189461006	12733.87	12653.12	12830.29	12606.91	-80.75	0
07/11/12	-0.195956946	12653.04	12604.53	12661.97	12534.33	-48.51	0
07/12/12	-0.214600266	12602.71	12573.27	12630.64	12492.25	-29.44	0
07/13/12	-0.234180128	12573.73	12777.09	12784.73	12573.04	203.36	1
07/16/12	-0.232607908	12776.33	12727.21	12779.58	12690.05	-49.12	0

This was accomplished using a massive VLOOKUP with another dataset. Example of how a massive VLOOKUP can be used in excel below if it might be useful to anyone else.

1	2	3	4	5	6	7	8	9	10
date	**News Sentiment**	**target**	**Top1**	**Top2**	**Top3**	**Top4**	**Top5**	**Top6**	**Top7**
1/1/2000	0.281809731	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1/2/2000	0.29749183	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1/3/2000	0.308622721	0	A 'hindrance tc	Scorecard	Hughes' in:	Jack gets h	Chaos as N	Depleted L	Hungry Spu
1/4/2000	0.291098159	0	Scorecard	The best la	Leader: Ge	Cheerio, b	The main r	Has Cubie	Has Cubie
1/5/2000	0.305734373	0	Coventry caugl	United's ri	Thatcher is	Police help	Tale of Tra	England or	Pakistan re
1/6/2000	0.318497865	1	Pilgrim knows I	Thatcher f	McIlroy ca	Leicester b	United bra	Auntie bac	Shoaib app
1/7/2000	0.309894685	1	Hitches and Hc	Beckham c	Breast can	Alan Parke	Guardian r	Hollywooc	Ashes and
1/8/2000	0.301847351	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1/9/2000	0.286981228	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1/10/2000	0.280951615	1	Fifth round dra	BBC unveil	Second Div	European	Third Divis	Welfare cc	Ferguson p
1/11/2000	0.278537152	1	Man Utd 2 - 0 S	How North	Buoyant B	Tranmere	United sit	Queen's Pa	Waugh hits
1/12/2000	0.291446413	0	Newcastle see	Liverpool a	Highlander	Edwards' p	Chelsea ga	Taylor sett	Tenth top-
1/13/2000	0.302617636	1	Bungling officia	And in the	United put	England ag	Donald po	Adams sta	Money mc
1/14/2000	0.304831842	1	Pompey plump	Roma und	Prenton Pa	OK, I didn't	Chelsea tu	Top storey	West Indie
1/15/2000	0.341403395	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1/16/2000	0.329100193	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1/17/2000	0.331310601	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A

1	2	3	4	5	6	7	8	9
date	**News Sentiment**	**target**	**Top1**	**Top2**	**Top3**	**Top4**	**Top5**	**Top6**
36526	0.281809730871954	=VLOOKUP($A3,Text!$A:$AA,Text!B$1,0	=VLOOKUP($A3,Text!$A:$AA,	=VLOOKUP($A3,Text!	=VLOOKUP($A3,Text!	=VLOOKUP($A3,Text!	=VLOOKUP($A3,Text!	=VLOOKUP($A3,Text!
36527	0.297491829811852	=VLOOKUP($A4,Text!$A:$AA,Text!B$1,0	=VLOOKUP($A4,Text!$A:$AA,	=VLOOKUP($A4,Text!	=VLOOKUP($A4,Text!	=VLOOKUP($A4,Text!	=VLOOKUP($A4,Text!	=VLOOKUP($A4,Text!
36528	0.308622721159146	=VLOOKUP($A5,Text!$A:$AA,Text!B$1,0	=VLOOKUP($A5,Text!$A:$AA,	=VLOOKUP($A5,Text!	=VLOOKUP($A5,Text!	=VLOOKUP($A5,Text!	=VLOOKUP($A5,Text!	=VLOOKUP($A5,Text!
36529	0.291098158985108	=VLOOKUP($A6,Text!$A:$AA,Text!B$1,0	=VLOOKUP($A6,Text!$A:$AA,	=VLOOKUP($A6,Text!	=VLOOKUP($A6,Text!	=VLOOKUP($A6,Text!	=VLOOKUP($A6,Text!	=VLOOKUP($A6,Text!
36530	0.305734373198226	=VLOOKUP($A7,Text!$A:$AA,Text!B$1,0	=VLOOKUP($A7,Text!$A:$AA,	=VLOOKUP($A7,Text!	=VLOOKUP($A7,Text!	=VLOOKUP($A7,Text!	=VLOOKUP($A7,Text!	=VLOOKUP($A7,Text!

