



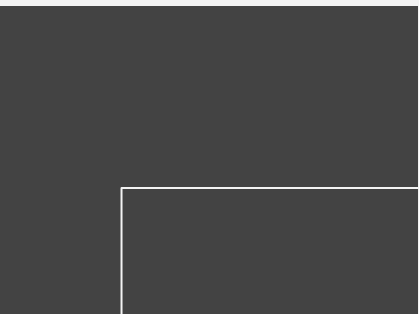
NEW YORK STOCK EXCHANGE

Better Predicting the Stock Market

By: Lauren Esser

PROBLEM STATEMENT

**Can we use News Headlines to better predict the
S&P 500?**



By creating a model that gives a high success rate of stock market predictions we can invest our money wisely to make good profits.



Business Value



Data

Stock Market: www.kibot.com/free_historical_data.aspx

Stock Market data comes from kibot.com which provides free historical intraday data on the S&P 500 dating back to September 2009.

News: <https://www.kaggle.com/rmisra/news-category-dataset>

News dataset contains around 200k news headlines from 2012 to 2018 obtained from the Huffington Post.

THE OSEMN PROCESS

OBTAIN - Data was obtained on
Huffington Post News Headlines and the
S&P 500

The logo for The Huffington Post, featuring the words "THE", "HUFFINGTON", and "POST" stacked vertically in a green, serif font. The background is a white rectangular box.

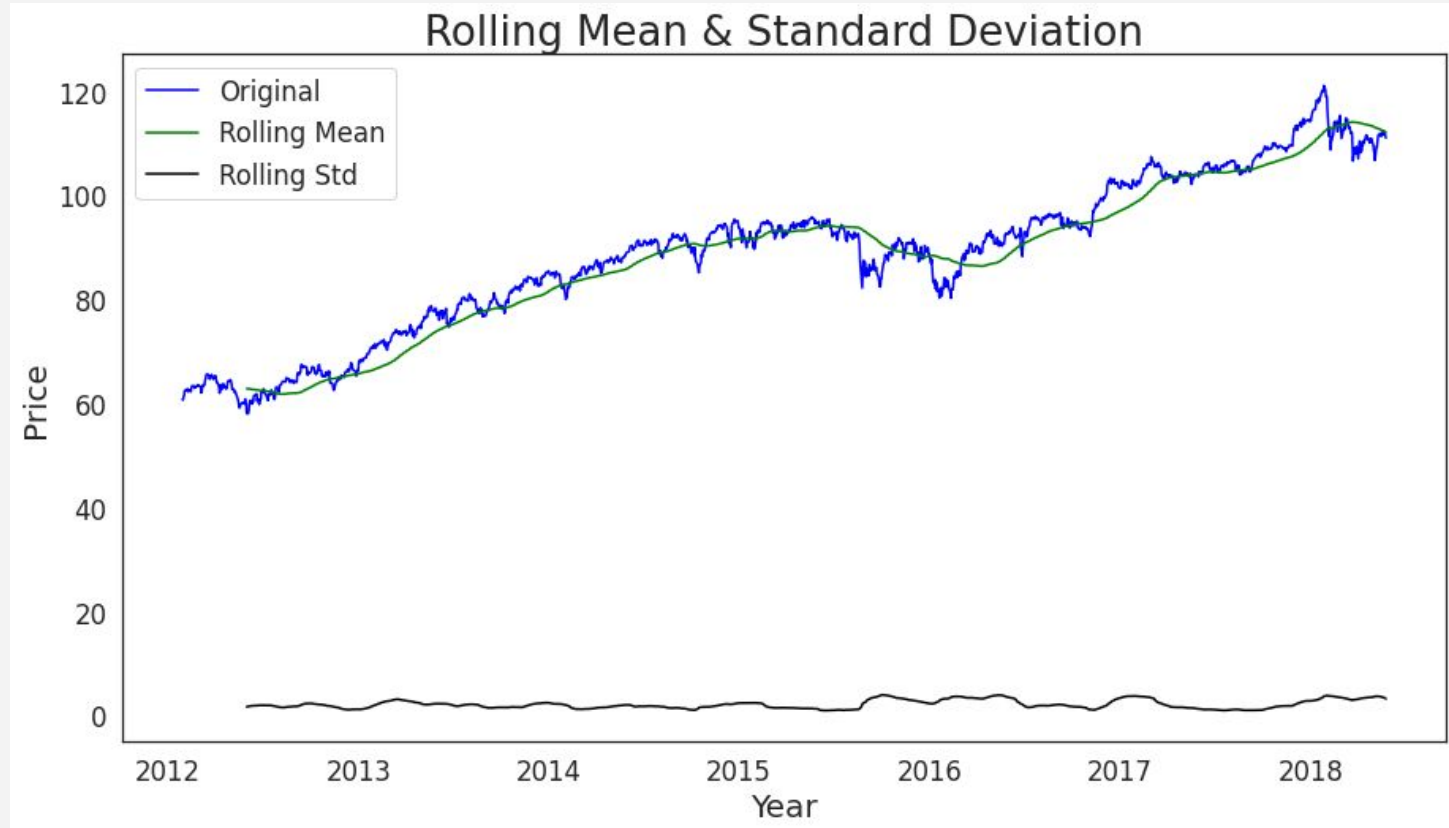
THE
HUFFINGTON
POST

The logo for Standard & Poor's 500, featuring the words "STANDARD", "& POOR'S", and "500" stacked vertically. "STANDARD" and "& POOR'S" are in a black, serif font, and "500" is in a larger, bold, black, sans-serif font. The background is a white rectangular box.

