ELON MUSK TWITTER ANALYSIS TIME SERIES TESLA PRICE PREDICTION

BY: JUSTIN HUANG

Elon's Tweets

- Hypothesis
  - Business Tweets - 1352 tweet observations
  - Personal Tweets - 400 tweet observations
  - No Tweets - 928 tweet observations

Kruskal-Wallis H Test

H_0: All sample distributions are equal.

H_A: One or more sample distributions are not equal.

alpha: 0.05

Kruskal-Wallis H-test Stat	p-value
241.164	0.000

Tesla CEO Elon Musk tweeted Friday that **Tesla's stock price** is "**too high imo**," and the **stock** fell immediately after. **Tesla's stock** is down more than 10 percent. May 1, 2020

Data Wrangling and Cleaning

- GetOldTweets3
  - Collected Old Tweets from Elon's first Tweet Jun 4th , 2010 till July 31, 2020.
- Pandas datareader
  - Financial Data collected through yahoo finance API
- Text Wrangling
  - Used Regex for Key Business Words then reviewed over manually
  - Then Applied Sentiment labeling with NLTK vader.
- Numeric Wrangling
  - Loaded in Financial data from June 29th , 2010 to July 31, 2020.
- Combining Text with Numeric
  - Combined on Date and decided to go from 2011-12-1 to 2020-7-31 since before that time Elon didn't tweet.

	Adj Close	retweet_count	fav_count	tweetLen		Business positive	Business neutral	Business negative	Personal positive	Personal neutral	Personal negative	Volume
2020-07-27	1476.489990	154954	1337692	145	4	0	0	9	10	2		16048700.0
2020-07-28	1499.109985	1245	28715	73	0	0	1	3	2	2		15808700.0
2020-07-29	1487.489990	5303	100007	79	1	0	0	3	5	0		9426900.0
2020-07-30	1430.760010	91933	1162051	309	4	0	4	10	7	3		7621000.0
2020-07-31	1430.760010	85083	770873	70	0	0	0	5	3	1		7621000.0

Classical Classification

Text Preprocessing

```
def clean_tweet(tweet):
    #remove stopwords
    #use beautiful soup to remove the &/amps etc in tweets as well as website links
    soup_ = BeautifulSoup(tweet, 'lxml')
    soup_ = soup_.get_text()
    soup_ = re.sub(r'https?://[A-Za-z0-9./]+', '', soup_)

    #lowercase the words and remove punctuation
    lower_ = ''.join([word.lower() for word in soup_])

    #remove puncutations using a custom list
    punc_ = ''.join([punc(word) for word in lower_])
    #tokenize
    token_ = re.split('\W+',punc_)
    #remove stopwords
    stop_ = [word for word in token_ if word not in stopwords]
    tweet = ' '.join(word for word in stop_)

    return tweet
```

Classical Methods

Algorithm	Word Vector	f1-score weighted avg	f1-score macro avg	Accuracy
SVM	tfidf	0.11	0.07	0.26
SVM	countvect	0.11	0.07	0.26
MultinomialNB	tfidf	0.41	0.31	0.48
MultinomialNB	countvect	0.48	0.42	0.51
Logistic Regression	tfidf	0.57	0.48	0.60
Logistic Regression	countvect	0.61	0.55	0.63
RandomForest Classifier	tfidf	0.57	0.50	0.60
RandomForest Classifier	countvect	0.54	0.48	0.58
xgBoost Classifier	tfidf	0.60	0.53	0.62
xgBoost Classifier	countvect	0.59	0.53	0.62