3	4
target	**Top1**
=VLOOKUP($A3,Text!$A:$AA,Text!B$1,0)	=VLOOKUP($A3,Text!$A:$AA,Text!C$1,0)
=VLOOKUP($A4,Text!$A:$AA,Text!B$1,0)	=VLOOKUP($A4,Text!$A:$AA,Text!C$1,0)
=VLOOKUP($A5,Text!$A:$AA,Text!B$1,0)	=VLOOKUP($A5,Text!$A:$AA,Text!C$1,0)
=VLOOKUP($A6,Text!$A:$AA,Text!B$1,0)	=VLOOKUP($A6,Text!$A:$AA,Text!C$1,0)
=VLOOKUP($A7,Text!$A:$AA,Text!B$1,0)	=VLOOKUP($A7,Text!$A:$AA,Text!C$1,0)

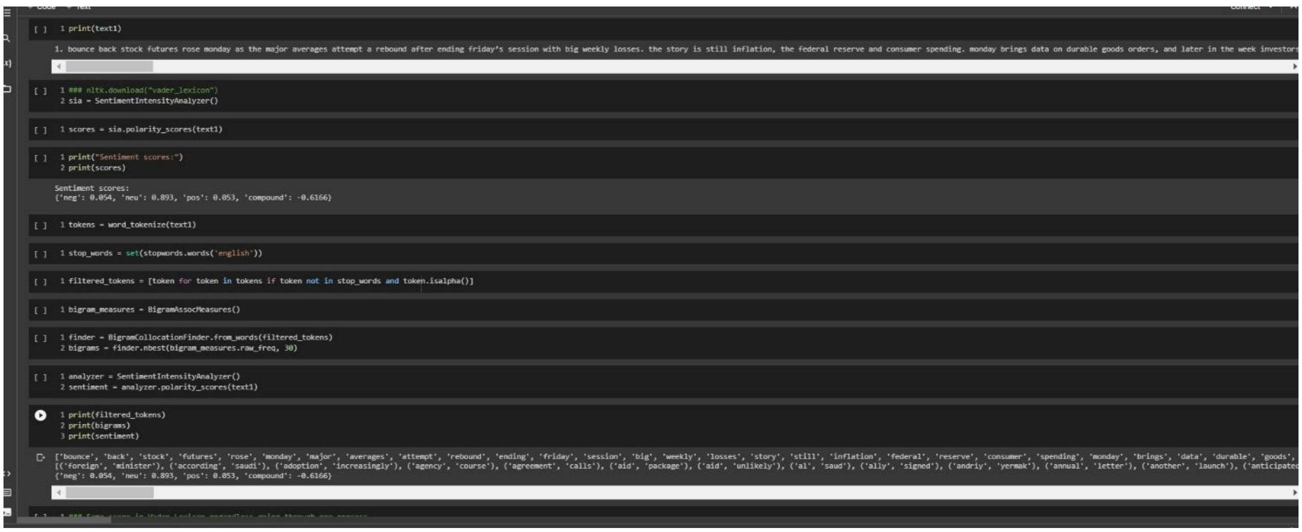
1	2	3	4	5	6	7	8	9	10	11	12	13	14
Date	Label	Top1	Top2	Top3	Top4	Top5	Top6	Top7	Top8	Top9	Top10	Top11	Top12
1/3/2000	0	A 'hindran	Scorecard	Hughes' in:	Jack gets h	Chaos as N	Depleted L	Hungry Spu	Gunners sc	Derby raise	Southgate	Hammers	Saints part
1/4/2000	0	Scorecard	The best la	Leader: Ge	Cheerio, b	The main r	Has Cubie	Has Cubie	Has Cubie	Hopkins 'fu	Has Cubie	A tale of tv	I say what
1/5/2000	0	Coventry c	United's ri	Thatcher is	Police help	Tale of Tra	England or	Pakistan re	Cullinan cc	McGrath p	Blair Witch	Pele turns	Party divid
1/6/2000	1	Pilgrim knc	Thatcher f	McIlroy ca	Leicester b	United bra	Auntie bac	Shoaib app	Hussain hu	England's c	Revenge is	Our choice	Profile of f
1/7/2000	1	Hitches an	Beckham c	Breast can	Alan Parke	Guardian r	Hollywooc	Ashes and	Whingers -	Alan Parke	Thuggery,	Met faces	Everton fa

CNBC's 5 Things to Know Before the Markets Open Today, Created Dataset

Below is how I prepared this dataset, which will be the TEST dataset for Machine Learning. It would be a much more manual ML process by measuring the real accuracy of the TEST, rather than do an 80/20 split. This seems much more interesting comparing the results side by side from the predicitons in Machine Learning versus the results in reality.