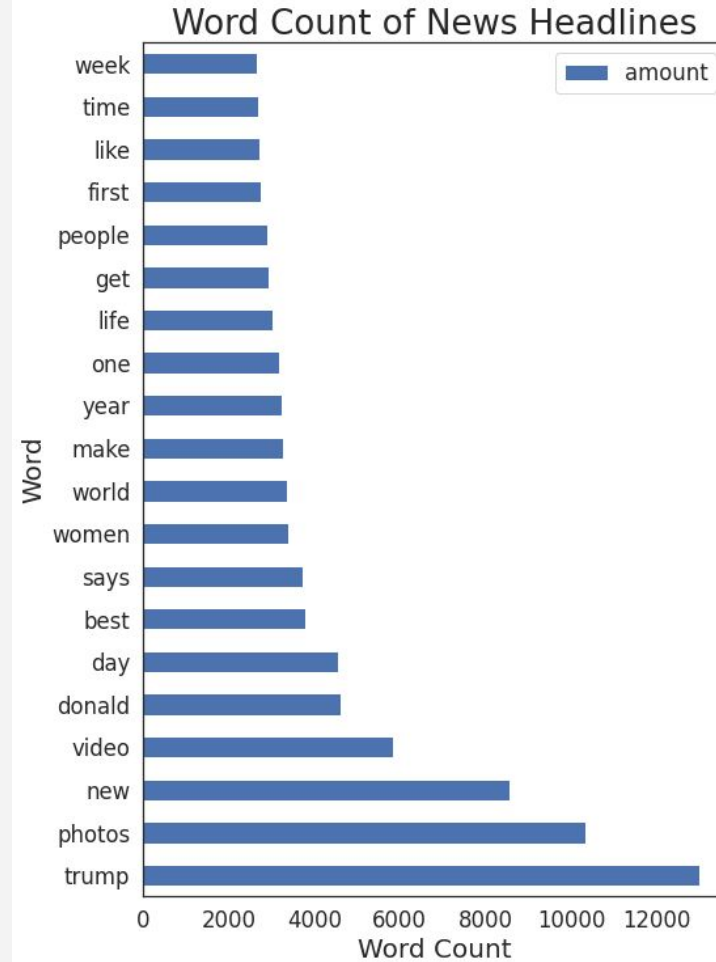
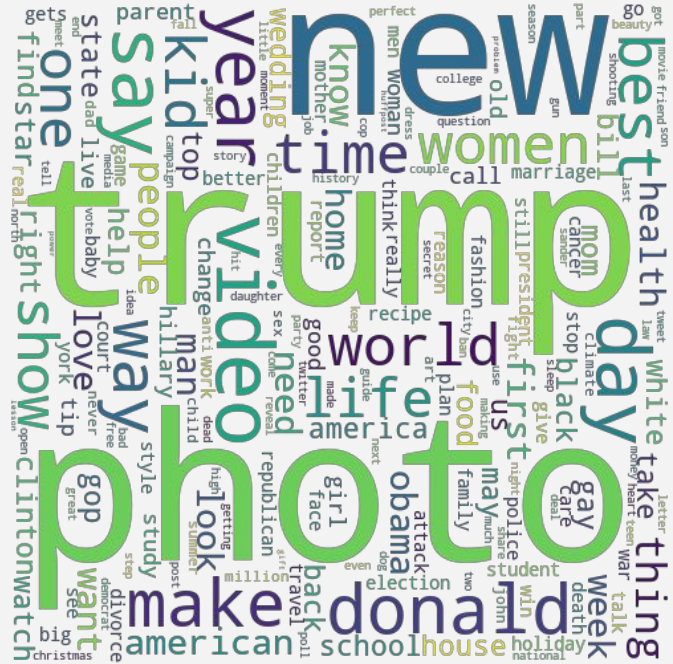
STANDARD
& POOR'S **500**

SCRUB - Index converted to datetime,
checked for nulls, and identified stop
words, punctuation, and tokenizing in news
dataset.

EXPLORE STOCKS - took a look at line plots, rolling statistics, dickey-fuller, density plots, and transformations.



EXPLORE NEWS - took a look headline by genre and year, as well as the top recurring words.