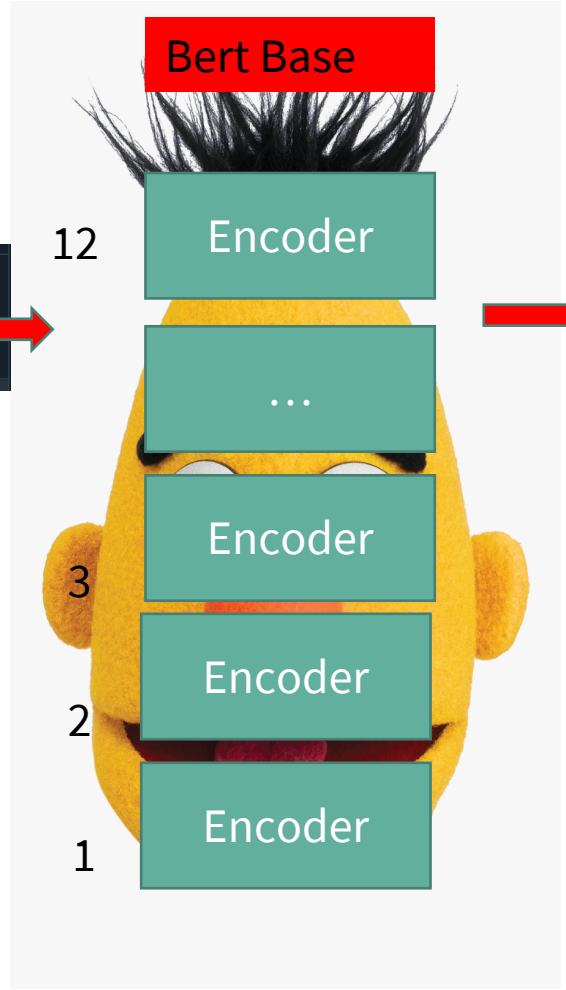
Hyperparameter Tuning

Algo	best score	parameters tuned	paramter value
Logistic Regression	0.62322	countvectorizer max features, max_iter, ngram range, C	7000, 700, (1,1), 31
xgBoost Classifier	0.58142	tfidfvectorizer max features, ngram range, max_depth, n estimators, min child weight	2000, (1, 3), 700, 800, 2