Id	Date	Market_Open	Adj _Market_Close	Daily_Change	target	Url	Text
1	2/27/2023	32906.16	32889.09	-17.07	0	https://www.cnbc.com/2023/02/27/5-things-to-know-before-the-stock-market-opens-monday-february-27.html	1. Bounce
2	2/28/2023	32873.47	32656.7	-216.77	0	https://www.cnbc.com/2023/02/28/5-things-to-know-before-the-stock-market-opens-tuesday-february-28.html	1. So long,
3	3/1/2023	32656.37	32661.84	5.47	1	https://www.cnbc.com/2023/03/01/5-things-to-know-before-the-stock-market-opens-wednesday-march-1.html	1. In like a
4	3/2/2023	32780.97	33003.57	222.6	1	https://www.cnbc.com/2023/03/02/5-things-to-know-before-the-stock-market-opens-thursday-march-2.html	1. Off to a
5	3/3/2023	33076.33	33390.97	314.64	1	https://www.cnbc.com/2023/03/03/5-things-to-know-before-the-stock-market-opens-friday-march-3.html'	1. What th
6	3/6/2023	33425.32	33431.44	6.12	1	https://www.cnbc.com/2023/03/06/5-things-to-know-before-the-stock-market-opens-monday-march-6.html'	1. Entering
7	3/7/2023	33428.31	32856.46	-571.85	0	https://www.cnbc.com/2023/03/07/5-things-to-know-before-the-stock-market-opens-tuesday-march-7.html	1. Those st
8	3/8/2023	32872.08	32798.4	-73.68	0	https://www.cnbc.com/2023/03/08/5-things-to-know-before-the-stock-market-opens-wednesday-march-8.html	1. Powell b
9	3/9/2023	32876.83	32254.86	-621.97	0	https://www.cnbc.com/2023/03/09/5-things-to-know-before-the-stock-market-opens-thursday-march-9.html	1. March n
10	3/10/2023	32185.14	31909.64	-275.5	0	https://www.cnbc.com/2023/03/10/5-things-to-know-before-the-stock-market-opens-friday-march-10.html	1. Jobs ma
11	3/13/2023	31819.93	31819.14	-0.79	0	https://www.cnbc.com/2023/03/13/5-things-to-know-before-the-stock-market-opens-monday-march-13.html	1. New we
12	3/14/2023	32055.29	32155.4	100.11	1	https://www.cnbc.com/2023/03/14/5-things-to-know-before-the-stock-market-opens-tuesday-march-14.html	1. Stress ar
13	3/15/2023	31759.87	31874.57	114.7	1	https://www.cnbc.com/2023/03/15/5-things-to-know-before-the-stock-market-opens-wednesday-march-15.html	1. More tu
14	3/16/2023	31827.65	32246.55	418.9	1	https://www.cnbc.com/2023/03/16/5-things-to-know-before-the-stock-market-opens-thursday-march-16.html	1. What's r
15	3/17/2023	32217.32	31861.98	-355.34	0	https://www.cnbc.com/2023/03/17/5-things-to-know-before-the-stock-market-opens-friday-march-17.html	1. Coming

After loading NTLK, I prepared the data into Google Colab/Python. I took the text and saved it into a data frame for each article. I used lower text function, stopwords function, then run it through the VADER package in NTLK. The surprising result that I noticed after a few preprocessing steps in the code, is that NTLK and Vader does a nice job of preprocessing the text, rather than do the preprocess steps one by one.



