Talk Algo to Me:
Applying algorithmic trading and natural language processing for better investment decisions

Rawad Habib, Nigil Jeyashekar,
Ken Lindgren, Jacinta Oduor, Aparna Pooleri

Motivation

Our motivation for this project stems from the need to supplement traditional trading methods with machine learning and deep learning techniques to make better trading decisions as investors.

We were interested in combining a technical analysis of a stock with the overall sentiment of it, and we chose stocks that interested us.

Definitions

Algorithmic Trading - a process for executing stock market orders utilizing automated and pre-programmed trading instructions to account for variables such as price, timing, and volume.

Moving Averages - a calculation used to analyze data points by creating a series of averages of different subsets of the full data set. In finance, a moving average (MA) is a stock.

Window Size - represents a number of samples, and a duration, simply put, how much data (in bytes) the receiving device is willing to receive at any point to control the flow of data, or as a flow control mechanism

Entry/Exit - Entry point refers to the price at which an investor buys or sells a security-usually a component of a predetermined trading strategy for minimizing investment risk and removing the emotional trading decision. An exit strategy is a contingency plan that is executed by an investor to sell the stock once predetermined criteria has been met.

Plan of Action

Goal: Test the effectiveness of Natural Language Processing (NLP) in supplementing Algorithmic Trading Decisions.

- Choose four stocks to use in **different sectors**, then others for comparison.
- Determine the period combinations of **moving averages** we want to use.
- Apply **Algorithmic Trading** to determine **buy/sell/hold trade decisions** on stocks.
- Apply **NLP** on news articles centered around the trading day to obtain **sentiment scores.**
- **Combine** NLP sentiment scores with Algorithmic Trading results to **make better trading decisions** for investors.

Tickers

We wanted to pick four starter tickers in different sectors such as:

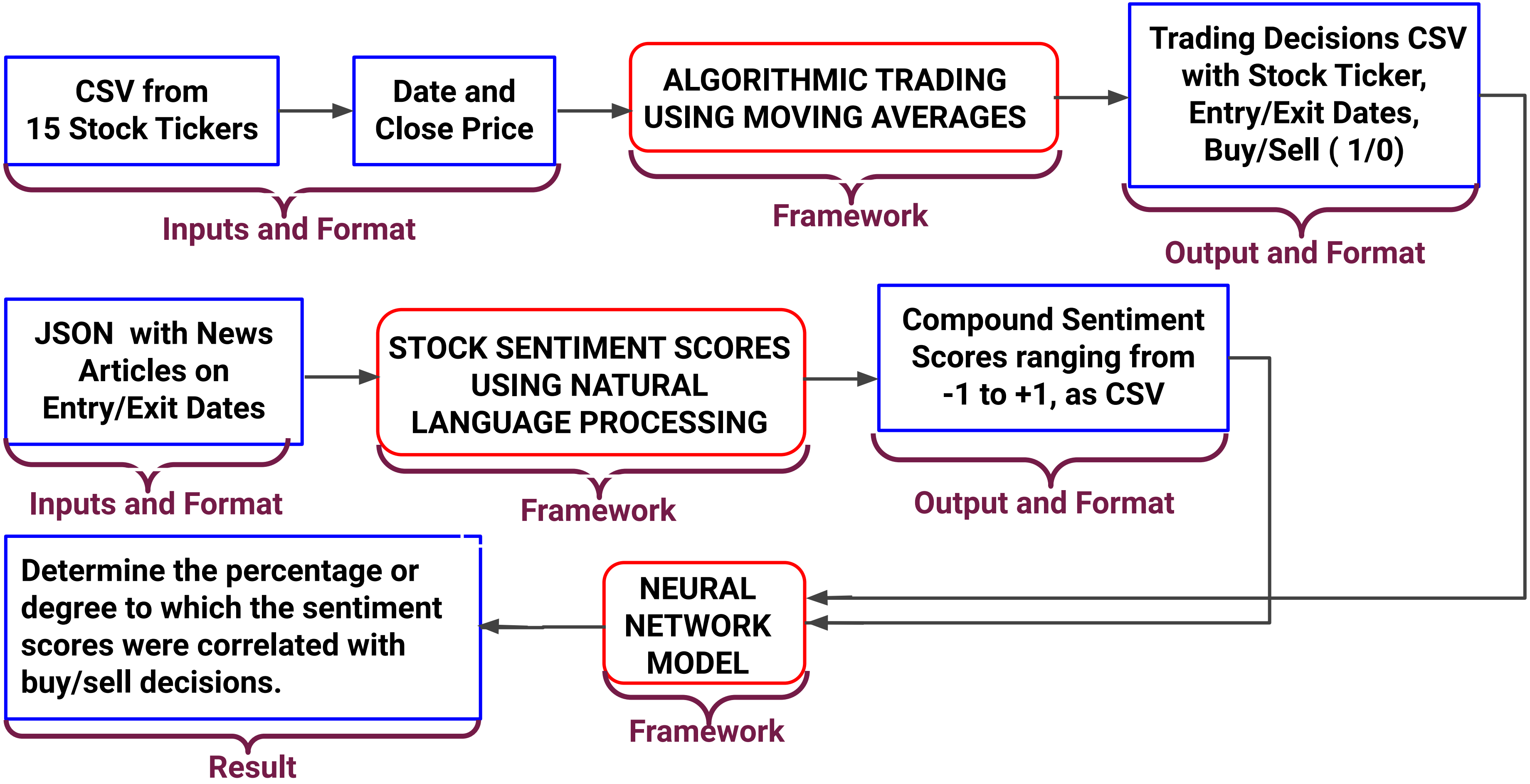
- Technology (AAPL)
- Cryptocurrency (BTC)
- Healthcare (MRNA)
- Bonds (TNX)

