TRUST THE PROCESS

POWERING THROUGH THE 80/20 RULE

PROJECT OUTLINE

MOTIVATION AND SUMMARY

WHAT WE LEARNED - INTRO

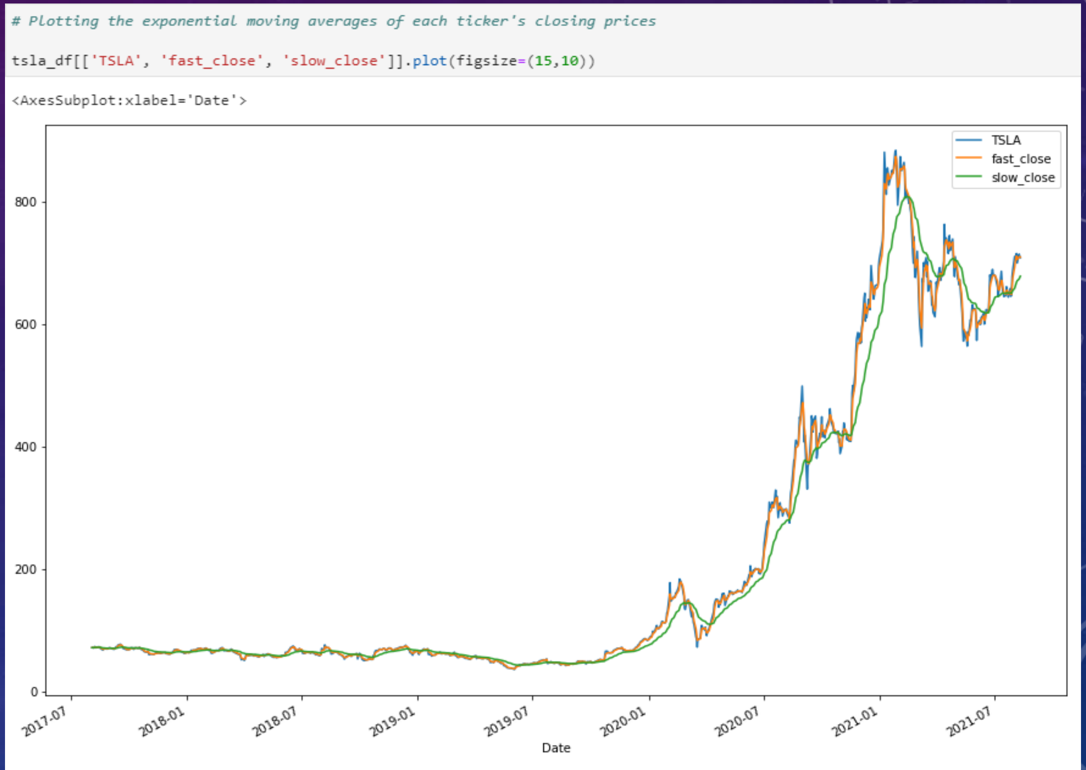
DATA SOURCES
- NewsAPI, the Guardian, MediaStack
- Wikipedia Views
- Y Finance

DATA CLEANUP + SIGNALS
- Assessing the value of the data sources
- Cleaning news and media data
- Bollinger Bands

TRAINING THE DATA + MODELS
- LSTM
- Random Forest Model

EVALUATING THE MODELS
- LSTM
- Random Forest Model

WHAT WE LEARNED - CLOSING



MOTIVATION AND SUMMARY

In this project, we initially hoped to leverage some of our members' niche knowledge to create a number of predictive trading models using NLP and sentiment analysis for one set of models, and LSTM and Random Forest models for the other, then compare them.

The goal was to determine first, if there were trends in the news/media data that could be used as predictors, and if those results were superior to other methods that relied on daily closing prices and % changes.

WHAT WE LEARNED

80% of the results, come from 20% of the effort.

– Pareto’s Law

The biggest lesson we learned is that 80% of our ideas seem to be bad ideas– but that’s OK. Pareto’s law applies here and shows us that 80% of our best results will come from 20% of our efforts. We’re just getting through the 80% of the useless efforts now. We also discovered two other laws during this project:

Anything that can go wrong, will go wrong.

– Murphy’s Law

Thinly sliced cabbage.

– Cole’s Law

DATA SOURCES 1: NEWSAPI

Since we had some familiarity with the NewsAPI service already, we turned to it first to see what we could get from it. The first limitation we found was that we could only go back 100 trading days– but we thought that would be good enough to see if there was some correlation between positive sentiment and closing price that we could use to educate our predictive models. This is where we hit our first major block.

NewsAPI, while capable of “going back” 100 days, would only serve us 20 articles to run sentiment analysis on, which wasn’t enough data to produce meaningful results (especially with companies like Ford or Tesla, which often had more than 20 articles published about them in a single given day).

We thought we could salvage some of our efforts here by setting up the logic with an easy “find and replace” structure that could serve “today’s sentiment” on any given ticker.

[18]:

	positive	negative	neutral
count	20.000000	20.000000	20.000000
mean	0.077250	0.029450	0.893250
std	0.069375	0.049338	0.067333
min	0.000000	0.000000	0.762000
25%	0.021750	0.000000	0.841250
50%	0.060000	0.000000	0.892500
75%	0.116000	0.051000	0.944000
max	0.197000	0.138000	1.000000

[19]:

```
## Converting the description to a df.

aapl_described_df = aapl_sentiment_df.describe()
aapl_described_df
```

[19]:

	positive	negative	neutral
count	20.000000	20.000000	20.000000
mean	0.077250	0.029450	0.893250
std	0.069375	0.049338	0.067333
min	0.000000	0.000000	0.762000
25%	0.021750	0.000000	0.841250
50%	0.060000	0.000000	0.892500
75%	0.116000	0.051000	0.944000
max	0.197000	0.138000	1.000000

[20]:

```
aapl_described_df['positive']['mean']
```

[20]: 0.07725000000000001

[21]:

```
## This is the logic for generating the "answers" to some of the questions.

aapl_pos = round(aapl_described_df['positive']['mean'],2)*100
aapl_neg = round(aapl_described_df['negative']['mean'],2)*100
aapl_neu = round(aapl_described_df['neutral']['mean'],2)*100
print(f"There were {aapl_total} articles written about AAPL in our range; of those, {aapl_pos}% of the articles were positive, {aapl_neg}% were negative,
```

There were 920 articles written about AAPL in our range; of those, 8.0% of the articles were positive, 3.0% were negative, and 89.0% were neutral.

DATA SOURCES 2: THE GUARDIAN

After realizing that the free NewsAPI results wouldn't be useful for our purposes, we searched for other APIs that would give us similar results, and one of these was the UK paper, The Guardian.

In addition to the usual registering for an API key and building that into the notebook that we'd done in class, we had to download and configure a .py file to be able to import The Guardian's results into our code.

You can find some of that documentation below:

- Guardian-API-Python (GitHub)
- The Guardian Open Platform Docs (Official)

In the end, despite the fact that we could "go back" as far as we wanted to, we were still limited by the number of results we could pull at any given time and decided to move on to other sources.

```python
"""
The content endpoint (/search) returns
all pieces of content in the API.
"""
import requests
import copy


class Content:

    def __init__(self, api, url=None, **kwargs):
        """
        :param api: api_key
        :param url: optional url to get the content.
        :param kwargs: optional header data
        :return: None
        """

        self.__headers = {
            "api-key": api,
            "format": "json"
        }
        self.__request_response = None

        if url is None:
            self.base_url = "https://content.guardianapis.com/search"
        else:
            self.base_url = url

        if kwargs:
            for key, value in kwargs.items():
                self.__headers[key] = value

    def __response(self, headers=None):

        """
        :param headers: optional header
        :return: returns raw response.
        """

        if headers is None:
            headers = self.__headers
        else:
            headers.update(self.__headers)

        res = requests.get(self.base_url, headers)

        return res

```

DATA SOURCES 3: MEDIASTACK

Our third attempt at sourcing usable natural language data, MediaStack seemed initially promising, allowing us to search articles from a number of data sources and filtering those by keyword, publication date, even down to the byline, which would– in theory– allow us to weigh the sentiment of certain industry "experts" over others.

The first hurdle we cam across was the way in which the data was presented– it looked like JSON, but it wasn't JSON, and required a fair bit of processing in order to make it usable.

The second was an issue of "pagination". Similar to NewsAPI and The Guardian, MediaStack limited a given result to the "first twenty" articles called. Again: an interesting exercise, but not enough information to be considered useful for our project.

