Better Predicting the Stock Market

By: Lauren Esser

PROBLEM STATEMENT

Can we use News Headlines to better predict when stocks will rise?

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

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&P500 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

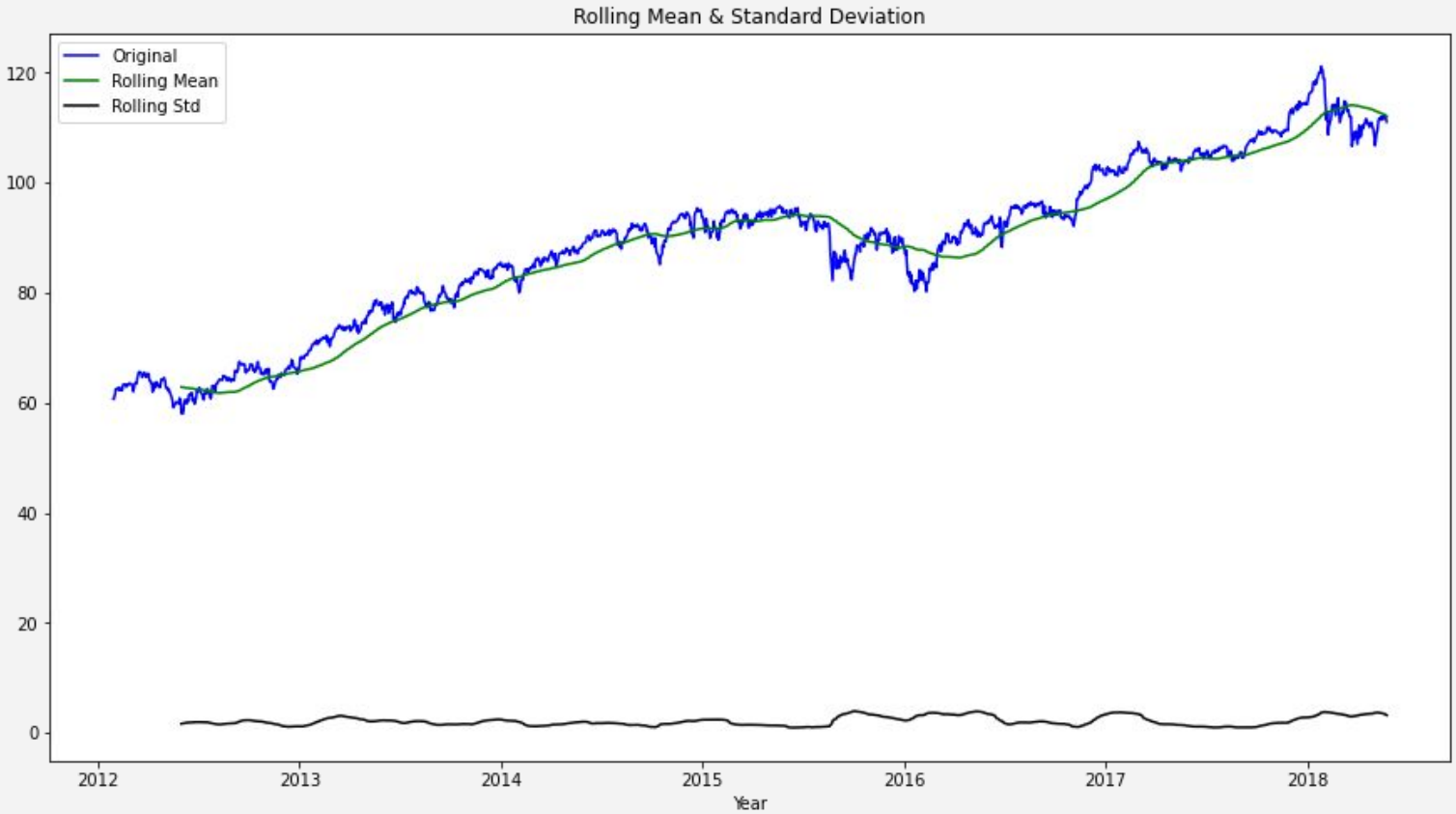
OBTAIN - Data was obtained on Huffington Post News Headlines and the S&P 500