One controversial method I used in the coding, was not use the dataset in Python, but added the raw text into Google Colab/Python and saved it as a data frame, rather than using it in the dataset. This is because I used the dataset several times, and had trouble with the file formatting, that I was unable to normally process it and use it like I would regularly. I was running into issues where, I think the copy and paste from the test messes up the excel file, and then when you try and run it in Python, it is not working properly.

```
print("Sentiment scores:")
print(scores)

Sentiment scores:
{'neg': 0.088, 'neu': 0.842, 'pos': 0.07, 'compound': -0.8494}

### 03/16/2023
### url = 'https://www.cnbc.com/2023/03/16/5-things-to-know-before-the-stock-market-opens-thursday-march-16.html'

text14 = '1. What's next in this wild week? Two more trading days to go in this chaotic week that's been driven by turmoil in banks. Credit Suisse, the current poster child for the upheaval, took up the Swiss central bank's offer of a lifeline. Will that calm

text14 = text14.lower()

scores = sia.polarity_scores(text14)

print("Sentiment scores:")
print(scores)

Sentiment scores:
{'neg': 0.103, 'neu': 0.782, 'pos': 0.115, 'compound': 0.5159}

### 03/17/2023
### url = 'https://www.cnbc.com/2023/03/17/5-things-to-know-before-the-stock-market-opens-friday-march-17.html'

text15 = '1. Coming out ahead It looks like major stock averages could come out ahead this week, on pace for a winning week despite the turmoil in the global banking sector. Through Thursday, the Dow Jones Industrial Average is up 1.06%, the S&P 500 has risen

text15 = text15.lower()

scores = sia.polarity_scores(text15)

print("Sentiment scores:")
print(scores)

Sentiment scores:
{'neg': 0.045, 'neu': 0.892, 'pos': 0.064, 'compound': 0.7341}
```

I then took the Sentiment Analysis scores, which is the compounded score, and added it to the dataset as the results/scores for the TEST dataset, which we will be looking at its accuracy from reality.

MODELING

The Modeling was done with the intention of a target variable, a binary result, which could easily be used for Logistic Regression, or more advanced Classification, however our PyCaret looks at the best models for us.

TRAINING –

The Training Dataset was primarily the Daily News Sentiment Index Score, including the Date, Open, Close, High, Low, Change, and target variable. The target variable is 1 for if the closing value was positive and the target variable 0 is if the closing value was negative for the Dow Jones Industrial Average, respectfully.

TEST

The TEST dataset was the dataset that I put together on my own from the VADER Sentiment Polarity Compounded Scores, alongside the actual/reality of the stock market results for those days. We use the TEST dataset to see how accurately it can predict reality on the daily CNBC Premarket Newsfeed.

```python
import pandas as pd
```

```python
from google.colab import files
uploaded = files.upload()
```

No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving train_text.csv to train_text.csv
```

```python
from google.colab import files
uploaded = files.upload()
```

No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving test_text.csv to test_text.csv
```

```python
train = pd.read_csv('train_text.csv', usecols=['News Sentiment','Open','Close','High','Low','Change','target'])
test = pd.read_csv('test_text.csv', usecols=['News Sentiment','Open','Close','High','Low','Change'])
```

```python
from pycaret.classification import *
s = setup(data=train,target='target')
```

	Description	Value
0	Session id	8399
1	Target	target
2	Target type	Binary
3	Original data shape	(2679, 7)
4	Transformed data shape	(2679, 7)
5	Transformed train set shape	(1875, 7)
6	Transformed test set shape	(804, 7)
7	Numeric features	6
8	Preprocess	[illegible]
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	StratifiedKFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name

We include all the necessary columns for Machine Learning, however we stayed away from the dates. Not that dates would cause time series, this is not a time series project.

EVALUATION

```python
best = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
xgboost	Extreme Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
lr	Logistic Regression	0.9984	1.0000	0.9990	0.9980	0.9985	0.9968	0.9968	[illegible]
et	Extra Trees Classifier	0.9691	0.9959	0.9722	0.9704	0.9712	0.9378	0.9380	[illegible]
lda	Linear Discriminant Analysis	0.9088	0.9943	0.9940	0.8599	0.9216	0.8141	0.8276	[illegible]
ridge	Ridge Classifier	0.9077	0.0000	0.9930	0.8590	0.9207	0.8119	0.8254	[illegible]
nb	Naive Bayes	0.8730	0.9905	0.9921	0.8140	0.8938	0.7401	0.7638	[illegible]
knn	K Neighbors Classifier	0.8688	0.9401	0.8955	0.8652	0.8797	0.7354	0.7367	[illegible]
qda	Quadratic Discriminant Analysis	0.8565	0.9806	0.9761	0.8131	0.8826	0.7047	0.7360	[illegible]
svm	SVM - Linear Kernel	0.7806	0.0000	0.8306	0.8546	0.7717	0.5545	0.6095	[illegible]
dummy	Dummy Classifier	0.5360	0.5000	1.0000	0.5360	0.6979	0.0000	0.0000	[illegible]

It was very interesting as how the Decision Tree Classifier, which is one of more simple Data Model Algorithms is selected as one of the best. From previous work, the more complex and computational intense algorithms such as Gradient Boosting seem to always be one of the best, alongside Random Forest.

Once the results were looked at and analyzed, the current model and how the data was prepared only provided a .5333 Accurate Result, even though Open, Close, High, Low, Change, and Target was used in the training. Did this modeling make sense to begin with? The simple answer is no. This is only slightly better than a 50/50 toss-up. However this project is not about Bayesian Inference.

Upon Reflection of the results, there were a couple issues of the modeling that did not make sense. 1. If I have Open, Close, High, Low, and Change for the day, how could that make sense, as we already have the data that tells us what happened, how would those values be useful in the training if we are trying to predict? Also, the Daily News Sentiment Index Score is not the same as the default VADER Sentiment Compounded Score. Changes must be made. The first line is the results, the second line is actually what happened, therefore we were 8/15, only 53% accurate.