MediaStack | Testing the API

[39]:
```
## From the MediaStack Documentation at https://mediastack.com/documentation

import http.client, urllib.parse

conn = http.client.HTTPConnection('api.mediastack.com')

params = urllib.parse.urlencode({
    'access_key': mediastack_api_key,
    'categories': 'business',
    'published_at': '2020-08-08',
    'keywords': 'BTC',
    'sort': 'published_desc',
    })

conn.request('GET', '/v1/news?{}'.format(params))

res = conn.getresponse()
btc_data = res.read()
```

[40]:
```
## print(ford_data)
```

[41]:
```
## data_decode_utf8 = data.decode('utf-8')
## print(data.decode(encoding='utf-8', errors='strict'))
```

[42]:
```
btc_data_json = btc_data.decode('utf8').replace("'", '"')
print(btc_data_json)
print('- ' * 20)
```


{"pagination":{"limit":25,"offset":0,"count":25,"total":351},"data":[{"author":"Cointelegraph","title":"Price analysis 8\/9: BTC, ETH, BNB, ADA, XRP, DOGE, DOT, UNI, BCH, LINK","description":"Price analysis 8\/9: BTC, ETH, BNB, ADA, XRP, DOGE, DOT, UNI, BCH, LINK","url":"https:\/\/www.investing.com\/news\/cryptocurrency-news\/price-analysis-89-btc-eth-bnb-ada-xrp-doge-dot-uni-bch-link-2586285","source":"Investing.com | Stock Market Quotes \u0026amp; Financial News","image":"https:\/\/d1-invdn-com.investing.com\/content\/pic787a818e48ffd062913b05c23fe8e5f7.jpg","category":"business","language":"en","country":"us","published_at":"2021-08-10T22:40:25+00:00"},{"author":"Cointelegraph","title":"No, Bitcoin isn\u2019t entering a 2018-like bear cycle, new data suggests, as BTC targets $45K","description":"No, Bitcoin isn\u2019t entering a 2018-like bear cycle, new data suggests, as BTC targets $45K","url":"https:\/\/www.investing.com\/news\/cryptocurrency-news\/no-bitcoin-isnt-entering-a-2018like-bear-cycle-new-data-suggests-as-btc-targets-45k-2585437","source":"Investing.com | Stock Market Quotes \u0026amp; Financial News","image":"https:\/\/d1-invdn-com.investing.com\/content\/pic694c8030513911372ff183256e6ea618.jpg","category":"business","language":"en","country":"us","published_at":"2021-08-10T12:20:24+00:00"},{"author":"Cointelegraph","title":"Large hodlers accumulate Bitcoin below $50K as BTC transactions over $1M soar","description":"Large hodlers accumulate Bitcoin below $50K as BTC transactions over $1M soar","url":"https:\/\/www.investing.com\/news\/cryptocurrency-news\/large-hodlers-accumulate-bitcoin-below-50k-as-btc-transactions-over-1m-soar-2585409","source":"Investing.com | Stock Market Quotes \u0026amp; Financial News","image":"https:\/\/d1-invdn-com.investing.com\/content\/picc6036077674e4b1d7e7aefc706a8337d.jpg","category":"business","language":"en","country":"us","published_at":"2021-08-10T12:00:16+00:00"},{"author":"Cointelegraph","title":"Bitcoin \u2018golden cross\u2019 due in days as bears draw a line at $47K BTC price","description":"Bitcoin \u2018golden cross\u2019 due in days as bears draw a line at $47K BTC price","url":"https:\/\/www.investing.com\/news\/cryptocurrency-news\/bitcoin-golden-cross-due-in-days-as-bears-draw-a-line-at-47k-btc-price-2585305","source":"Investing.com | Stock Market Quotes \u0026amp; Financial News","image":"https:\/\/d1-invdn-com.investing.com\/content\/piccb2b2f23677325741531920e8197d416.jpg","category":"business","language":"en","country":"us","published_at":"2021-08-10T10:40:16+00:00"},{"author":"Cointelegraph","title":"Cinema operator AMC plans to accept BTC by 2022","description":"Cinema operator AMC plans to accept BTC by 2022","url":"https:\/\/www.investing.com\/news\/cryptocurrency-news\/cinema-operator-amc-plans-to-accept-btc-by-2022-2585036","source":"Investing.com | Stock Market Quotes \u0026amp; Financial News","image":"https:\/\/d1-invdn-com.investing.com\/content\/picf0cbb5e50d070736aab4aa79ef00972b.jpg","category":"business","language":"en","country":"us","published_at":"2021-08-10T03:40:16+00:00"},{"author":"Cointelegraph","title":"Top 5 cryptocurrencies to watch this week: BTC, LTC, ICP, THETA, FTT","description":"Top 5 cryptocurrencies to watch this week: BTC, LTC, ICP, THETA, FTT","url":"https:\/\/www.investing.com\/news\/cryptocu

DATA SOURCES 4: WIKIPEDIA

After three attempts at finding article data that was useful, we pivoted and decided to track pageviews on WikiPedia. We believed this would be similar to Google Search trends, with greater interest in learning about a given company translating to positive movement in that stock's closing price.

The data was readily available and we were able to successfully integrate it into our code on different models and notebooks.

Run notebook

```
#Storing article(ticker) name, timestamp, and view counts into df
Companies=['Tesla,_Inc.','Ford_Motor_Company','General_Motors', 'CarMax']
pageviewdf_len=[]
for i in Companies:
    page_views=pageviewapi.per_article('en.wikipedia', i, '20170821','20210811',
                                       access='all-access', agent='all-agents', granularity='daily')
    if
```

Execution has been cancelled.

```
SyntaxError: invalid syntax (4277848722.py, line 7)
```

Show error details

```
#Creating wikipv_df to store wiki pageview
var  = [i for i in range(len(page_views['items']))]
wikipv_df = pd.DataFrame(index=var)
wikipv_df['article'] = np.nan
wikipv_df['views'] = np.nan
wikipv_df['time_stamp'] = ''
```

Execution has been cancelled.

```
#Storing article(ticker) name, timestamp, and view counts into df
Companies=['Tesla,_Inc.','Ford_Motor_Company','General_Motors', 'CarMax']
for i in Companies:
    page_views=pageviewapi.per_article('en.wikipedia', i, '20170821','20210811',
                                       access='all-access', agent='all-agents', granularity='daily')
    #print(page_views)
    for i in range(len(page_views['items'])):
        wikipv_df['article'][i] = page_views['items'][i]['article']
        wikipv_df['views'][i] = page_views['items'][i]['views']
        wikipv_df['time_stamp'][i] = page_views['items'][i]['timestamp'][0:8]
```

Execution has been cancelled.

```
#reformat timestamp
wikipv_df['time_stamp'] = pd.to_datetime(df_save['time_stamp'], format='%Y%m%d')
```

Execution has been cancelled.

DATA SOURCES 5: Y FINANCE

We used the y_finance library to pull 500d and 1000d of trading data on the S&P 500 stocks, then converted it into a .csv file so we could more easily manipulate the stock data.

Run notebook

```
period_days = "500d"
symbol1 = tickers[1]
rawSymbol1_ticker = yf.Ticker(symbol1)
rawSymbol1 = rawSymbol1_ticker.history(period=period_days)
rawSymbol1.head()
```

	Open	High	Low	Close	Volume	Dividends	Stock Splits
Date							
2019-08-16	1792.890015	1802.910034	1784.550049	1792.569946	3018000	0	0
2019-08-19	1818.079956	1826.000000	1812.609985	1816.119995	2816300	0	0
2019-08-20	1814.500000	1816.819946	1799.880005	1801.380005	1929500	0	0
2019-08-21	1819.390015	1829.579956	1815.000000	1823.540039	2031800	0	0
2019-08-22	1828.000000	1829.410034	1800.099976	1804.660034	2653500	0	0

```
day_count = "1000d"
raw_all_symbols = yf.Tickers(tickers)
raw_all_symbols = raw_all_symbols.history(period=day_count)
raw_all_symbols.head()
```

```
[*********************100%***********************]  505 of 505 completed

2 Failed downloads:
- BRK.B: No data found, symbol may be delisted
- BF.B: No data found for this date range, symbol may be delisted
```

	Adj Close		Close								...	Volume		
	BF.B	BRK.B	A	AAL	AAP	AAPL	ABBV	ABC	ABMD	ABT	...	XEL	XLNX	XOM
Date														
2017-08-21	NaN	NaN	59.741249	44.454647	93.436043	37.509666	58.532715	73.017456	153.320007	45.729797	...	1903300	2130600	100849
2017-08-22	NaN	NaN	60.921734	44.162891	94.034775	38.122852	59.402012	74.130867	154.990005	46.506310	...	2257400	1587500	950360

DATA CLEANUP AND SIGNALS

1 ASSESSING THE VALUE OF THE DATA SOURCES

We already covered some of the shortcomings of the news and natural language APIs, which– while seemingly valuable– were so limited in scope in their “free” versions as to be mostly unusable. Still, reading in and cleaning the data was valuable repetition of skills, and is worth sharing in this section.

The S&P500 closing price history was available as a .csv file. That proved valuable and useful for our final models, as it gave us enough history to start to pull signal from noise in our predictive models.

DATA CLEANUP AND SIGNALS

2 CLEANING THE NEWSAPI DATA (1:2)

For the NewsAPI section, we pulled in the data using the same methods we used during our crypto sentiment homework.