Then we added other stocks to use in our models.

Stock Symbol	Stock Name
AAPL	Apple
AMZN	Amazon
BTC	Bitcoin
FB	Facebook
GOLD	Gold
GOOG	Google
JPM	JPMorgan Chase & Co
MA	Mastercard
MRNA	Moderna
MSFT	Microsoft
SLV	Silver
TNX	Treasury Yield 10 Year Bond
TSLA	Tesla
TWTR	Twitter
V	Visa

CODING DEVELOPMENT, FRAMEWORK, AND IMPLEMENTATION

Code Development and Framework



Algorithmic Trading Framework Implementation

Step 1: Importing Libraries and Dependencies: numpy, pandas, hvplot, pathlib, and matplotlib.
Step 2: Input parameters and a utility script calls the “trade_evaluation” function to generate trading decisions CSV with Stock tickers, Entry/Exit Dates, Buy/Sell Decisions.

```
# Input Parameters

# Stock Source File Names
stock_name = ['AAPL','BTC-USD','JPM','MA','MRNA','TNX','V','AMZN','FB','Gold','GOOG','M

# Set the short window and long trade windows
short_window = [9, 21, 9, 50, 21, 50]
long_window = [21, 50, 50, 100, 100, 200]

# Set initial capital
initial_capital = float(100000)

# Set the share size
share_size = 500

# Building a composite dataframe
for i in range(len(stock_name)):
    for j in range(len(short_window)):
        temp = trade_evaluation(stock_name[i], short_window[j], long_window[j], share_s
        if len(temp['Exit Portfolio Holding'])>0 and temp['Exit Portfolio Holding'][(le
                composite_trade_evaluation_df = composite_trade_evaluation_df.append(te

composite_trade_evaluation_df = composite_trade_evaluation_df.reset_index()
composite_trade_evaluation_df
```

```
# Function to Read Source CSV

def trade_evaluation(stock_name, short_window, long_window, share_size, initial_capital

    # Read the CSV located at the file path into a Pandas DataFrame
    filepath = Path('Resources/'+str(stock_name)+'.csv')

    # Set the `Date` column as the index and auto-format the datetime string
    stock_csv = pd.read_csv(filepath, parse_dates=True, infer_datetime_format=True)
    stock_csv = stock_csv.dropna()

    # Set short and long window
    str_short="SMA"+str(short_window)
    str_long="SMA"+str(long_window)
```

Stock	Date	Buy/Sell
JPM	2/5/2019	1
JPM	2/21/2019	1
JPM	3/19/2019	1
JPM	4/9/2019	1
JPM	6/13/2019	1