```
        verbose=False), 'train_text.csv.pkl')
```

```
print(predictions)
```

```
            Open         Close          High           Low      Change  \
0   32906.160156  32889.089844  33189.281250  32814.179688  -17.070000
1   32873.468750  32656.699219  32873.468750  32636.429688 -216.770004
2   32656.369141  32661.839844  32746.150391  32500.710938    5.470000
3   32780.968750  33003.570312  33083.449219  32665.849609  222.600006
4   33076.328125  33390.968750  33405.820312  33008.410156  314.640015
5   33425.320312  33431.441406  33572.218750  33383.468750    6.120000
6   33428.308594  32856.460938  33453.250000  32838.210938 -571.849976
7   32872.078125  32798.398438  32903.441406  32612.699219  -73.680000
8   32876.828125  32254.859375  32990.460938  32190.599609 -621.969971
9   32185.140625  31909.640625  32422.099609  31783.410156 -275.500000
10  31819.929688  31819.140625  32240.349609  31624.869141   -0.790000
11  32055.289062  32155.400391  32306.589844  31805.400391  100.110001
12  31759.869141  31874.570312  31906.470703  31429.820312  114.699997
13  31827.650391  32246.550781  32281.609375  31571.460938  418.899994
14  32217.320312  31861.980469  32217.320312  31728.699219 -355.339996

    News Sentiment  prediction_label  prediction_score
0          -0.6166                 0               1.0
1           0.2023                 0               1.0
2           0.7061                 1               1.0
3           0.5859                 1               1.0
4           0.3612                 1               1.0
5          -0.9584                 1               1.0
6          -0.9535                 0               1.0
7          -0.6788                 0               1.0
8          -0.9830                 0               1.0
9          -0.4939                 0               1.0
10          0.9990                 0               1.0
11         -0.9601                 1               1.0
12         -0.8494                 1               1.0
13          0.5159                 1               1.0
14          0.7341                 0               1.0
```

```
predictions.describe()
```

```
predictions
```

```
### 0,1,1,1,0,0,0,0,0,1.1.1,1,0,0
### 0,0,1,1,1,1,0,0,0,0,0,1,1,1,0
### 1,0,1,1,0,0,1,1,1,0,0,1,1,0,0

8/15
```

```
0.5333333333333333
```

This is the conclusion of the Model. The conclusion is simple, that using CNBC's Pre-Market Newsfeed, '5 Things to Know Before the Markets Open' since 02/27/2023, which has been a very negative news cycle, which has been very bad news about the economy potentially. Therefore

MODELING 2

We want down to a more basic model with the training and test datasets. I removed the Open, Close, High, Low, Kept daily change, and target. For the test dataset I removed Open, Close, High, Low, Kept daily change, but have the daily change precede to the previous day. The logic is that the premarket newsfeeds feeds off the previous days emotions, and results.

TRAIN

date	News Sentiment	Change	target
02/24/23	-0.033027543	-182.27	0
02/23/23	-0.031600946	-21.48	0
02/22/23	-0.020291626	-124.24	0
02/21/23	-0.014879815	-570.1	0
02/17/23	0.00248806	149.68	1
02/16/23	0.024803413	-295.24	0
02/15/23	0.021685863	119.42	1
02/14/23	0.018055353	-104.82	0
02/13/23	-0.005527485	358.54	1
02/10/23	-0.040999878	197.73	1

TEST

Notice how I shifted the Daily Change up 1 cell to reflect the previous day in the TEST. This logic is used as an assumption of reality that the daily Premarket Newsfeeds, feeds partially from the results of the previous day. In the end, the prediction values were much more accurate.

Date	News Sentiment	Change
02/27/23	-0.6166	-182.27
02/28/23	0.2023	-17.07
03/01/23	0.7061	-216.77
03/02/23	0.5859	5.47
03/03/23	0.3612	222.6
03/06/23	-0.9584	314.64
03/07/23	-0.9535	6.12
03/08/23	-0.6788	-571.85
03/09/23	-0.983	-73.68
03/10/23	-0.4939	-621.97
03/13/23	0.999	-275.5
03/14/23	-0.9601	-0.79
03/15/23	-0.8494	100.11
03/16/23	0.5159	114.7
03/17/23	0.7341	418.9