Bidirectional encoder representations from transformers

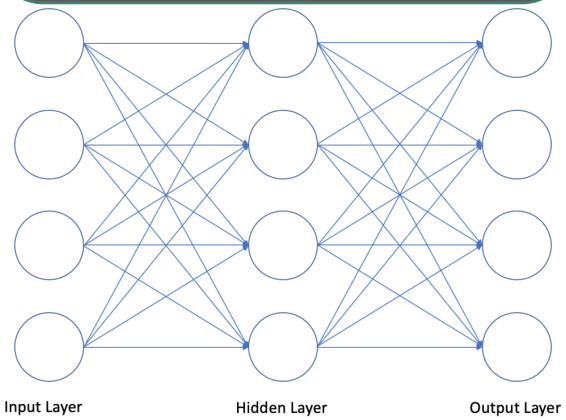
Tweets

Bert Base



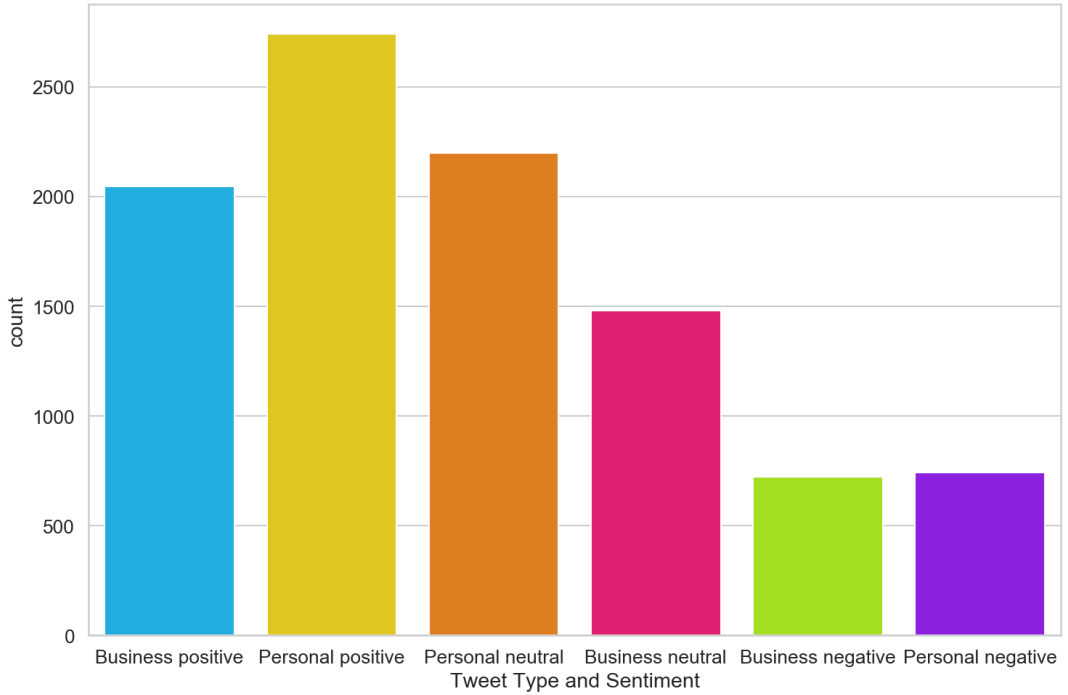
Classifier

Feed Forward Neural Network + Softmax



0 %	Business Positive
15%	Business Neutral
85%	Business Negative
0 %	Personal Positive
0 %	Personal Neutral
0 %	Personal Negative

BERT and DistilBERT

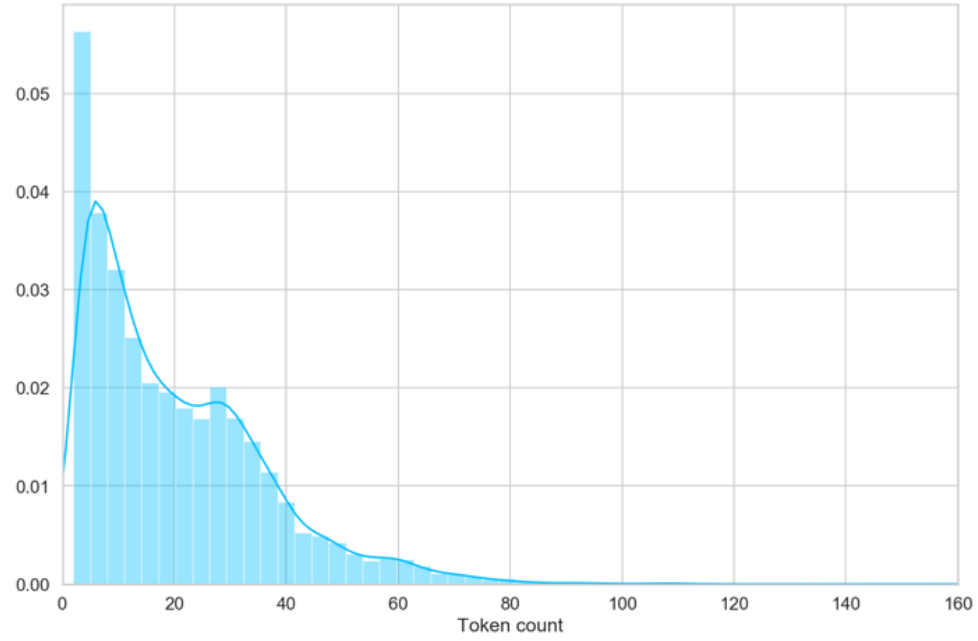


Bert will convert the text to tokens.
Special Tokens:

- CLS – BERT knows it's a classification
- SEP – marker end of sentence
- PAD – Token for padding
- UNK – Understands token in training set everything else will marked as UNK

['[CLS]', 'When', 'the', 'zombie', 'apocalypse', 'happens', 'you', ''' 'll' 'be' 'glad' 'you' 'bought' 'a' 'flamethrower' '.' '[SEP]' 'Works' 'against' 'hordes' 'of' 'the' 'undead' 'or' 'your' 'money' 'back' '!' '[SEP]', '[PAD]']

DistilBERT



DistilBERT! BERT with 44 million less parameters who needs a nose! 2 GB model vs 280 mb model

- Trained with 80 tokens
- Batch size was 16
- Transfer Learning 10 epochs
- drop out at 0.3
- Adam Optimizer lr 2e-5

```
#get classification report
print(classification_report(y_test, y_pred, target_names = class_names
```

	precision	recall	f1-score	support
Business negative	0.62	0.55	0.58	75
Business neutral	0.70	0.82	0.76	136
Business positive	0.81	0.83	0.82	225
Personal negative	0.70	0.78	0.74	58
Personal neutral	0.88	0.84	0.86	238
Personal positive	0.88	0.83	0.85	262
accuracy			0.81	994
macro avg	0.77	0.77	0.77	994
weighted avg	0.81	0.81	0.81	994