JPM	2/12/2019	0
JPM	3/11/2019	0
JPM	3/25/2019	0
JPM	5/14/2019	0
JPM	8/6/2019	0

NLP Stock Sentiment Score Framework Implementation

Step 1: Relevant Libraries and Dependencies: nltk and SentimentIntensityAnalyzer()
Step 2: JSON source files with news articles, Pre-processing Data, and JSON File Name Generator.

```
# Sourcing and Preprocessing Input Data
filepath = Path("NLP_Resource/Resource/_COMPOSITE2.csv")
algo_results_df = pd.read_csv(filepath, parse_dates=True, infer_datetime_format=True)
for i in range(len(algo_results_df)):
    dt=dateutil.parser.parse(algo_results_df['Date'][i])
    mm=dt.month
    dd=dt.day
    yyyy=dt.year
    if mm<10:
        mm='0'+str(mm)
    if dd<10:
        dd='0'+str(dd)
    algo_results_df['Date'][i]=str(mm)+str(dd)+str(yyyy)
algo_results_df.head()
```

	Stock	Date	Buy/Sell	JSON File Name
0	JPM	02052019	1	JPM02052019
1	JPM	02212019	1	JPM02212019
2	JPM	03192019	1	JPM03192019
3	JPM	04092019	1	JPM04092019
4	JPM	06132019	1	JPM06132019

```
# JSON Source File Name Generation
temp=[]
for i in range(len(algo_results_df)):
    temp.append(algo_results_df['Stock'][i]+algo_results_df['Date'][i])
algo_results_df['JSON File Name']=temp
algo_results_df.head()
```

NLP Stock Sentiment Score Framework Implementation

Step 3: Calculating Sentiment Scores from JSON source files using Functions.

```
# Function to Calculate Sentiment Scores
def stock_score(json_path):
```

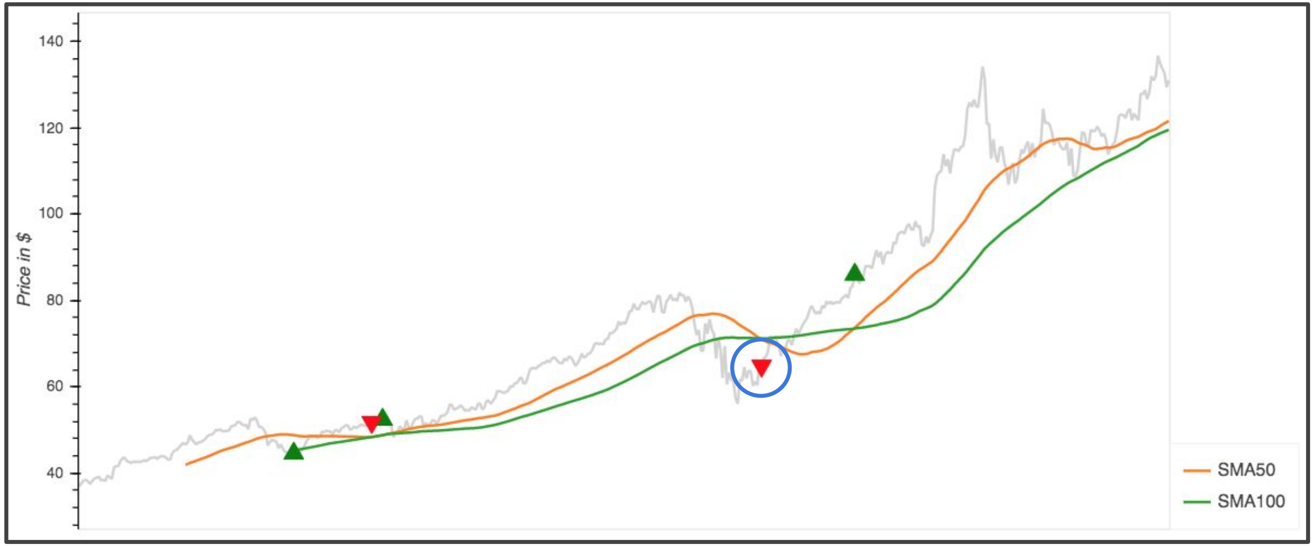
```
    temp_data=[]
    for i in range(0,len(new_data['data'])):
        temp=(new_data['data'][i]['title'])+' '+(new_data['data'][i]['text'])
        temp_data.append(temp)
    data=String(temp_data)
    sentiment = analyzer.polarity_scores(data)
    compound = sentiment["compound"]

return compound
```

```
# Script to Calculate Compound Scores
sentiment_score=[]
for i in range(len(algo_results_df)):
    json_path="NLP_Resource/"+algo_results_df['JSON File Name'][i]+'.json'
    sentiment_score.append(stock_score(json_path))
algo_results_df['Sentiment Score']=sentiment_score
algo_results_df.head()
```