```
import pandas as pd
```

```
from google.colab import files
uploaded = files.upload()
```

No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving train_text_v2.csv to train_text_v2.csv
```

```
from google.colab import files
uploaded = files.upload()
```

No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving test_text_v2.csv to test_text_v2.csv
```

```
train = pd.read_csv('train_text_v2.csv', usecols=['News Sentiment','Change','target'])
test = pd.read_csv('test_text_v2.csv', usecols=['News Sentiment','Change'])
```

```
from pycaret.classification import *
s = setup(data=train,target='target')
```

	Description	Value
0	Session id	8644
1	Target	target
2	Target type	Binary
3	Original data shape	(2679, 3)
4	Transformed data shape	(2679, 3)
5	Transformed train set shape	(1875, 3)
6	Transformed test set shape	(804, 3)
7	Numeric features	2
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	StratifiedKFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name
18	USI	fd06

```
best = compare_models()
```

EVALUATION 2

The second evaluation is has provided much more interesting and successful results, which you would expect from the first evaluation, however many lessons were learned.

```
1 best = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
xgboost	Extreme Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
lr	Logistic Regression	0.9995	1.0000	1.0000	0.9990	0.9995	0.9989	0.9989	[illegible]
knn	K Neighbors Classifier	0.9984	1.0000	0.9980	0.9990	0.9985	0.9968	0.9968	[illegible]
et	Extra Trees Classifier	0.9925	0.9998	0.9950	0.9912	0.9930	0.9850	0.9852	[illegible]
lda	Linear Discriminant Analysis	0.9237	0.9998	1.0000	0.8764	0.9339	0.8449	0.8557	[illegible]
ridge	Ridge Classifier	0.9232	0.0000	1.0000	0.8757	0.9335	0.8438	0.8548	[illegible]
nb	Naive Bayes	0.9168	0.9986	1.0000	0.8667	0.9283	0.8306	0.8433	[illegible]
qda	Quadratic Discriminant Analysis	0.9115	0.9982	0.9990	0.8598	0.9239	0.8196	0.8334	[illegible]
dummy	Dummy Classifier	0.5360	0.5000	1.0000	0.5360	0.6979	0.0000	0.0000	[illegible]

```
[9] 1 predictions = predict_model(best, data=test)
```

Logistic Regression is much more accurate and at the top, yet not perfect as in some of the other data models, however the accuracy is not reality.

```
print(predictions)
```

```
    News Sentiment      Change  prediction_label  prediction_score
0          -0.6166 -182.270004                 0               1.0
1           0.2023  -17.070000                 0               1.0
2           0.7061 -216.770004                 0               1.0
3           0.5859    5.470000                 1               1.0
4           0.3612  222.600006                 1               1.0
5          -0.9584  314.640015                 1               1.0
6          -0.9535    6.120000                 1               1.0
7          -0.6788 -571.849976                 0               1.0
8          -0.9830  -73.680000                 0               1.0
9          -0.4939 -621.969971                 0               1.0
10          0.9990 -275.500000                 0               1.0
11         -0.9601   -0.790000                 0               1.0
12         -0.8494  100.110001                 1               1.0
13          0.5159  114.699997                 1               1.0
14          0.7341  418.899994                 1               1.0
```

```
predictions.describe()
```