MODEL 1:

Stocks Time Series

```
#plug in optimal parameter values
arima_model = sm.tsa.statespace.SARIMAX(X_train,
                                         order = (2,1,2),
                                         seasonal_order = (1,1,2,12),
                                         enforce_stationarity=False,
                                         enforce_invertibility=False)

#fit model
output = arima_model.fit()

display(output.summary())
```

Statespace Model Results

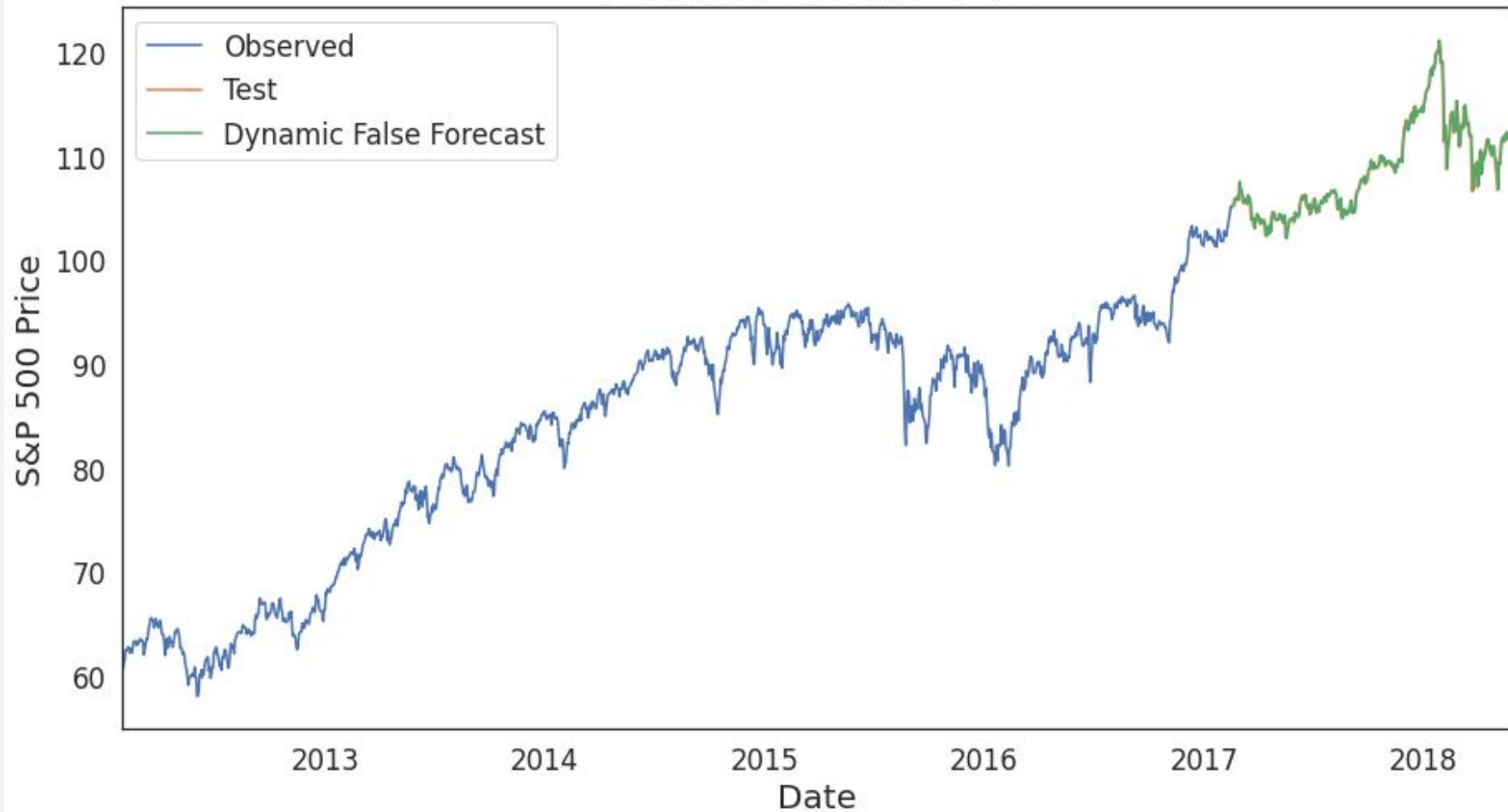
Dep. Variable:	Price	No. Observations:	1320
Model:	SARIMAX(2, 1, 2)x(1, 1, 2, 12)	Log Likelihood	3600.164
Date:	Tue, 01 Dec 2020	AIC	-7184.327
Time:	16:47:01	BIC	-7143.090
Sample:	01-30-2012 - 02-17-2017	HQIC	-7168.843

Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.8463	1.272	0.665	0.506	-1.648	3.340
ar.L2	-0.0281	1.052	-0.027	0.979	-2.090	2.034
ma.L1	-1.2201	1.351	-0.903	0.367	-3.869	1.429
ma.L2	0.0417	1.535	0.027	0.978	-2.968	3.051
ar.S.L12	-0.6949	0.283	-2.459	0.014	-1.249	-0.141
ma.S.L12	-0.3230	0.294	-1.100	0.271	-0.898	0.253
ma.S.L24	-0.6730	0.294	-2.288	0.022	-1.250	-0.097
sigma2	0.0001	5.03e-05	2.883	0.004	4.65e-05	0.000

Ljung-Box (Q): 38.05 Jarque-Bera (JB): 376.23
Prob(Q): 0.56 Prob(JB): 0.00
Heteroskedasticity (H): 2.26 Skew: -0.43
Prob(H) (two-sided): 0.00 Kurtosis: 5.51

Forecast of S&P 500



MODEL 2:

NLP CLASSIFICATION MODEL USING STOCK DATA:

53% Accuracy

```
model = Sequential()
model.add(Embedding(max_words, 300)) #can change 100 for how many datapts
model.add(LSTM(64, activation = 'relu', return_sequences=True,
              kernel_regularizer=regularizers.l1(0.001)))

model.add(Dropout(0.3))
model.add(LSTM(32, activation = 'relu', return_sequences=False,
              kernel_regularizer=regularizers.l1(0.001)))
model.add(Dropout(0.5))
model.add(Dense(32, activation = 'relu', kernel_regularizer=regularizers.l1(0.001)))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(optimizer= optimizers.Adam(), loss = 'binary_crossentropy',
              metrics = ['acc', precision, recall])

#display(model.summary())
history = model.fit(X_train_padded, y_train, batch_size = 32, epochs = 8,
                  callbacks = callback, validation_split = .1,
                  class_weight = class_weight)
```

MODEL 3: Using News Headlines to better predict the stock market

News and S&P500 Predictions



```
final_model = Sequential()

final_model.add(LSTM(64, activation = 'relu', input_shape = input_shape,
                    return_sequences = True))
#final_model.add(Dropout(0.5))
final_model.add(LSTM(32, activation = 'relu', return_sequences = False))
final_model.add(Dense(1, activation = 'relu'))

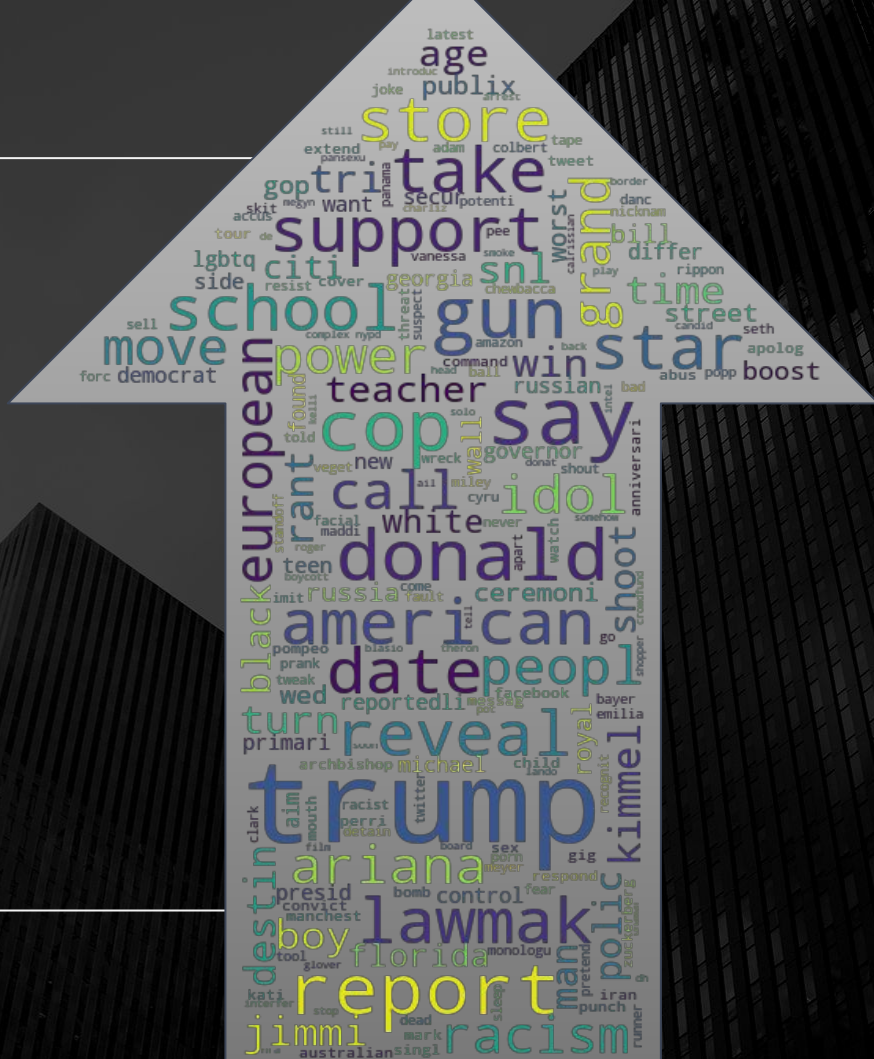
final_model.compile(optimizer = optimizers.Nadam(), loss = 'mse',
                   metrics = ['mse'])

display(final_model.summary())
history = final_model.fit(generator, epochs = 20)
```