	Stock	Date	Buy/Sell	JSON File Name	Sentiment Score
0	JPM	02052019	1	JPM02052019	NaN
1	JPM	02212019	1	JPM02212019	-0.4767
2	JPM	03192019	1	JPM03192019	0.9075
3	JPM	04092019	1	JPM04092019	0.5236
4	JPM	06132019	1	JPM06132019	NaN

SUMMARY: ALGORITHMIC TRADING AND NLP SENTIMENT SCORE FRAMEWORKS

AAPL

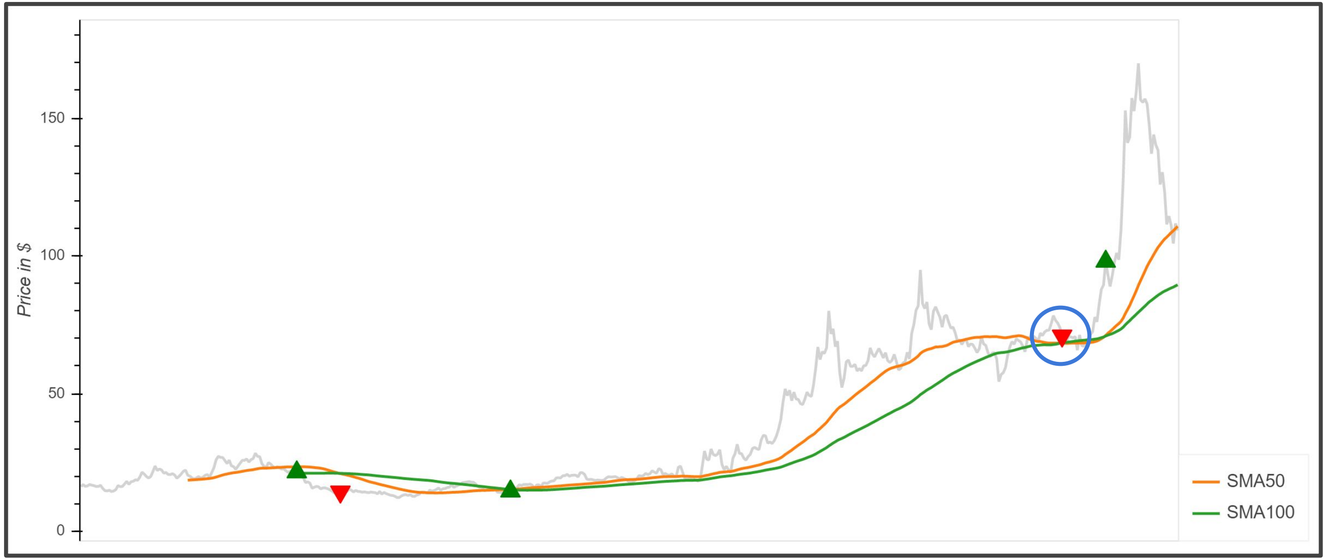


04/07/2020

MA: Sell
Price: $64.86
NLP Sentiment: 0.959

Using sentiment:
- Ignore Sell signal
- Ignore Buy at $86.00
- Saves $15.14 / share