In the image at right, you'll see us pull the total number of articles available in our "100 days". This was a later addition, after we had pivoted from a complex sentiment data to a more simple, **volume** data approach.

```
[9]:  # Read your api key environment variable

      newsapi_key = os.getenv("NEWSAPI_ORG_KEY")

[10]: newsapi = NewsApiClient(api_key=newsapi_key)
      newsapi

[10]: <newsapi.newsapi_client.NewsApiClient at 0x23efa66f7f0>

[11]: # Fetch the relevant news articles
      ## Study group says we're sorting by "relevancy"; sounds right.
      ## Naming this one tezos_articles, because I'm using for articles about Tezos.

      tsla_articles = newsapi.get_everything(
          q="tsla", language="en", sort_by="relevancy"
          )

[12]: ## Let's see if that worked.
      ## Show some sample articles.

      tsla_articles["articles"][:1]

[12]: [{'source': {'id': None, 'name': 'MarketBeat'},
        'author': 'Sam Quirke',
        'title': 'Where Does Tesla (NASDAQ: TSLA) Go From Here?',
        'description': 'Tesla's (NASDAQ: TSLA) Q2 earnings were released after the bell rang to end yesterday's sessio
        'url': 'https://www.marketbeat.com/originals/where-does-tesla-nasdaq-tsla-go-from-here/?utm_source=entrepreneu
        'urlToImage': 'https://assets.entrepreneur.com/providers/marketbeat/hero-image-marketbeat-378510.jpeg',
        'publishedAt': '2021-07-27T11:00:00Z',
        'content': 'This story originally appeared on MarketBeatTesla's (NASDAQ: TSLA) Q2 earnings were released after
      eet alike a much-awaited … [+4129 chars]'}]

[13]: ## Let's make a variable to count the results.

      tsla_total = tsla_articles["totalResults"]
      print(tsla_total)

      746

[28]: ## Now I'll print an f-string (for later).
      print(f"There were {tsla_total} articles written about tsla in our News API range.")

      There were 746 articles written about tsla in our News API range.
```

DATA CLEANUP AND SIGNALS

2 CLEANING THE NEWSAPI DATA (2:2)

At right, you can see how we used the same sort of polarity_scores analyzer from the homework, looking at the “standard” positive, negative, and neutral. As you read the code, it’s important to note two things in this figure:

1. There is no “niche knowledge” here, as we had already established we would have insufficient data to draw relevant conclusions.
2. The code is written in such a way that a “Replace All” command in the notebook would effectively generate a recent sentiment score for any stock ticker.

AAPL
TSLA Replace

[16]:
```python
# Creating the ticker sentiment scores DataFrame
## This came with a *TON* of help from the study group and copy/pasting from class work.

tsla_sentiment = []

for article in tsla_articles["articles"]:
    try:
        text = article["content"]
        sentiment = analyzer.polarity_scores(text)
        pos = sentiment["pos"]
        neu = sentiment["neu"]
        neg = sentiment["neg"]

        tsla_sentiment.append({
            "positive": pos,
            "negative": neg,
            "neutral": neu,
            "text": text
        })

    except AttributeError:
        pass
```

[17]:
```python
tsla_sentiment_df = pd.DataFrame(tsla_sentiment)
tsla_sentiment_df.head()
```

[17]:

	positive	negative	neutral	text
0	0.000	0.0	1.000	This story originally appeared on MarketBeatTe...
1	0.000	0.0	1.000	A version of this story first appeared in CNN ...
2	0.000	0.0	1.000	Tesla's second quarter earnings revealed a rec...
3	0.164	0.0	0.836	August\r\n10, 2021\r\n6 min read\r\nThis story...
4	0.182	0.0	0.818	This story originally appeared on MarketBeat\r...

[18]:
```python
## Describe the ticker Sentiment

tsla_sentiment_df.describe()
```

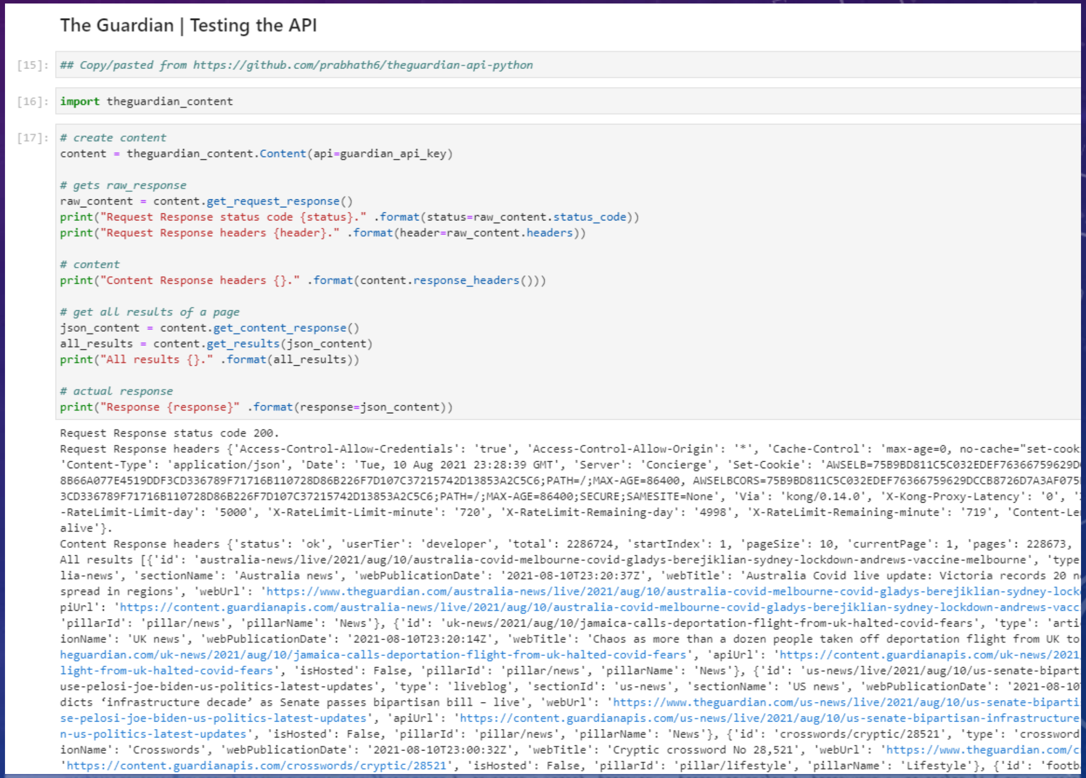
[18]:

	positive	negative	neutral
count	20.000000	20.000000	20.000000
mean	0.062500	0.020600	0.916900
std	0.068416	0.045981	0.072209

DATA CLEANUP AND SIGNALS

2 CLEANING THE GUARDIAN DATA (1:2)

After downloading the necessary .py files and importing theguardian_content library, we followed the instructions in the documentation and got our responses back in a JSON format.



The Guardian | Testing the API

[15]:
```
## Copy/pasted from https://github.com/prabhath6/theguardian-api-python
```

[16]:
```
import theguardian_content
```

[17]:
```
# create content
content = theguardian_content.Content(api=guardian_api_key)

# gets raw_response
raw_content = content.get_request_response()
print("Request Response status code {status}." .format(status=raw_content.status_code))
print("Request Response headers {header}." .format(header=raw_content.headers))

# content
print("Content Response headers {}." .format(content.response_headers()))

# get all results of a page
json_content = content.get_content_response()
all_results = content.get_results(json_content)
print("All results {}." .format(all_results))

# actual response
print("Response {response}" .format(response=json_content))
```


Request Response status code 200.
Request Response headers {'Access-Control-Allow-Credentials': 'true', 'Access-Control-Allow-Origin': '*', 'Cache-Control': 'max-age=0, no-cache="set-cook
'Content-Type': 'application/json', 'Date': 'Tue, 10 Aug 2021 23:28:39 GMT', 'Server': 'Concierge', 'Set-Cookie': 'AWSELB=75B9BD811C5C032EDEF76366759629D
8B66A077E4519DDF3CD336789F71716B110728D86B226F7D107C37215742D13853A2C5C6;PATH=/;MAX-AGE=86400, AWSELBCORS=75B9BD811C5C032EDEF76366759629DCCB8726D7A3AF075
3CD336789F71716B110728D86B226F7D107C37215742D13853A2C5C6;PATH=/;MAX-AGE=86400;SECURE;SAMESITE=None', 'Via': 'kong/0.14.0', 'X-Kong-Proxy-Latency': '0', '
-RateLimit-Limit-day': '5000', 'X-RateLimit-Limit-minute': '720', 'X-RateLimit-Remaining-day': '4998', 'X-RateLimit-Remaining-minute': '719', 'Content-Le
alive'}.
Content Response headers {'status': 'ok', 'userTier': 'developer', 'total': 2286724, 'startIndex': 1, 'pageSize': 10, 'currentPage': 1, 'pages': 228673,
All results [{'id': 'australia-news/live/2021/aug/10/australia-covid-melbourne-covid-gladys-berejiklian-sydney-lockdown-andrews-vaccine-melbourne', 'type
lia-news', 'sectionName': 'Australia news', 'webPublicationDate': '2021-08-10T23:20:37Z', 'webTitle': 'Australia Covid live update: Victoria records 20 n
spread in regions', 'webUrl': 'https://www.theguardian.com/australia-news/live/2021/aug/10/australia-covid-melbourne-covid-gladys-berejiklian-sydney-lock
piUrl': 'https://content.guardianapis.com/australia-news/live/2021/aug/10/australia-covid-melbourne-covid-gladys-berejiklian-sydney-lockdown-andrews-vacc
'pillarId': 'pillar/news', 'pillarName': 'News'}, {'id': 'uk-news/2021/aug/10/jamaica-calls-deportation-flight-from-uk-halted-covid-fears', 'type': 'arti
ionName': 'UK news', 'webPublicationDate': '2021-08-10T23:20:14Z', 'webTitle': 'Chaos as more than a dozen people taken off deportation flight from UK to
heguardian.com/uk-news/2021/aug/10/jamaica-calls-deportation-flight-from-uk-halted-covid-fears', 'apiUrl': 'https://content.guardianapis.com/uk-news/2021
light-from-uk-halted-covid-fears', 'isHosted': False, 'pillarId': 'pillar/news', 'pillarName': 'News'}, {'id': 'us-news/live/2021/aug/10/us-senate-bipart
use-pelosi-joe-biden-us-politics-latest-updates', 'type': 'liveblog', 'sectionId': 'us-news', 'sectionName': 'US news', 'webPublicationDate': '2021-08-10
dicts ‘infrastructure decade’ as Senate passes bipartisan bill – live', 'webUrl': 'https://www.theguardian.com/us-news/live/2021/aug/10/us-senate-biparti
se-pelosi-joe-biden-us-politics-latest-updates', 'apiUrl': 'https://content.guardianapis.com/us-news/live/2021/aug/10/us-senate-bipartisan-infrastructure
n-us-politics-latest-updates', 'isHosted': False, 'pillarId': 'pillar/news', 'pillarName': 'News'}, {'id': 'crosswords/cryptic/28521', 'type': 'crossword
ionName': 'Crosswords', 'webPublicationDate': '2021-08-10T23:00:32Z', 'webTitle': 'Cryptic crossword No 28,521', 'webUrl': 'https://www.theguardian.com/c
'https://content.guardianapis.com/crosswords/cryptic/28521', 'isHosted': False, 'pillarId': 'pillar/lifestyle', 'pillarName': 'Lifestyle'}, {'id': 'footb

DATA CLEANUP AND SIGNALS

2 CLEANING THE GUARDIAN DATA (2:2)

We eventually got the data to pretty print using the dumps() method.