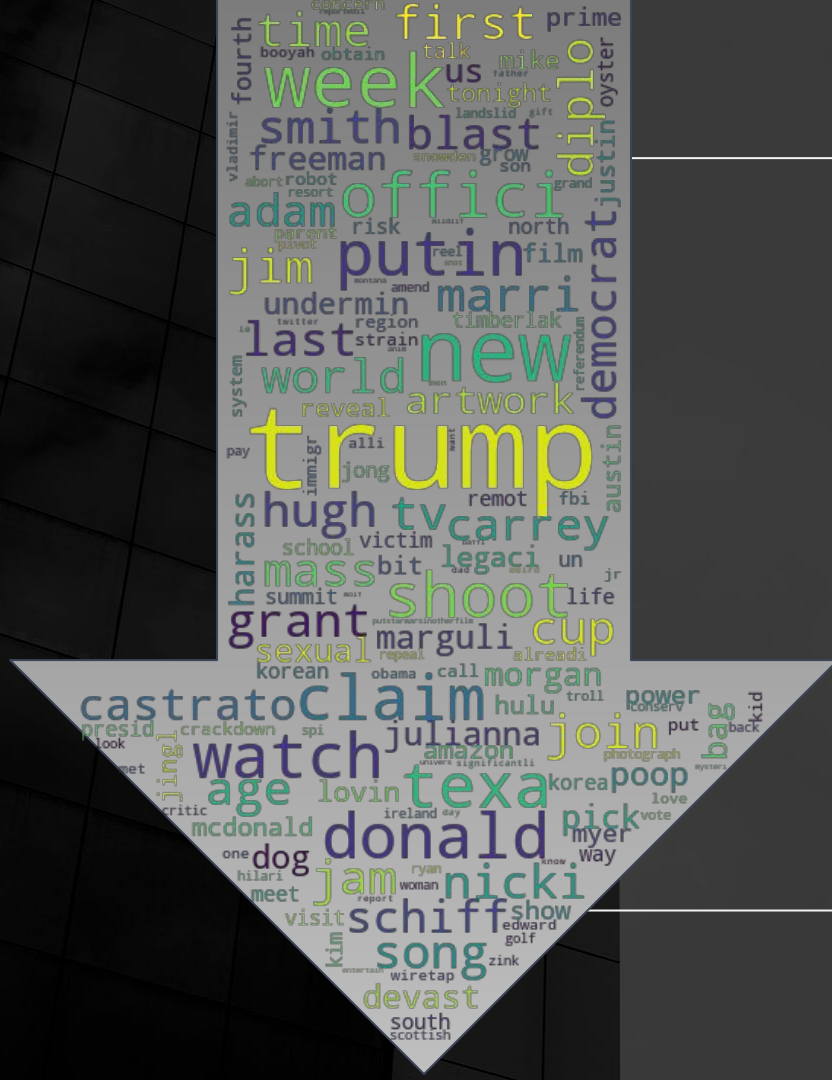
Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 50, 64)	17152
lstm_4 (LSTM)	(None, 32)	12416
dense_3 (Dense)	(None, 1)	33
Total params: 29,601		
Trainable params: 29,601		
Non-trainable params: 0		