MRNA



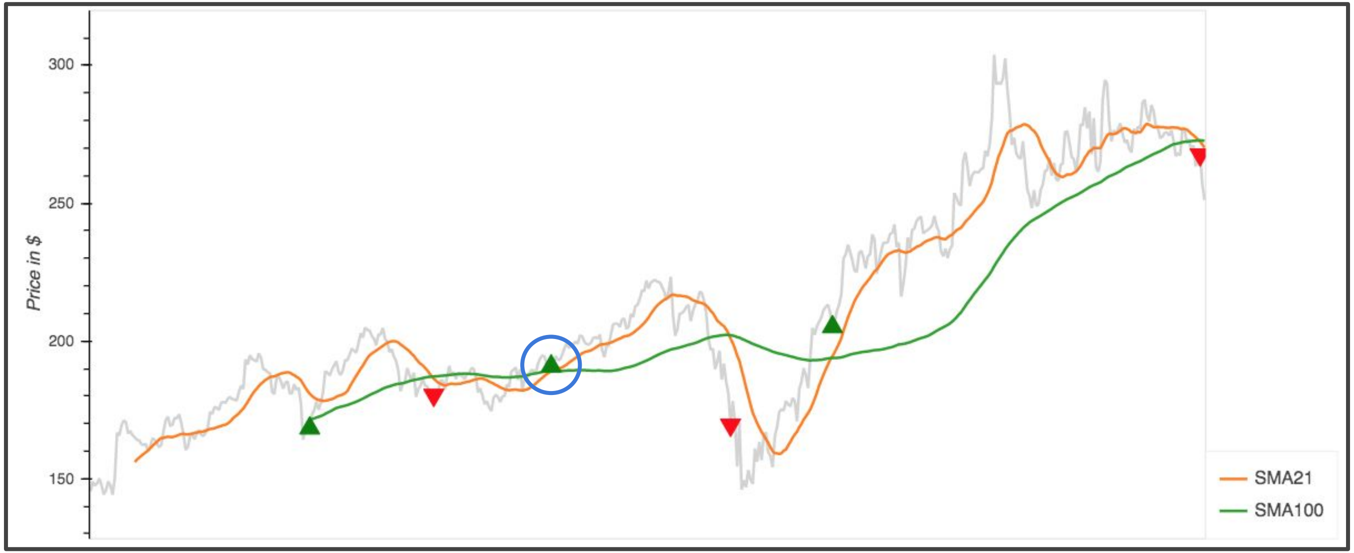
10/19/2020

MA: Sell
Price: $70.96
NLP Sentiment: 0.632

Using sentiment:

- Ignore Sell signal
- Ignore Buy at $97.95
- Saves $27.95 / share

FB



11/08/2019

MA: Buy
Price: $190.84
NLP Sentiment: -0.731

Using sentiment:

- Ignore Buy signal
- Ignore Sell at $169.50
- Saves $21.34 / share

Results Recap and Neural Network Implementation

How are NLP Sentiment Scores useful?

- Moving average results sometimes indicate sell decisions when the sentiment scores are highly positive.

- Therefore, sentiment scores can be applied as an *additional tool* to the trader to make hold decisions when the moving average trading decision provides a contradictory sell decision.

- Artificial Neural Network Model will be used to determine the percentage/degree to which the sentiment scores were correlated with buy/sell decisions.

Step 1: Reading in Source Data.

```
# Read in data
df = pd.read_csv("Sentiment_Scores_1.csv")
sentiment_scores = df.dropna()
```

Neural Network Model Implementation

Step 2: Setting Features and Targets.

```
X = data['Sentiment Score']
y = data['Buy/Sell']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state=79,
                                                    )
```