Because of the pagination issues, and the difficulty in ensuring that the environment could be duplicated by other users, we abandoned The Guardian API here.


```
[59]: [{'author': 'Cointelegraph',
  'title': 'Price analysis 8/9: BTC, ETH, BNB, ADA, XRP, DOGE, DOT, UNI, BCH, LINK',
  'description': 'Price analysis 8/9: BTC, ETH, BNB, ADA, XRP, DOGE, DOT, UNI, BCH, LINK',
  'url': 'https://www.investing.com/news/cryptocurrency-news/price-analysis-89-btc-eth-bnb-ada-xrp-doge-dot-uni-bch-link-2586285',
  'source': 'Investing.com | Stock Market Quotes &amp; Financial News',
  'image': 'https://d1-invdn-com.investing.com/content/pic787a818e48ffd062913b05c23fe8e5f7.jpg',
  'category': 'business',
  'language': 'en',
  'country': 'us',
  'published_at': '2021-08-10T22:40:25+00:00'},
 {'author': 'Cointelegraph',
  'title': 'No, Bitcoin isn’t entering a 2018-like bear cycle, new data suggests, as BTC targets $45K',
  'description': 'No, Bitcoin isn’t entering a 2018-like bear cycle, new data suggests, as BTC targets $45K',
  'url': 'https://www.investing.com/news/cryptocurrency-news/no-bitcoin-isnt-entering-a-2018like-bear-cycle-new-data-suggests-as-btc-targets-45k-2585437',
  'source': 'Investing.com | Stock Market Quotes &amp; Financial News',
  'image': 'https://d1-invdn-com.investing.com/content/pic694c8030513911372ff183256e6ea618.jpg',
  'category': 'business',
  'language': 'en',
  'country': 'us',
  'published_at': '2021-08-10T12:20:24+00:00'},
 {'author': 'Cointelegraph',
  'title': 'Large hodlers accumulate Bitcoin below $50K as BTC transactions over $1M soar',
  'description': 'Large hodlers accumulate Bitcoin below $50K as BTC transactions over $1M soar',
  'url': 'https://www.investing.com/news/cryptocurrency-news/large-hodlers-accumulate-bitcoin-below-50k-as-btc-transactions-over-1m-soar-2585409',
  'source': 'Investing.com | Stock Market Quotes &amp; Financial News',
  'image': 'https://d1-invdn-com.investing.com/content/picc6036077674e4b1d7e7aefc706a8337d.jpg',
  'category': 'business',
  'language': 'en',
  'country': 'us',
  'published_at': '2021-08-10T12:00:16+00:00'},
 {'author': 'Cointelegraph',
  'title': 'Bitcoin ‘golden cross’ due in days as bears draw a line at $47K BTC price',
  'description': 'Bitcoin ‘golden cross’ due in days as bears draw a line at $47K BTC price',
  'url': 'https://www.investing.com/news/cryptocurrency-news/bitcoin-golden-cross-due-in-days-as-bears-draw-a-line-at-47k-btc-price-2585305',
  'source': 'Investing.com | Stock Market Quotes &amp; Financial News',
  'image': 'https://d1-invdn-com.investing.com/content/piccb2b2f23677325741531920e8197d416.jpg',
  'category': 'business',
  'language': 'en',
  'country': 'us',
  'published_at': '2021-08-10T10:40:16+00:00'},
 {'author': 'Cointelegraph',
  'title': 'Cinema operator AMC plans to accept BTC by 2022',
  'description': 'Cinema operator AMC plans to accept BTC by 2022',
  'url': 'https://www.investing.com/news/cryptocurrency-news/cinema-operator-amc-plans-to-accept-btc-by-2022-2585036',
  'source': 'Investing.com | Stock Market Quotes &amp; Financial News',
  'image': 'https://d1-invdn-com.investing.com/content/picf0cbb5e50d070736aab4aa79ef00972b.jpg',
  'category': 'business',
  'language': 'en',
```

DATA CLEANUP AND SIGNALS

2 CLEANING THE MEDIASTACK DATA

The MediaStack documentation was incredibly thorough, and we were able to import the http.client, set the params, and get a response very easily.

The biggest challenge was that MediaStack's results returned with apostrophes (‘) instead of quotation marks (“) which had to be replaced in the string in order to be recognized as JSON.

```
[39]: ## From the MediaStack Documentation at https://mediastack.com/documentation

      import http.client, urllib.parse

      conn = http.client.HTTPConnection('api.mediastack.com')

      params = urllib.parse.urlencode({
          'access_key': mediastack_api_key,
          'categories': 'business',
          'published_at': '2020-08-08',
          'keywords': 'BTC',
          'sort': 'published_desc',
          })

      conn.request('GET', '/v1/news?{}'.format(params))

      res = conn.getresponse()
      btc_data = res.read()

[40]: ## print(ford_data)

[41]: ## data_decode_utf8 = data.decode('utf-8')
      ## print(data.decode(encoding='utf-8', errors='strict'))

[42]: btc_data_json = btc_data.decode('utf8').replace("'", '"')
      print(btc_data_json)
      print('- ' * 20)
```

DATA CLEANUP AND SIGNALS

2 CLEANING THE WIKIPEDIA DATA

The Wikipedia views data was pretty straightforward, we created a df to store the pageviews with the granularity of the query set to “daily” and indexed by date.

From there we binned the views count and applied the standard scalar to use it as an indicator in our later predictive models.

```
import pageviewapi
#pip install git+https://github.com/Commonists/pageview-api.git
import datetime
```

```
#Storing article(ticker) name, timestamp, and view counts into df
Companies=['Tesla,_Inc.','Ford_Motor_Company','General_Motors', 'CarMax']
#Companies=['Tesla,_Inc.']
#Companies=['Ford_Motor_Company']
#Companies=['General_Motors']
pageviewdf_len=[]
for i in Companies:
    page_views=pageviewapi.per_article('en.wikipedia', i, '20170803','20210811',
                        access='all-access', agent='all-agents', granularity='daily')
```

```
#Creating wikipv_df to store wiki pageview
var  = [i for i in range(len(page_views['items']))]
wikipv_df = pd.DataFrame(index=var)
Companies=['Tesla,_Inc.','Ford_Motor_Company','General_Motors', 'CarMax']
for i in Companies:
    wikipv_df[i+'views'] = np.nan
    wikipv_df[i+'time_stamp'] = ''
```

```
page_views=pageviewapi.per_article('en.wikipedia', 'Ford_Motor_Company', '20170803','20210811',
                        access='all-access', agent='all-agents', granularity='daily')
page_views['items'][1]
```

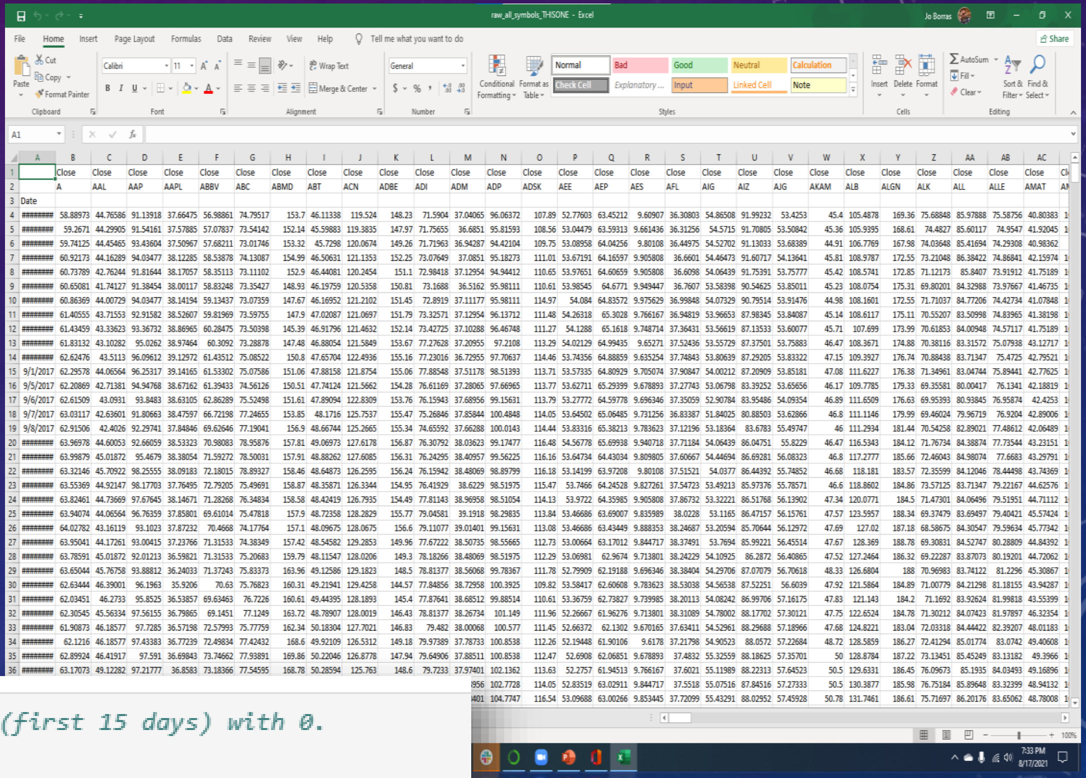
```
{'project': 'en.wikipedia',
 'article': 'Ford_Motor_Company',
 'granularity': 'daily',
 'timestamp': '2017080400',
 'access': 'all-access',
 'agent': 'all-agents',
 'views': 4304}
```

DATA CLEANUP AND SIGNALS

2 CLEANING THE S&P 500 DATA

We converted the y finance data to a .csv file (shown here, at right).