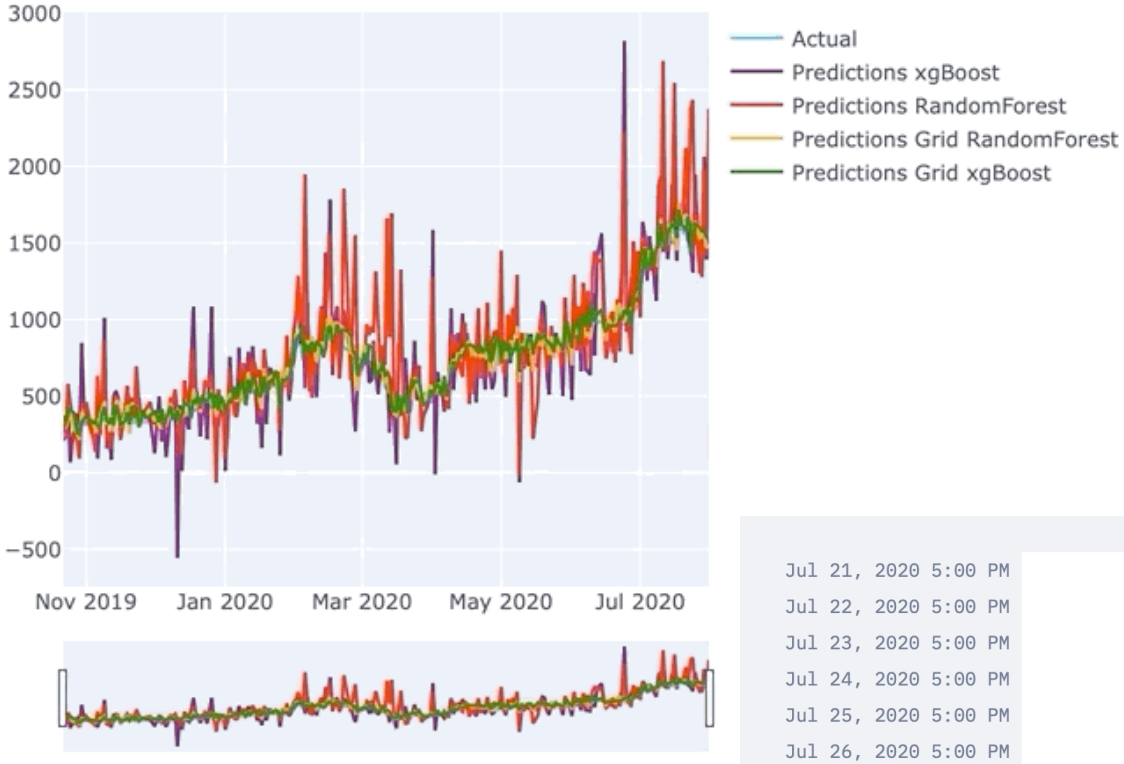
```
data2 = ['Congrats Tesla Team - U ROK!!',
         'Happy 4th of July!!',
         'Please take a moment to report accounts clearly engaged in harassment. It is the only way to maintain p
         'Side note: Chomsky sucks',
         'North American Supercharger usage is now at pre-covid high, Europe about a week behind, China & Asia-Pa
         'Beautiful fireworks in LA tonight',
         'Limited edition short shorts now available at',
         'Only $69.420!!',
         'Dang, we broke the website',
         'Read The Story of Civilization by Will & Ariel Durant']
```

```
reloaded_predictor.predict(data2)
```

```
['Business positive',
 'Business positive',
 'Personal negative',
 'Personal negative',
 'Business positive',
 'Personal positive',
 'Personal negative',
 'Business neutral',
 'Personal neutral',
```