**Invest when
positive
words are
present in
news
headlines.**

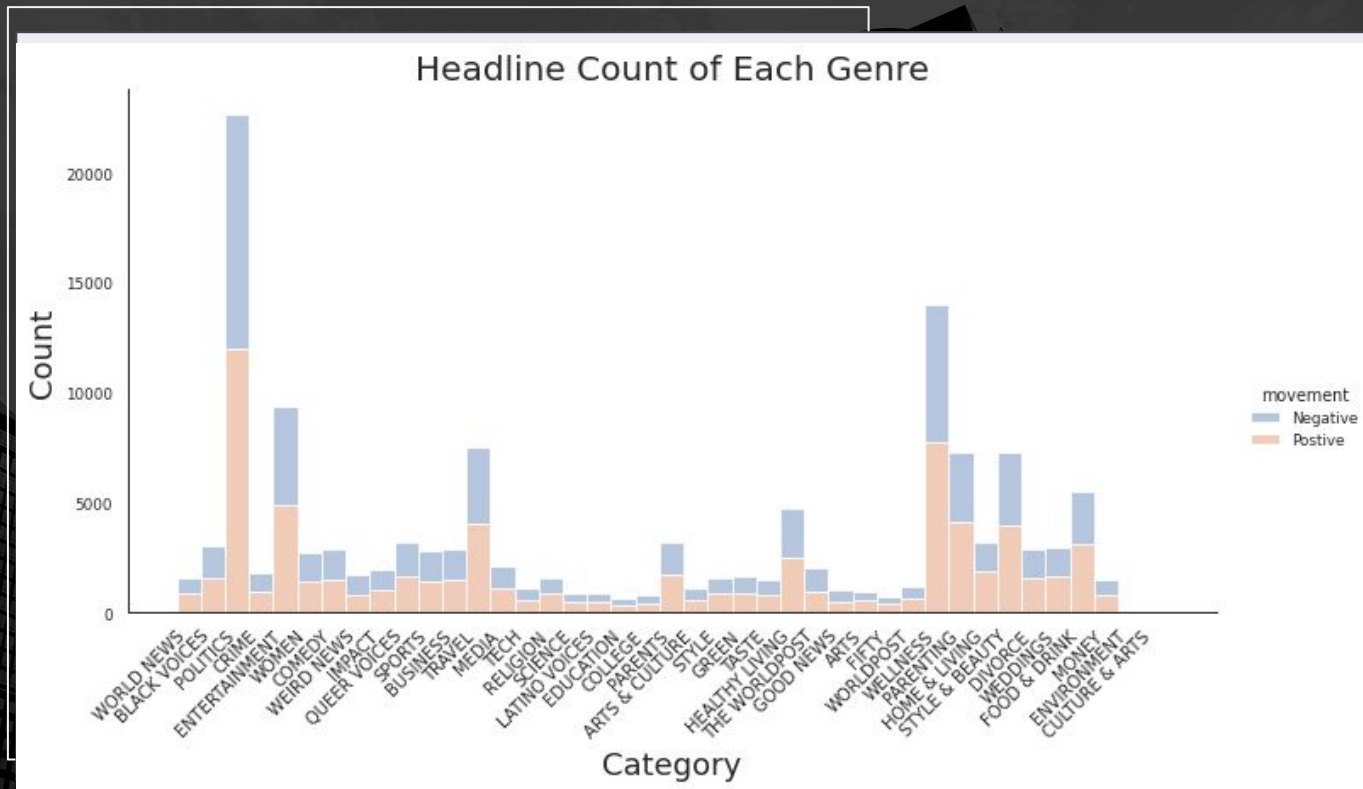


02



03

USE
SPECIFIC
GENRE
HEADLINES



- 1. Build a simple Time Series Model**
- 2. Create a NLP Model to see what words cause the stock to increase or decrease**
- 3. Build a Time Series Model to see which news headlines better predict the market.**

**FOLLOW
SAME PATH**

04



OUR RECOMMENDATIONS

1. Invest in the S&P 500 when positive words are present in the news headlines.
2. Wait to invest when the negative words are present in news headlines.
3. Use specific genre headlines
4. Follow same path when creating your models.

FUTURE WORK

- 1. Separate News Headlines by Category to see which Category impacts the S&P 500 more.**
- 2. Test different Headline Sources (ex. Wall Street Journal, New York Times, ESPN etc.) to see if one news source has a greater impact than others.**
- 3. Test if categorical papers impact categorical stocks. Ex. sports headlines impacting sports company stocks.**
- 4. Try different model types. Ex. PDArima model for initial time series model.**

Thank you for your time and consideration!

QUESTIONS?

Feel free to e-mail me at
Lauren.Esser02@gmail.com or reach
out via LinkedIn

APPENDIX A1:

Neural Network

Stocks Time Series

```
#define model
model = Sequential()
model.add(LSTM(units = 64, activation = 'relu', input_shape = input_shape))
model.add(Dense(1))

model.compile(optimizer= optimizers.Nadam(), loss = 'mse', metrics = ['mse'])

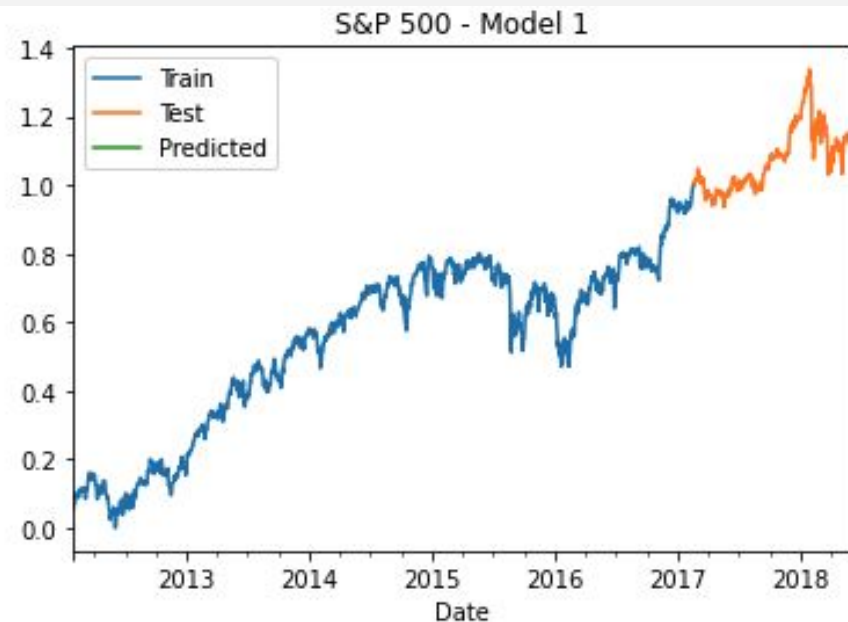
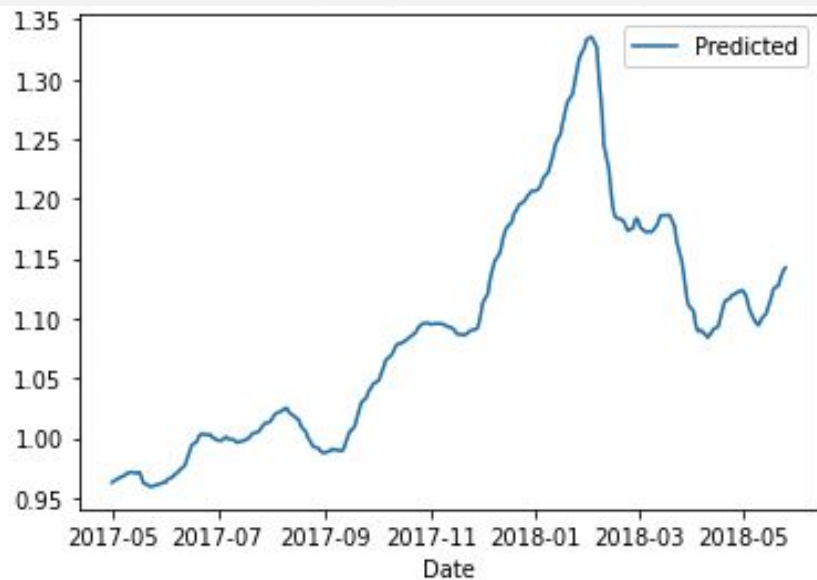
display(model.summary())
history = model.fit(generator, epochs = 20)
```