Early in the project we decided to focus on automotive stocks (specifically F, GM, KMX, and TSLA) so cleaning this data was a matter of dropping columns and replacing NaN values for TSLA (the first 15 trading days in our data) with a “0” value.



```
[7]: ## Filling the NaN values (first 15 days) with 0.

tsla_df = tsla_df.fillna(0)
tsla_df
```

DATA CLEANUP AND SIGNALS

LOOKING FOR SIGNALS IN THE DATA (1:3)

We then isolated each ticker and created a pct_chg column using the daily close data. We decided that a positive swing of more than 2% should return a 1 signal, while a negative swing greater than 2% should return a -1 signal. 0 signals signified a hold, or “no signal”.

NOTE: the logic is written such that you could find and replace the ticker symbol and generate the same type of results quickly.

```
[8]: ## I need to get a pct change from the TSLA closing prices so I can get a + or - on the day.
     ## We need a margin of error ("x%" = 0 to compensate for flat days or $0.01 or$0.02, for example); isolates large
     ## I need to convert the + or - on the day to a binary input (trade on +1 or -1, no action on 0).
     ## I need to make the binary data the y in my features set.
```

```
[9]: ## Getting % daily change from daily close column (named as ticker).

     pct_chg = tsla_df['TSLA'].pct_change()
     pct_chg
```

```
[9]: Date
     2017-08-04         NaN
     2017-08-07   -0.004875
     2017-08-08    0.028296
     2017-08-09   -0.004627
     2017-08-10   -0.022364
                     ...
     2021-08-05    0.005219
     2021-08-06   -0.021732
     2021-08-09    0.020970
     2021-08-10   -0.005282
```

```
pulat the "Daily_Change" column in my DF.
```

```
[13]: ## I am going to construct a trading signal based on +/- 2% daily change.
      ## Construct a trading signal using the Daily Change value.
      ## We want a signal when the stock is going up, going down, or trading sideways. up/down/hold.

      tsla_df['up_signal'] = np.where(tsla_df['Daily Change'] > 0.02, 1.0, 0.0)
      tsla_df['down_signal'] = np.where(tsla_df['Daily Change'] < -0.02, -1.0, 0.0)
```

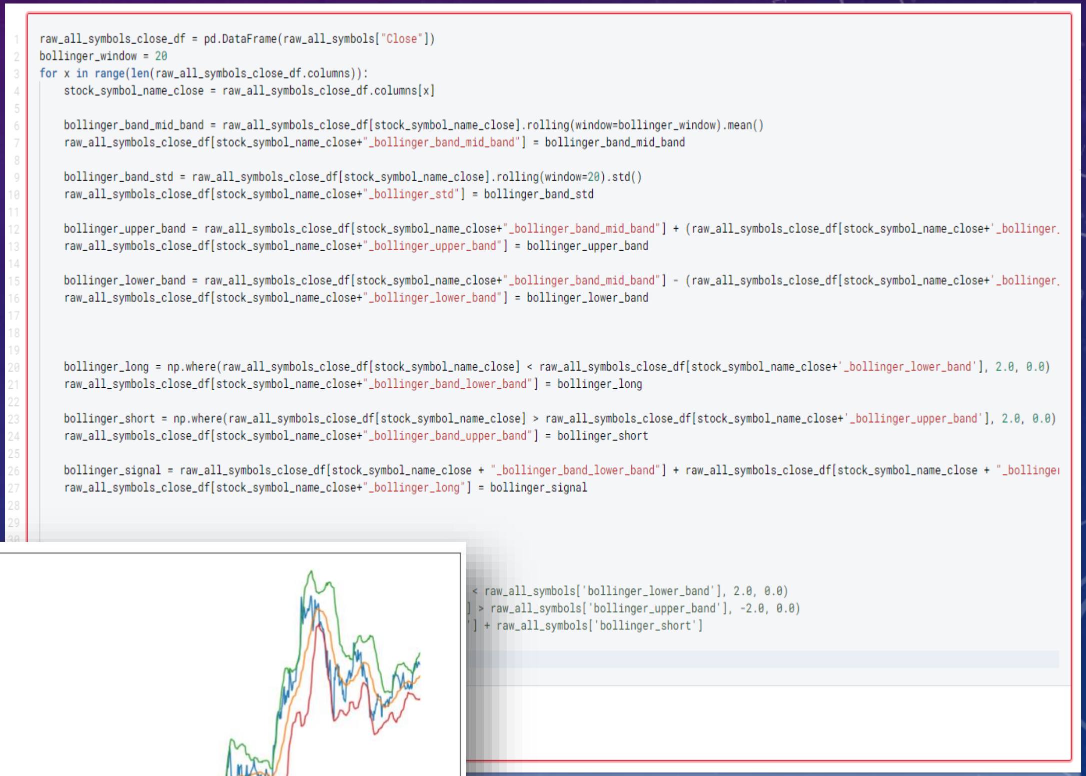
```
[14]: ## I want to COMBINE the up and down signals into 1 binary value.
      tsla_return = tsla_df['up_signal'] + tsla_df['down_signal']
      tsla_return
```

DATA CLEANUP AND SIGNALS

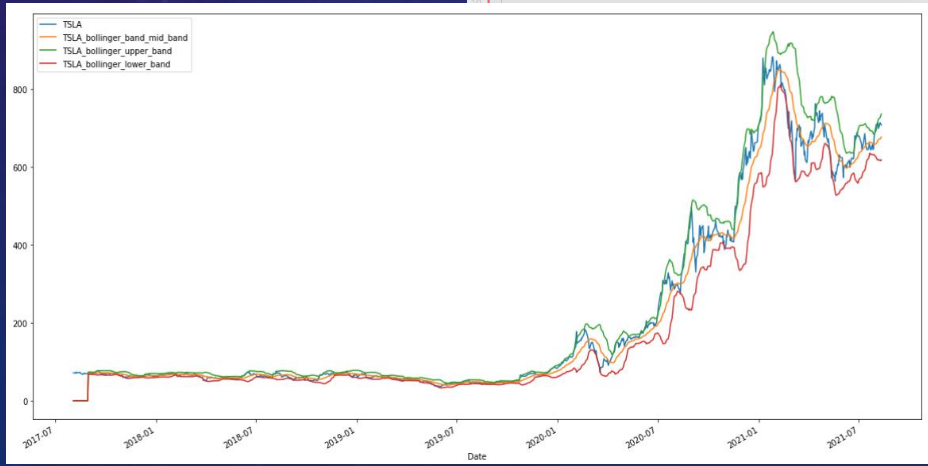
LOOKING FOR SIGNALS IN THE DATA (2:3)

For our next set of signals, we used Bollinger Bands representing two standard deviations above the 20 day moving average and two standard deviations below the 20 day moving average.

Our model created a signal whenever the actual closing price pierced the upper and lower Bollinger Bands our model signals a sell or buy signal depending on which band was pierced.



```
raw_all_symbols_close_df = pd.DataFrame(raw_all_symbols["Close"])
bollinger_window = 20
for x in range(len(raw_all_symbols_close_df.columns)):
    stock_symbol_name_close = raw_all_symbols_close_df.columns[x]

    bollinger_band_mid_band = raw_all_symbols_close_df[stock_symbol_name_close].rolling(window=bollinger_window).mean()
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_band_mid_band"] = bollinger_band_mid_band

    bollinger_band_std = raw_all_symbols_close_df[stock_symbol_name_close].rolling(window=20).std()
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_std"] = bollinger_band_std

    bollinger_upper_band = raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_band_mid_band"] + (raw_all_symbols_close_df[stock_symbol_name_close+'_bollinger.
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_upper_band"] = bollinger_upper_band

    bollinger_lower_band = raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_band_mid_band"] - (raw_all_symbols_close_df[stock_symbol_name_close+'_bollinger.
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_lower_band"] = bollinger_lower_band



    bollinger_long = np.where(raw_all_symbols_close_df[stock_symbol_name_close] < raw_all_symbols_close_df[stock_symbol_name_close+'_bollinger_lower_band'], 2.0, 0.0)
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_band_lower_band"] = bollinger_long

    bollinger_short = np.where(raw_all_symbols_close_df[stock_symbol_name_close] > raw_all_symbols_close_df[stock_symbol_name_close+'_bollinger_upper_band'], 2.0, 0.0)
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_band_upper_band"] = bollinger_short

    bollinger_signal = raw_all_symbols_close_df[stock_symbol_name_close + "_bollinger_band_lower_band"] + raw_all_symbols_close_df[stock_symbol_name_close + "_bollinge
    raw_all_symbols_close_df[stock_symbol_name_close+"_bollinger_long"] = bollinger_signal

< raw_all_symbols['bollinger_lower_band'], 2.0, 0.0)
] > raw_all_symbols['bollinger_upper_band'], -2.0, 0.0)
'] + raw_all_symbols['bollinger_short']
```



TRAINING THE DATA AND MODELS

LSTM MODEL (1:3)

We built the LSTM model out with each of the automotive stock tickers using the same “rinse and repeat” code blocks we went over in class.