Time Series Prep and Classical Models

RF and XGBoost



Dickey Fuller Test

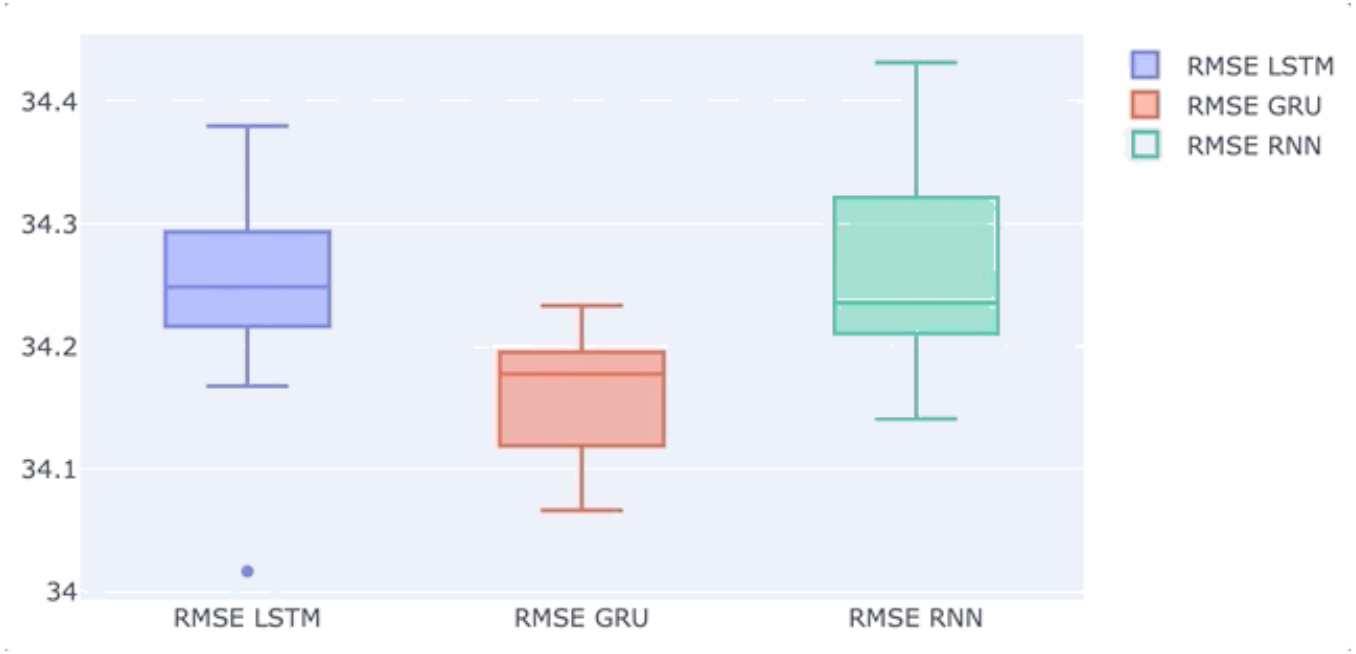
```
   Augmented Dickey-Fuller Test on "Adj Close"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -10.5447
No. Lags Chosen       = 28
Critical value 1%     = -3.433
Critical value 5%     = -2.863
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


   Augmented Dickey-Fuller Test on "retweet_count"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -17.4791
No. Lags Chosen       = 27
Critical value 1%     = -3.433
Critical value 5%     = -2.863
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
```

- First, we used the Dickey Fuller Test to test if the data was stationary.
- first difference
- Scaled the data using MinMaxScaler (-1 to 1)

	Adj Close	Predictions xgB	Predictions RF	Predictions gridRF	Predictions gridXGB
Jul 21, 2020 5:00 PM	1,513.0699	1,522.8699	1,613.0688	1,663.4145	1,674.5514
Jul 22, 2020 5:00 PM	1417	1,477.8072	2,371.3485	1,603.9026	1,614.2341
Jul 23, 2020 5:00 PM	1,539.6000	1,306.7477	2,434.6067	1,523.1501	1,427.5184
Jul 24, 2020 5:00 PM	1,518.5633	1,947.3002	1,489.5970	1,643.8980	1,624.1561
Jul 25, 2020 5:00 PM	1,497.5267	1,546.9496	1,688.1069	1,603.6173	1,619.7274
Jul 26, 2020 5:00 PM	1,476.4900	1,303.7742	1,495.5311	1,521.9383	1,598.6908
Jul 27, 2020 5:00 PM	1,499.1100	1,315.3640	1,276.4710	1,571.2491	1,577.6541
Jul 28, 2020 5:00 PM	1,487.4900	2,061.3078	1,984.2163	1,556.1579	1,574.4478
Jul 29, 2020 5:00 PM	1,430.7600	1,391.6629	1,414.0648	1,467.3171	1,539.2197
Jul 30, 2020 5:00 PM	1,430.7600	1,536.6747	2,375.1479	1,486.7384	1,531.9241