THE HUFFINGTON POST

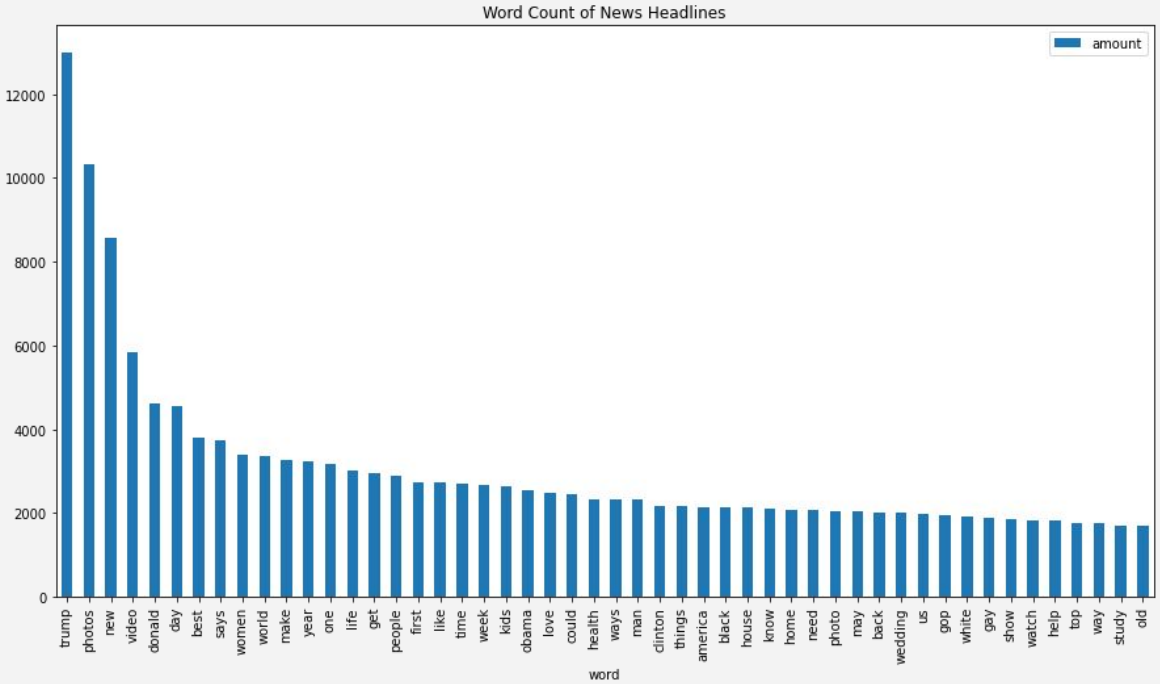
STANDARD &POOR'S 500

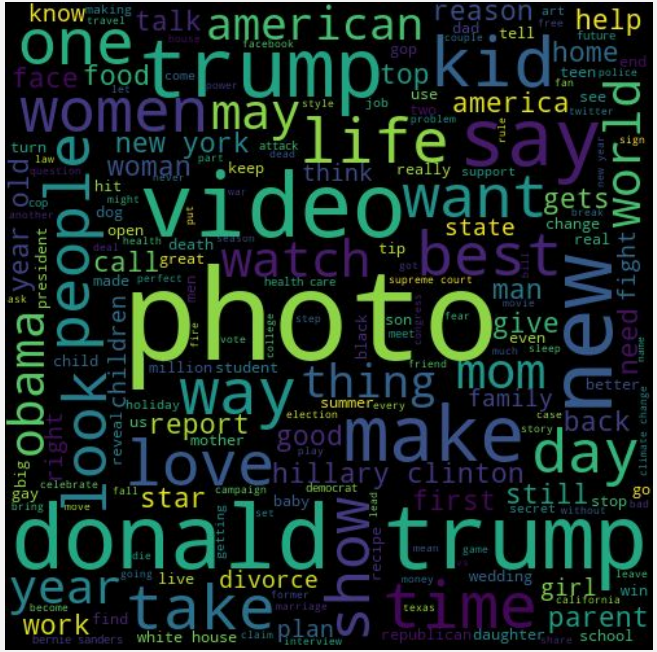
SCRUB - Index converted to datetime, checked for nulls, and identified stop words, punctuation, and tokenizing in news dataset.