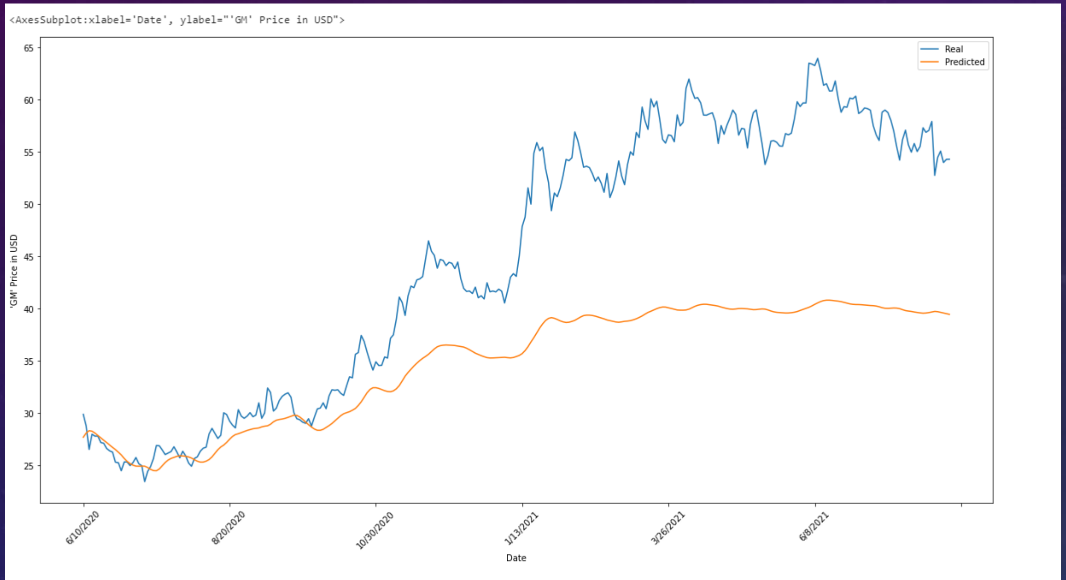
```
[16]: ## Importing tensorflow.

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
```

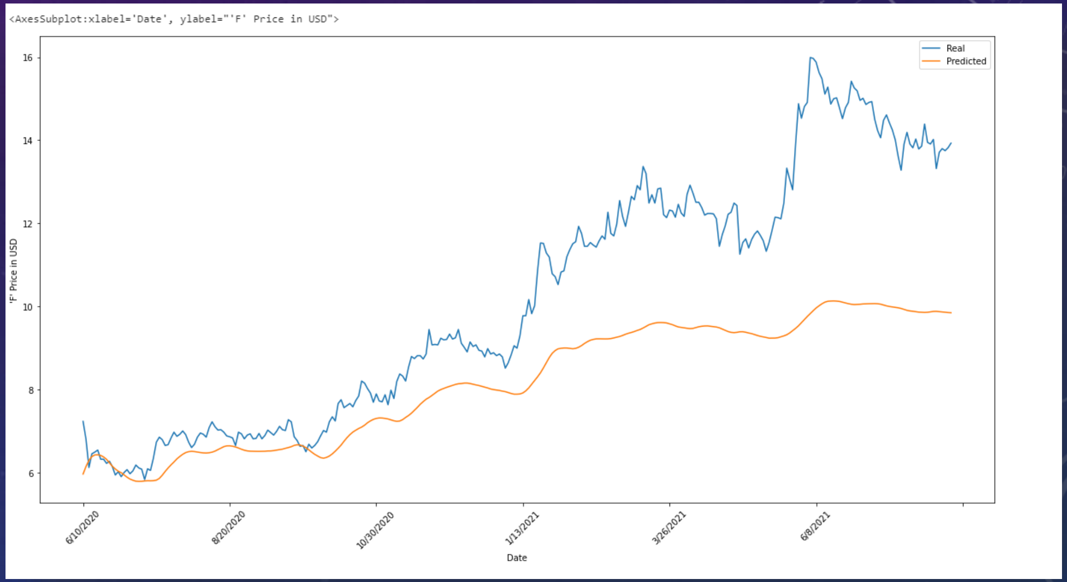
```
[17]: # Build the LSTM model.
# The return sequences need to be set to True if you are adding additional LSTM layers, but
# You don't have to do this for the final layer.
# Note: The dropouts help prevent overfitting
# Note: The input shape is the number of time steps and the number of indicators
# Note: Batching inputs has a different input shape of Samples/TimeSteps/Features
## LSTM = Long, Short-term Memory

model = Sequential()

number_units = 30
dropout_fraction = 0.2

# Layer 1
model.add(LSTM(
    units=number_units,
    return_sequences=True,
    ## playing with [x], 1 for the shape.
    input_shape=(X_train.shape[-1], 1))
    )
model.add(Dropout(dropout_fraction))

# Layer 2
model.add(LSTM(units=number_units, return_sequences=True))
model.add(Dropout(dropout_fraction))

# Layer 3

model.add(LSTM(units=number_units))
model.add(Dropout(dropout_fraction))

# Output layer

model.add(Dense(1))
```

```
[18]: # Compile the model
## Copied from the student-dos and classwork.
## Who is adam?

model.compile(optimizer="adam", loss="mean_squared_error")
```

TRAINING THE DATA AND MODELS

LSTM MODEL (2:3)

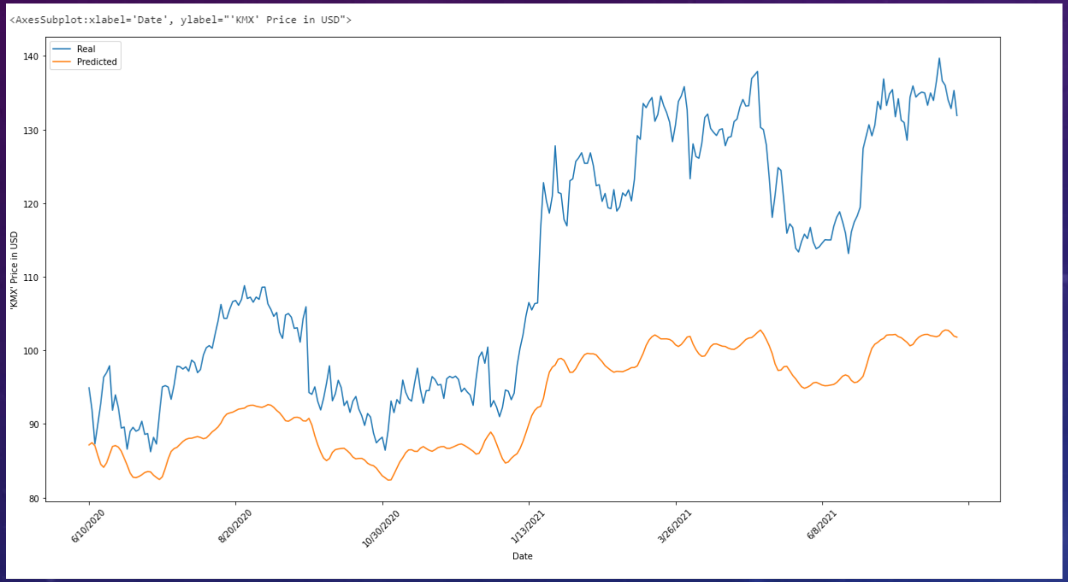


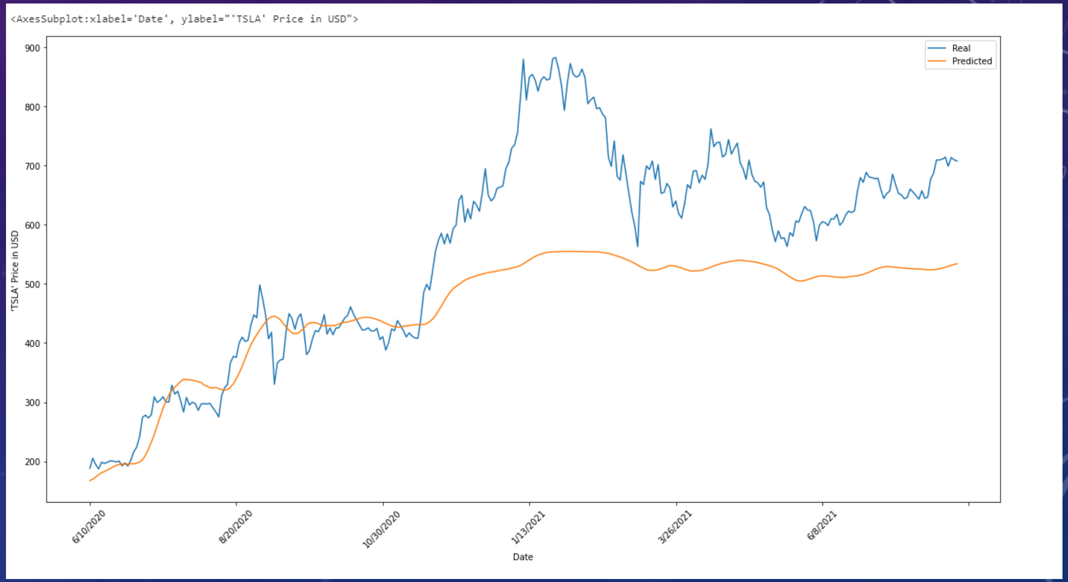


The LSTM models "hugged" the real prices fairly well in the first six months before trailing behind– but they still followed the overall trend of the closing prices. Here you can see the GM and F stocks.