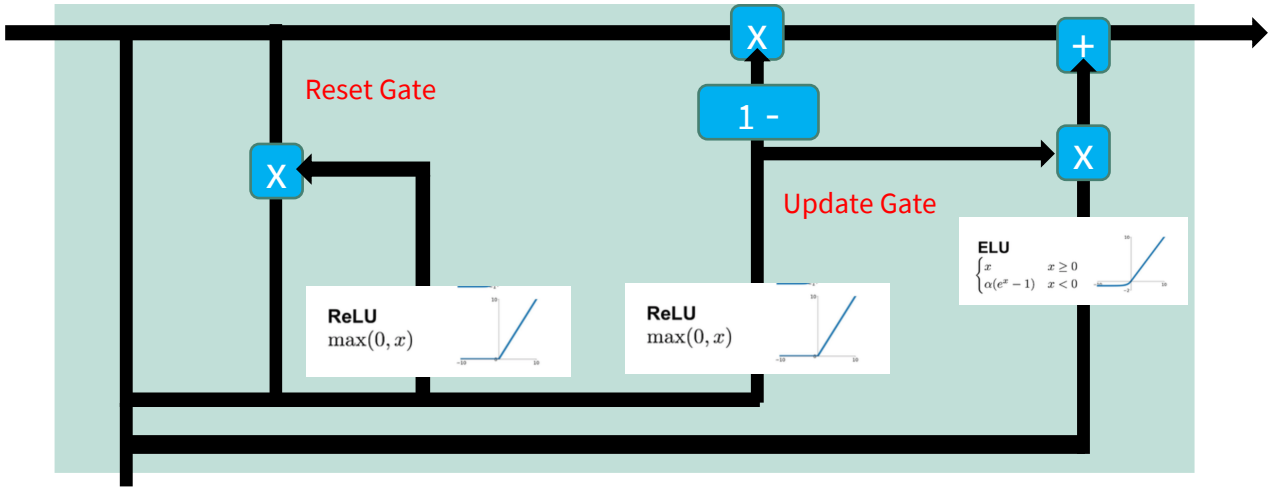
LSTM GRU Vanilla RNN

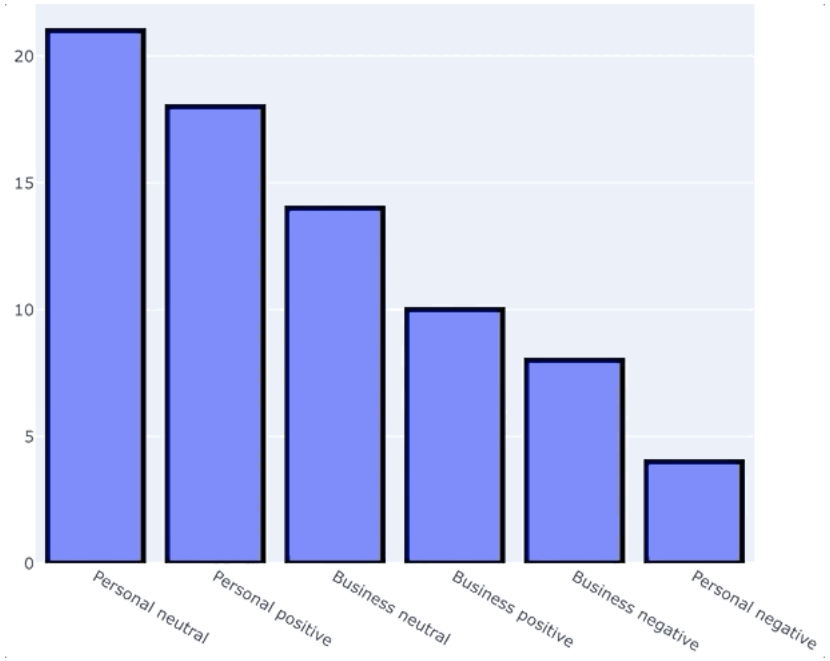


	Adj Close	Predictions RNN	Predictions GRU	Predictions LSTM
Jul 21, 2020 5:00 PM	1,513.0699	1,592.0158	1,593.5771	1,592.7886
Jul 22, 2020 5:00 PM	1417	1,516.8167	1,516.6201	1,528.2539
Jul 23, 2020 5:00 PM	1,539.6000	1,420.9523	1,418.7486	1,415.6421
Jul 24, 2020 5:00 PM	1,518.5633	1,535.9294	1,546.6326	1,536.3072
Jul 25, 2020 5:00 PM	1,497.5267	1,518.5666	1,518.4226	1,518.1289
Jul 26, 2020 5:00 PM	1,476.4900	1,498.1283	1,497.7016	1,495.8783
Jul 27, 2020 5:00 PM	1,499.1100	1,478.6246	1,477.4278	1,477.8969
Jul 28, 2020 5:00 PM	1,487.4900	1,499.2004	1,499.5888	1,500.3470
Jul 29, 2020 5:00 PM	1,430.7600	1,487.0665	1,487.7107	1,486.8862
Jul 30, 2020 5:00 PM	1,430.7600	1,434.7386	1,432.3531	1,432.8982

- RNN is good for sequence data