	News Sentiment	Change	prediction_label	prediction_score
count	15.000000	15.000000	15.000000	15.0
mean	-0.159280	-51.824009	0.466667	1.0
std	0.753938	290.649231	0.516398	0.0
min	-0.983000	-621.969971	0.000000	1.0
25%	-0.901450	-199.520004	0.000000	1.0
50%	-0.493900	-0.790000	0.000000	1.0
75%	0.550900	107.404999	1.000000	1.0
max	0.999000	418.899994	1.000000	1.0

```
# Prediction ### 0,0,0,1,1,1,1,0,0,0,0,0,1,1,1
# Actual     ### 0,0,1,1,1,1,0,0,0,0,0,1,1,1,0
# Results    ### 1,1,0,1,1,1,0,1,1,1,1,0,1,1,0

11/15
```

```
0.7333333333333333
```

The accuracy of the results were much better, which was more focused on how the previous daily change of the DJIA affected the possible mood and direction of the next day. Even though still a small test dataset, I am more confidence in the modeling. We not have 11/15 accuracy, a 73% accuracy rating.

```
1 best = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
svm	SVM - Linear Kernel	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
xgboost	Extreme Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	[illegible]
lr	Logistic Regression	0.9995	1.0000	1.0000	0.9990	0.9995	0.9989	0.9989	[illegible]
knn	K Neighbors Classifier	0.9984	1.0000	0.9980	0.9990	0.9985	0.9968	0.9968	[illegible]
et	Extra Trees Classifier	0.9925	0.9998	0.9950	0.9912	0.9930	0.9850	0.9852	[illegible]
lda	Linear Discriminant Analysis	0.9237	0.9998	1.0000	0.8764	0.9339	0.8449	0.8557	[illegible]
ridge	Ridge Classifier	0.9232	0.0000	1.0000	0.8757	0.9335	0.8438	0.8548	[illegible]
nb	Naive Bayes	0.9168	0.9986	1.0000	0.8667	0.9283	0.8306	0.8433	[illegible]
qda	Quadratic Discriminant Analysis	0.9115	0.9982	0.9990	0.8598	0.9239	0.8196	0.8334	[illegible]
dummy	Dummy Classifier	0.5360	0.5000	1.0000	0.5360	0.6979	0.0000	0.0000	[illegible]

```
[9] 1 predictions = predict_model(best, data=test)
```

Even though our second attempt in Machine Learning resulted in a much more accurate result of at least 73% accuracy, the overall potential is much more promising. A more weighted, calculated analysis in data preparation, feature engineering, and data modeling would significantly increase the accuracy. Run it through Deep Learning, the accuracy could be in the high 90%. Would it be better than the twitter 87% accuracy, I am unsure at this time.

In conclusion, this does not promise an absolute accurate prediction of how well the premarket newsfeed will forecast the daily close value. However, in regard to Chaos Theory, the premarket newsfeed, might be a strong suspect for the Strange Attractor, as it is dependent on the previous day's emotions. Thank you.

References and Sources

1. Chaos Theory

https://medium.datadriveninvestor.com/chaos-theory-and-market-predictions-9e03ec47d67e

2. Federal Reserve Bank of San Francisco

https://www.frbsf.org/economic-research/indicators-data/daily-news-sentiment-index/

3. MarketWatch – DJIA Historical Open Close

https://www.marketwatch.com/investing/index/djia/download-data?startDate=7/5/2016&endDate=3/30/2017

4. Sentiment Analysis for Stock Price Prediction in Python

https://towardsdatascience.com/sentiment-analysis-for-stock-price-prediction-in-python-bed40c65d178

5. A Short Introduction to VADER

https://towardsdatascience.com/an-short-introduction-to-vader-3f3860208d53

6. Understanding Lexicon Based Sentiment Analysis

https://www.knime.com/blog/lexicon-based-sentiment-analysis

7. NLP In the Stock Market

https://towardsdatascience.com/nlp-in-the-stock-market-8760d062eb92

news_sentiment_merged_data_train.xlsx

Final_Project_MDS564_Brian_Acquafredda (1

Final_Project_MDS564_Machine_Learning_Re

Final_Project_MDS564_Machine_Learning_Re

Project_MDS564_Pre_Market_Text_Analytics

Project_Tokenize_NTLK_v1.ipynb

Project_MDS564_Check_Dataset.ipynb

train_text.xlsx

test_text.xlsx

Premarket_Newsfeeds_v2.csv

Replication Code - 03.26.2023.xlsx

LoughranMcDonald_2016.xlsx

STOCK MARKET TEXT ANALYTICS.pptx