TRAINING THE DATA AND MODELS

LSTM MODEL (2:3)





The TSLA model was interesting, as changing the number of epochs and batch sizes run change the results dramatically. Again, it was very good for the first six months, but went “flat” during TSLA’s bull run between OCT20 and FEB21.

TRAINING THE DATA AND MODELS

RANDOM FOREST (1:8)

For the random forest model, we started with the whole of the S&P and with a regression strategy that used Bollinger bands and RSI to find over extended markets. We narrowed our focus to automotive stocks within the S&P with plans to add NLTK indicators later.

```
[10]: ## Next, I need to get the data in pct_change to populat the "Daily_Change" column in my DF.
```

```
[11]: tsla_df["Daily Change"] = pct_chg
      tsla_df
```

[11]:

Date	Tesla,_Inc.views	TSLA_RSI	TSLA	TSLA_bollinger_band_mid_band	TSLA_bollinger_std	TSLA_bollinger_upper_band	TSLA_bollinger_lower_band	Daily Change
2017-08-04	9210.0	0.000000	71.382004	0.000000	0.000000	0.000000	0.000000	NaN
2017-08-07	8892.0	0.000000	71.033997	0.000000	0.000000	0.000000	0.000000	-0.004875
2017-08-08	8558.0	0.000000	73.043999	0.000000	0.000000	0.000000	0.000000	0.028296
2017-08-09	7669.0	0.000000	72.706001	0.000000	0.000000	0.000000	0.000000	-0.004627
2017-08-10	7189.0	0.000000	71.080002	0.000000	0.000000	0.000000	0.000000	-0.022364
...	...	...	...	...	...	...	...	...
2021-08-05	8790.0	67.541974	714.630005	668.646494	25.338475	719.323443	617.969544	0.005219
2021-08-06	8811.0	60.301009	699.099976	670.753992	26.057127	722.868246	618.639738	-0.021732
2021-08-09	9394.0	64.202437	713.760010	672.156992	27.613183	727.383357	616.930626	0.020970
2021-08-10	8950.0	62.501349	709.989990	674.229492	28.855007	731.939506	616.519478	-0.005282
2021-08-11	9051.0	61.491428	707.820007	676.951492	29.348230	735.647952	618.255033	-0.003056

1012 rows × 8 columns

```
[12]: ## Now, I want to make a signal using the % change; higher than 0.02
      print(tsla_df['Daily Change'])

Date
2017-08-04         NaN
2017-08-07   -0.004875
2017-08-08    0.028296
2017-08-09   -0.004627
2017-08-10   -0.022364
```

TRAINING THE DATA AND MODELS

RANDOM FOREST (2:8)

We used a 10 day and 20 day rolling average and converted the different data to binary for the up and down signals in our model. The resulting model was a bit slower than we would have liked, missing the “peak” of the late 2020-Feb. 2021 bull runs by about 10%– but still finishing “ahead”.

You can see the Ford (F) model, at right.

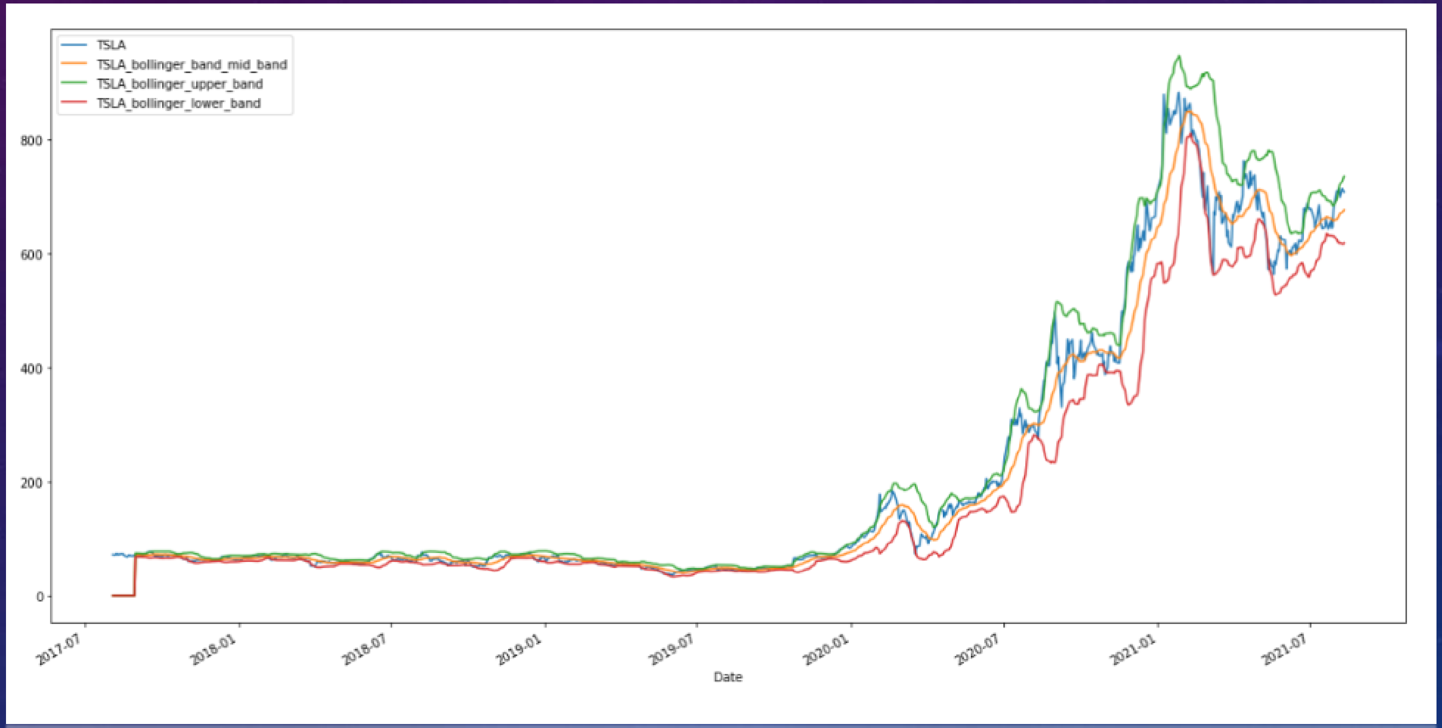
TRAINING THE DATA AND MODELS

RANDOM FOREST (3:8)

We saw a similar “delay” using this 10/20 day crossover method across all the stocks we used. You can see the Tesla (TSLA) model, at right.

TRAINING THE DATA AND MODELS

RANDOM FOREST (4:8)



Here you can see the Bollinger upper, lower, and middle bands and what that "two standard deviations" looks like. The results were similar across the automotive stocks. **NOTE**: the first 15 days are "0".

TRAINING THE DATA AND MODELS

RANDOM FOREST (5:8)

After using the Random Forest Model to test/train our features and run our predictive models, our confusion matrix were, initially, almost as confused as we were.

It seems like the features we used– the rolling average crossover points, the Bollinger curves, the views, etc., showed some importance, but weren't ultimately very good price predictions. (next slide).

Model Evaluation

[66]:
```python
# Calculating the confusion matrix
cm = confusion_matrix(y_test, predictions)
cm_df = pd.DataFrame(
    cm, index=["Actual -1", "Actual 0", "Actual 1"], columns=["Predicted -1", "Predicted 0", "Predicted 1"]
)

# Calculating the accuracy score
acc_score = accuracy_score(y_test, predictions)
```

[67]:
```python
# Displaying results
print("Confusion Matrix")
display(cm_df)
print(f"Accuracy Score : {acc_score}")
print("Classification Report")
print(classification_report(y_test, predictions))
```

Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	42	24	0
Actual 0	45	79	0
Actual 1	19	44	0

```
Accuracy Score : 0.4782608695652174
Classification Report
              precision    recall  f1-score   support

        -1.0       0.40      0.64      0.49        66
         0.0       0.54      0.64      0.58       124
         1.0       0.00      0.00      0.00        63

    accuracy                           0.48       253
   macro avg       0.31      0.42      0.36       253
weighted avg       0.37      0.48      0.41       253
```

TRAINING THE DATA AND MODELS

RANDOM FOREST (6:8)

FORD

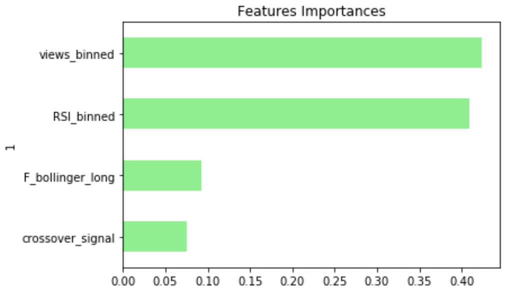
Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	4	31	0
Actual 0	14	164	0
Actual 1	7	33	0