  - Text, audio, time series
  - A sequence can be like the Alphabet
- Problem with RNN it suffers from the vanishing gradient problem and forgets earlier time steps
  - Vanishing gradient happens in backpropagation when the network learns from the loss function.
- To solve this short-term memory LSTM and GRU was developed to include ”gates”
  - These gates control the flow of information so earlier steps can flow to the very end.

Gated Recurrent Units

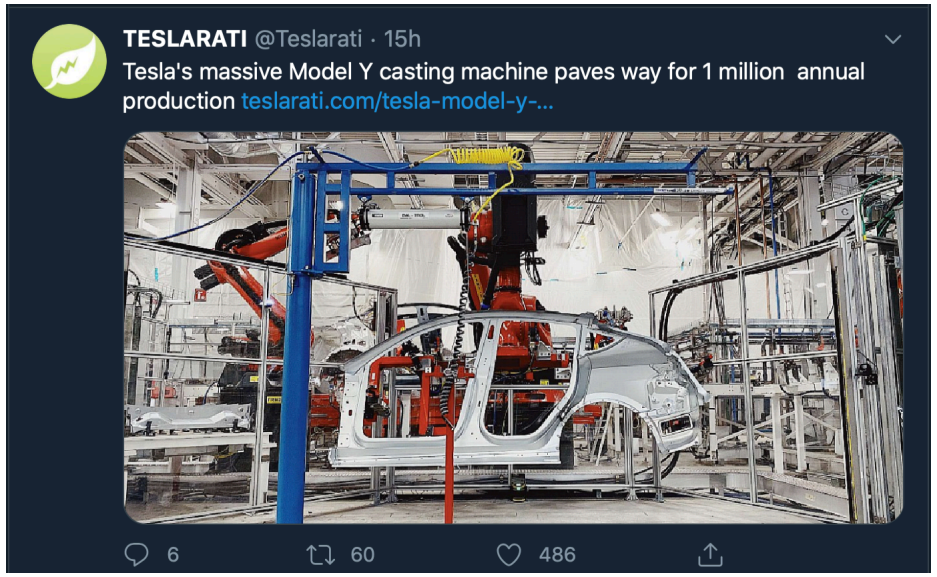




- Adam optimizer learning rate 0.0001
- Batch Size 8
- Neurons 500
- Loss – mean squared error
- Dropout – 0.3
- Acivation – ELU , recurrent activation ReLu

Future Work to think about

Include SEC Edgar 8k information



Include a business side tweeter

Instead of classifying as business and personal classify as 3 types of losses and gains