Model: "sequential"

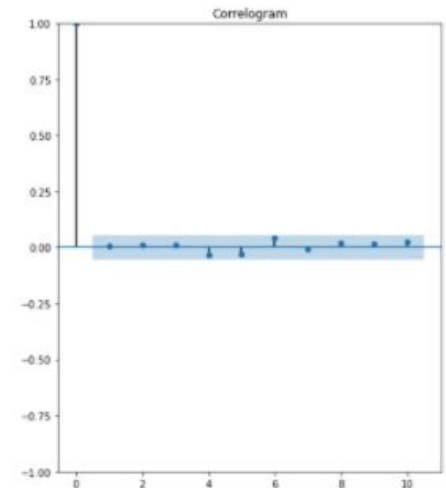
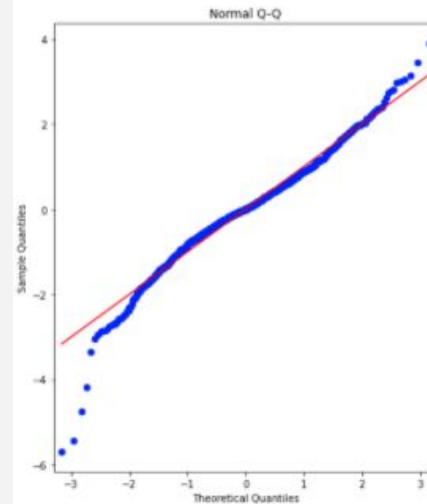
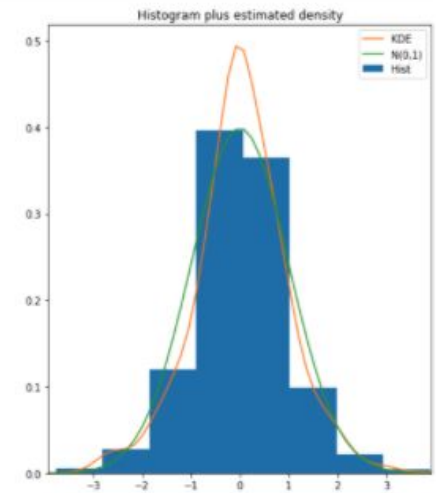
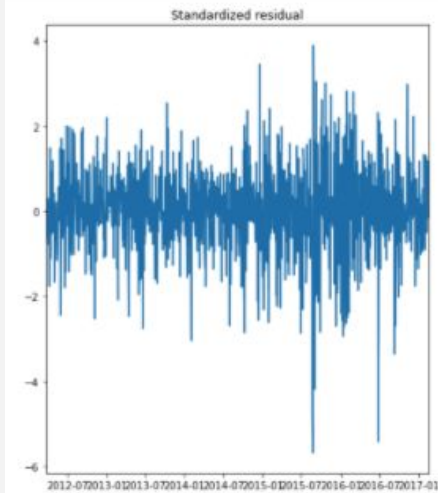
Layer (type)	Output Shape	Param #
=====		
lstm (LSTM)	(None, 64)	16896
dense (Dense)	(None, 1)	65
=====		

Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0

APPENDIX A2:

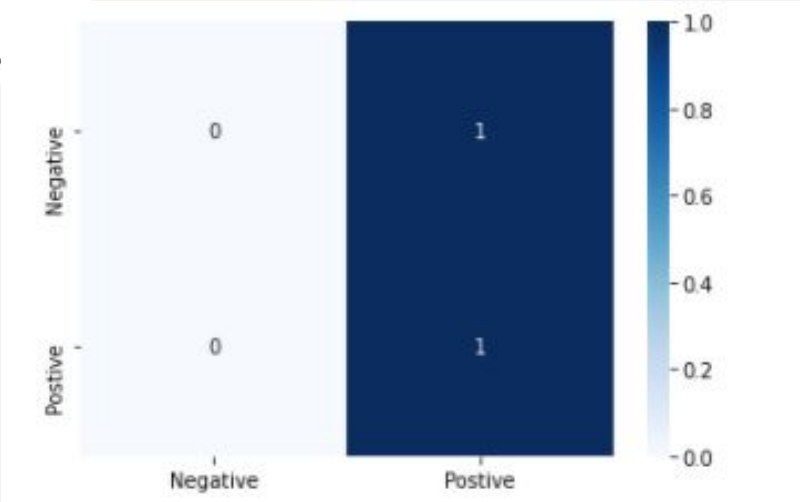


Appendix A3: SARMINA MODEL CHARTS

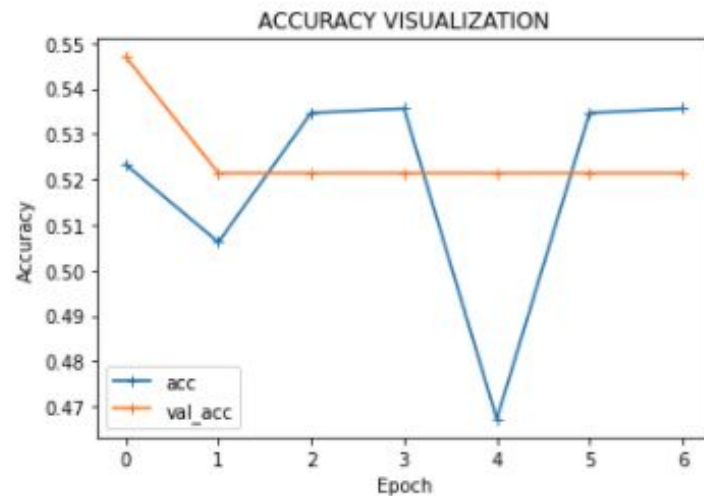
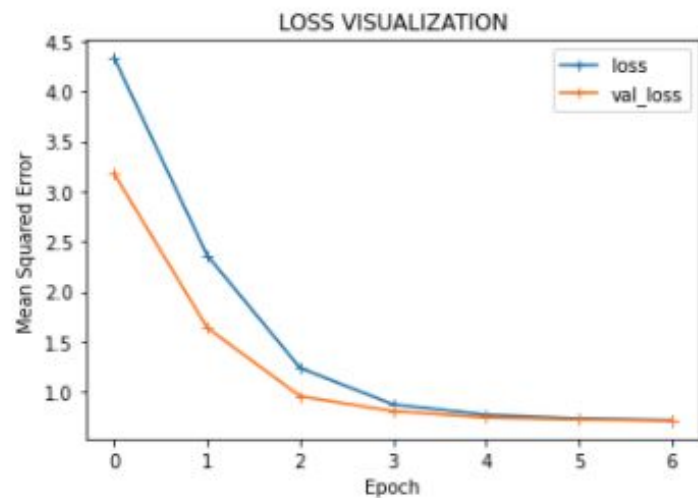
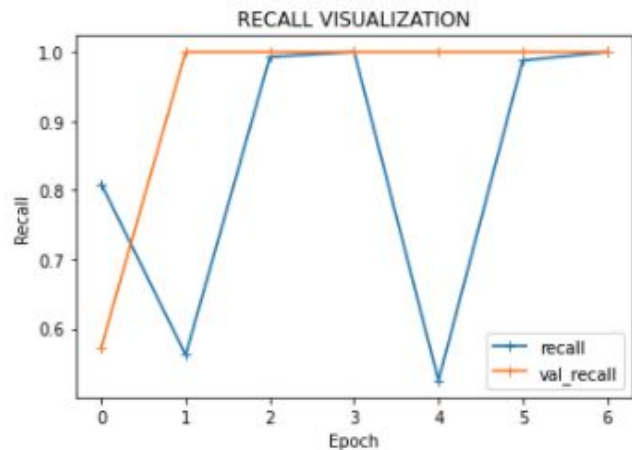
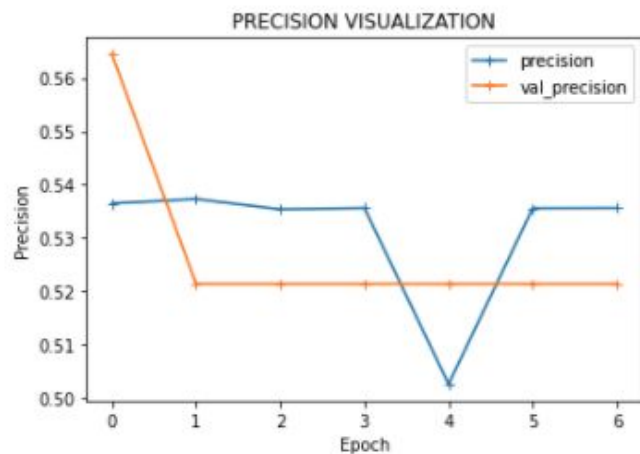


APPENDIX B1

CLASSIFICATION REPORT				
	precision	recall	f1-score	support
Negative	0.00	0.00	0.00	136
Postive	0.53	1.00	0.70	156
accuracy			0.53	292
macro avg	0.27	0.50	0.35	292
weighted avg	0.29	0.53	0.37	292



APPENDIX B2



APPENDIX C: RANDOM FOREST FOR MODEL 2

```
clf = RandomForestClassifier(random_state = 1212, criterion= 'gini',
                             max_depth= 1, min_samples_leaf= 1,
                             min_samples_split= 2 )
clf.fit(X_train_tf, y_train.ravel())

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=1, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=100,
                        n_jobs=None, oob_score=False, random_state=1212,
                        verbose=0, warm_start=False)
```

Testing Accuracy for Classifier: 53.42%

CLASSIFICATION REPORT

	precision	recall	f1-score	support
Negative	0.00	0.00	0.00	136
Positive	0.53	1.00	0.70	156
accuracy			0.53	292
macro avg	0.27	0.50	0.35	292
weighted avg	0.29	0.53	0.37	292