Accuracy Score : 0.6640316205533597

Classification Report

	precision	recall	f1-score	support
-1.0	0.16	0.11	0.13	35
0.0	0.72	0.92	0.81	178
1.0	0.00	0.00	0.00	40
accuracy			0.66	253
macro avg	0.29	0.35	0.31	253
weighted avg	0.53	0.66	0.59	253



GM

Confusion Matrix

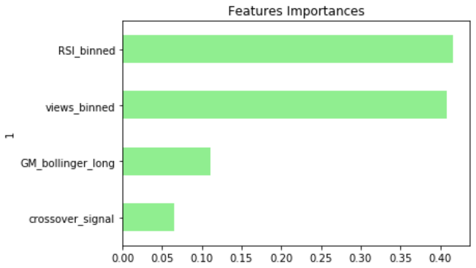
	Predicted -1	Predicted 0	Predicted 1
Actual -1	3	38	0
Actual 0	4	171	0
Actual 1	2	35	0

Accuracy Score : 0.6877470355731226

Classification Report

	precision	recall	f1-score	support
-1.0	0.33	0.07	0.12	41
0.0	0.70	0.98	0.82	175
1.0	0.00	0.00	0.00	37
accuracy			0.69	253
macro avg	0.34	0.35	0.31	253
weighted avg	0.54	0.69	0.58	253

<matplotlib.axes._subplots.AxesSubplot at 0x2338b7a4648>



KMX

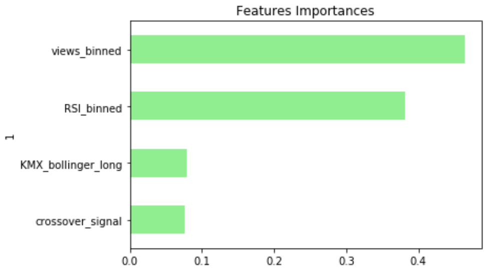
Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	1	28	0
Actual 0	17	155	15
Actual 1	10	26	1

Accuracy Score : 0.6205533596837944

Classification Report

	precision	recall	f1-score	support
-1.0	0.04	0.03	0.04	29
0.0	0.74	0.83	0.78	187
1.0	0.06	0.03	0.04	37
accuracy			0.62	253
macro avg	0.28	0.30	0.29	253
weighted avg	0.56	0.62	0.59	253



TSLA

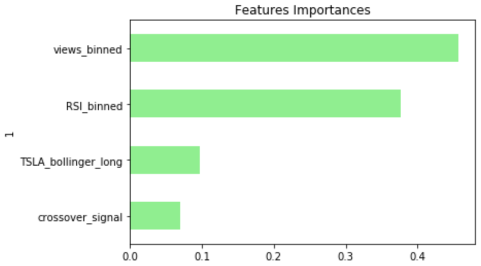
Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	17	49	0
Actual 0	31	93	0
Actual 1	27	36	0

Accuracy Score : 0.43478260869565216

Classification Report

	precision	recall	f1-score	support
-1.0	0.23	0.26	0.24	66
0.0	0.52	0.75	0.62	124
1.0	0.00	0.00	0.00	63
accuracy			0.43	253
macro avg	0.25	0.34	0.29	253
weighted avg	0.32	0.43	0.36	253



TRAINING THE DATA AND MODELS

RANDOM FOREST (7:8)

After adjusting the crossover days in the moving averages and correcting an error in the binning process we were able to refine the models to give us more sensible results (next table).

TRANSLATION: this was a more satisfying result than our previous conclusion, which would have been “Wikipedia Views are the primary driver of stock price action” and seemed wrong.

TRAINING THE DATA AND MODELS

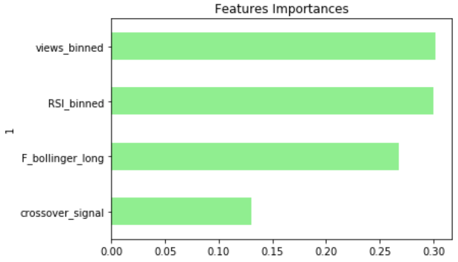
RANDOM FOREST (8:8)

FORD

Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	4	31	0
Actual 0	1	175	2
Actual 1	0	35	5

```
Accuracy Score : 0.7272727272727273
Classification Report
              precision    recall  f1-score   support

        -1.0       0.80      0.11      0.20        35
         0.0       0.73      0.98      0.84       178
         1.0       0.71      0.12      0.21        40

    accuracy                           0.73       253
   macro avg       0.75      0.41      0.42       253
weighted avg       0.73      0.73      0.65       253

<matplotlib.axes._subplots.AxesSubplot at 0x2338d693948>
```

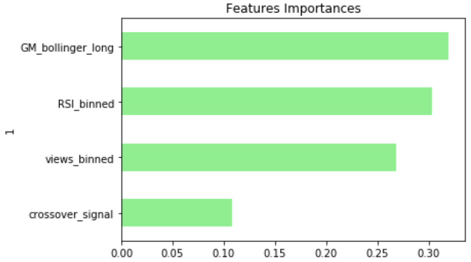


GM

Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	7	34	0
Actual 0	3	172	0
Actual 1	0	37	0

```
Accuracy Score : 0.7075098814229249
Classification Report
              precision    recall  f1-score   support

        -1.0       0.70      0.17      0.27        41
         0.0       0.71      0.98      0.82       175
         1.0       0.00      0.00      0.00        37

    accuracy                           0.71       253
   macro avg       0.47      0.38      0.37       253
weighted avg       0.60      0.71      0.61       253

<matplotlib.axes._subplots.AxesSubplot at 0x2338b92b108>
```

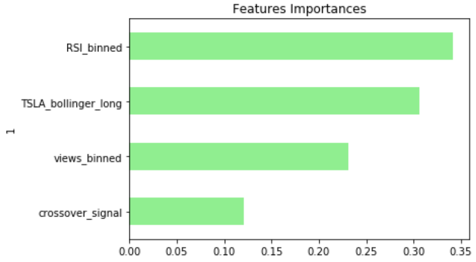


TSLA

Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	4	62	0
Actual 0	1	120	3
Actual 1	0	59	4

```
Accuracy Score : 0.5059288537549407
Classification Report
              precision    recall  f1-score   support

        -1.0       0.80      0.06      0.11        66
         0.0       0.50      0.97      0.66       124
         1.0       0.57      0.06      0.11        63

    accuracy                           0.51       253
   macro avg       0.62      0.36      0.29       253
weighted avg       0.60      0.51      0.38       253

<matplotlib.axes._subplots.AxesSubplot at 0x2338c05c688>
```

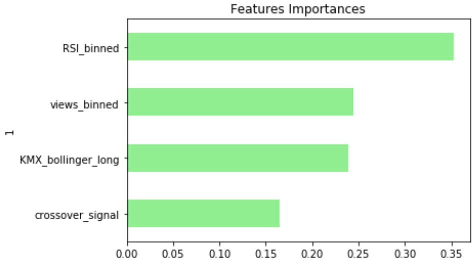


KMX

Confusion Matrix

	Predicted -1	Predicted 0	Predicted 1
Actual -1	5	24	0
Actual 0	3	184	0
Actual 1	0	37	0

```
Accuracy Score : 0.7470355731225297
Classification Report
              precision    recall  f1-score   support

        -1.0       0.62      0.17      0.27        29
         0.0       0.75      0.98      0.85       187
         1.0       0.00      0.00      0.00        37

    accuracy                           0.75       253
   macro avg       0.46      0.39      0.37       253
weighted avg       0.63      0.75      0.66       253

<matplotlib.axes._subplots.AxesSubplot at 0x2338b9fab08>
```

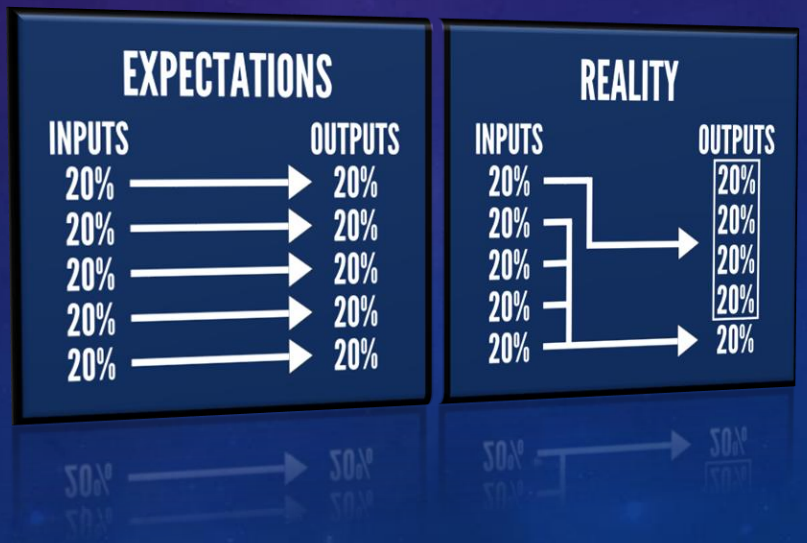


WHAT WE LEARNED

CONCLUSION

Our biggest lesson learned was that sourcing usable data can be a challenge. While data sources say API's are available, there is a wide range of quality that ranges from “useless and awful” to actually quite useful, and it’s not always obvious which is which.

Further, running deep learning models and tweaking features small amounts can add a lot to the accuracy of the model. We hear the phrase, “garbage in, garbage out”, and that’s very applicable here



JONATHAN BARAJAS // JO BORRAS // GENKI HIRAYAMA // VINCE PACILIO
MODULE 2 // GROUP 3