Step 3: Defining the Model Architecture.

```
# Define the model
number_inputs = 1
number_hidden_nodes = 150

nn = Sequential()
nn.add(Dense(units=number_hidden_nodes, input_dim=1, activation="relu"))
```

```
nn.add(Dense(1, activation="sigmoid"))
# Compile model
nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

Neural Network Model Implementation

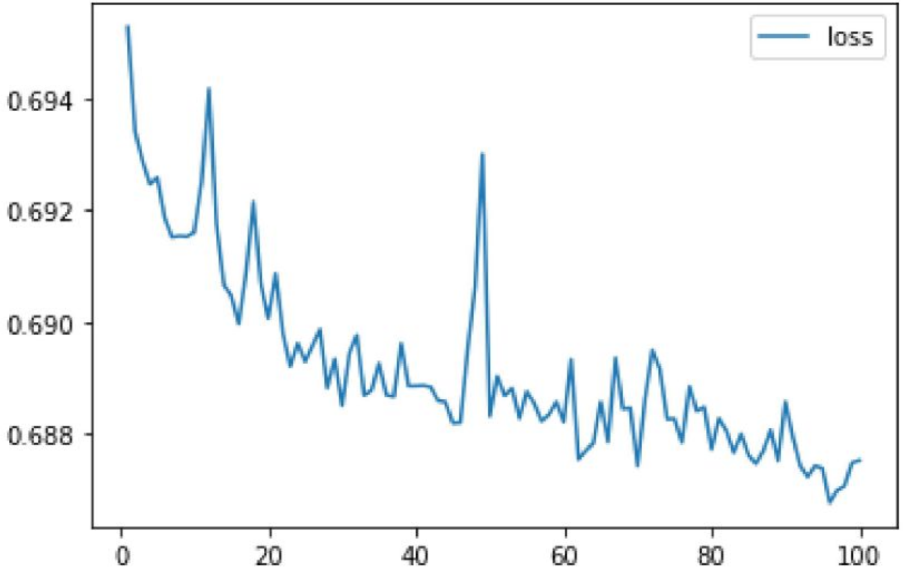
Step 4: Model Training, and Evaluation

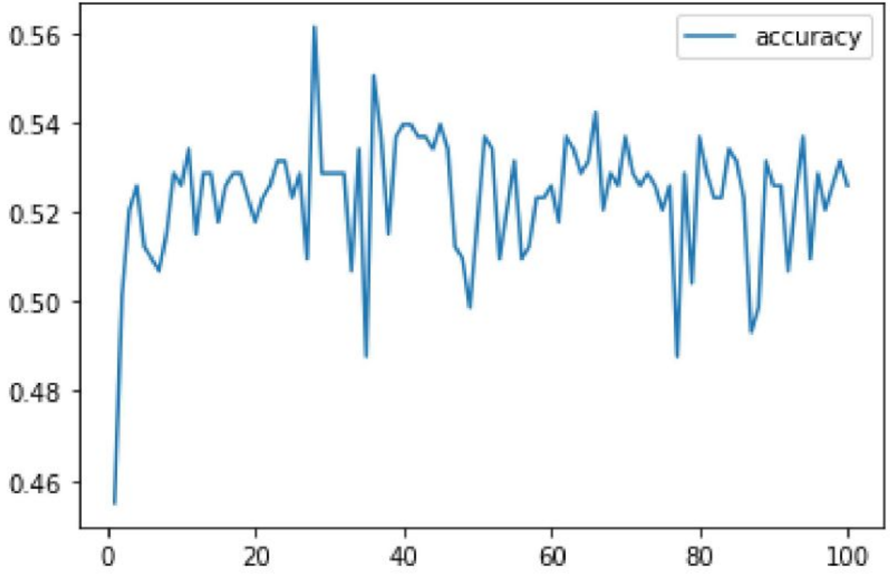
```
# Fit the model
model = nn.fit(X_train, y_train, epochs=100)
```

```
model_loss, model_accuracy = nn.evaluate(X_test, y_test, verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

Step 5: Results

```
4/4 - 0s - loss: 0.6732 - accuracy: 0.6423
Loss: 0.6731773018836975, Accuracy: 0.642276406288147
```





Discussion

- Algorithmic Trading using Moving Average algorithm was implemented on 15 stock tickers, 6 moving average windows to determine buy/sell trading decisions for different entry/exit date using fixed initial capital and share price.
- News articles in JSON format were sourced based on the entry/exit dates. Stock Sentiment was calculated from JSON source files based on compound scores using Sentiment Intensity Analyzer from Natural Language Toolkit (nltk).
- Buy/Sell Trading Decisions from Moving Average Algorithm and NLP Sentiment Scores were fed into an Artificial Neural Network Model to predict the correlation of NLP Sentiment Scores to Algorithmic Trading Decisions.
- Based on Model Evaluation, it was determined that approximately 64% of the time that sentiment scores matches with the buy/sell results obtained from moving average trading algorithm.

Post-Mortem

As to be expected, there were a few hurdles during our project:

- We weren't sure what the perfect amount of entry/exit points would be for the most accurate model.
- Some of the tickers, like BTC and TNX, didn't have sentiment scores.
- All of us received different accuracy scores.
    - For example, Ken got 64% and Nigil got 57%, so we went off of Ken's

If we had more time, we would expand our project by...

- Analyzing how far in the past NLP should be pulled for a more accurate representation of sentiment on price trends.
- Determining if the NLP dates should be averaged together.
- Determining an accurate stop loss when sentiment and moving average cross are non-correlated but sentiment inaccurately predicts price trend.

Questions?