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.

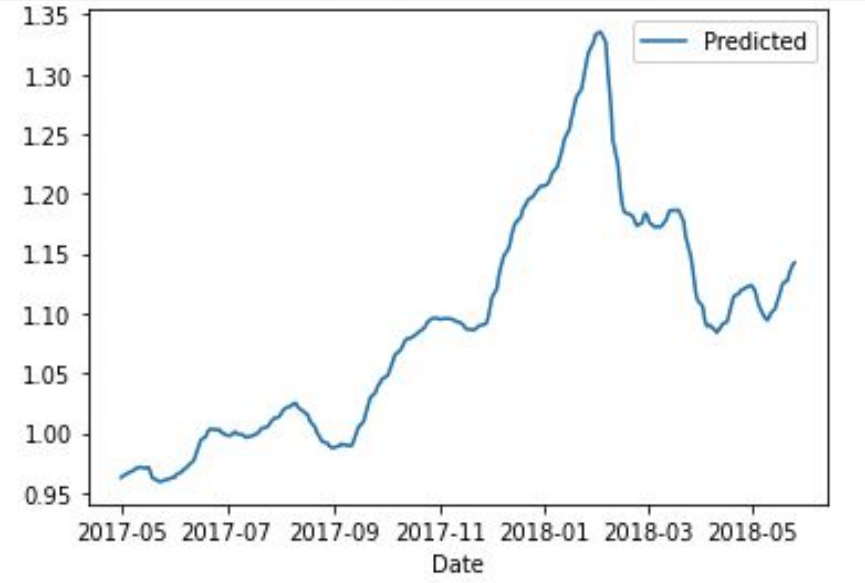


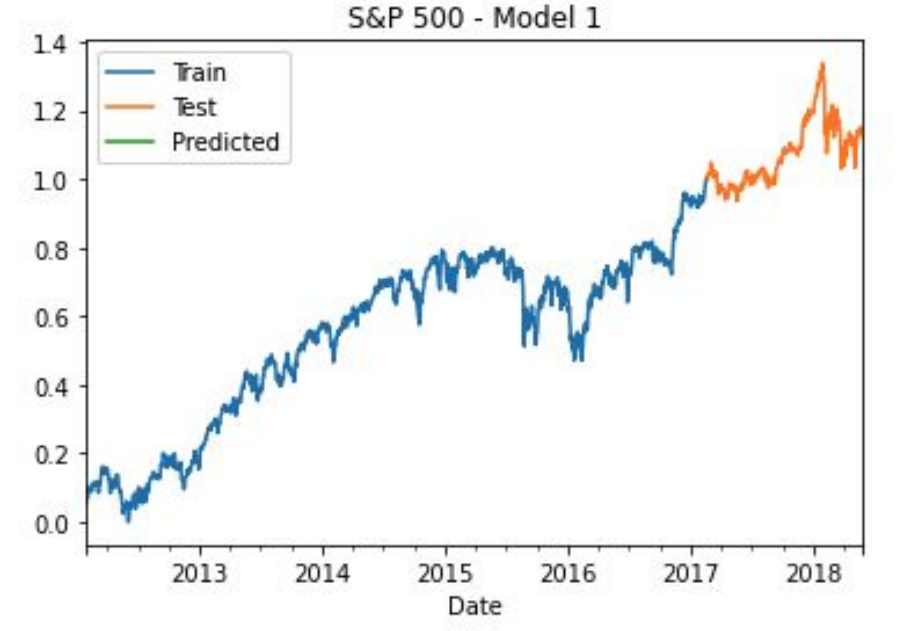


MODEL 1:
Stocks Time Series

```
#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
```


Predicted
1.35
1.30
1.25
1.20
1.15
1.10
1.05
1.00
0.95
2017-05
2017-07
2017-09
2017-11
2018-01
2018-03
2018-05
Date


S&P 500 - Model 1
Train
Test
Predicted
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2013
2014
2015
2016
2017
2018
Date

MODEL 2: NLP USING STOCK DATA

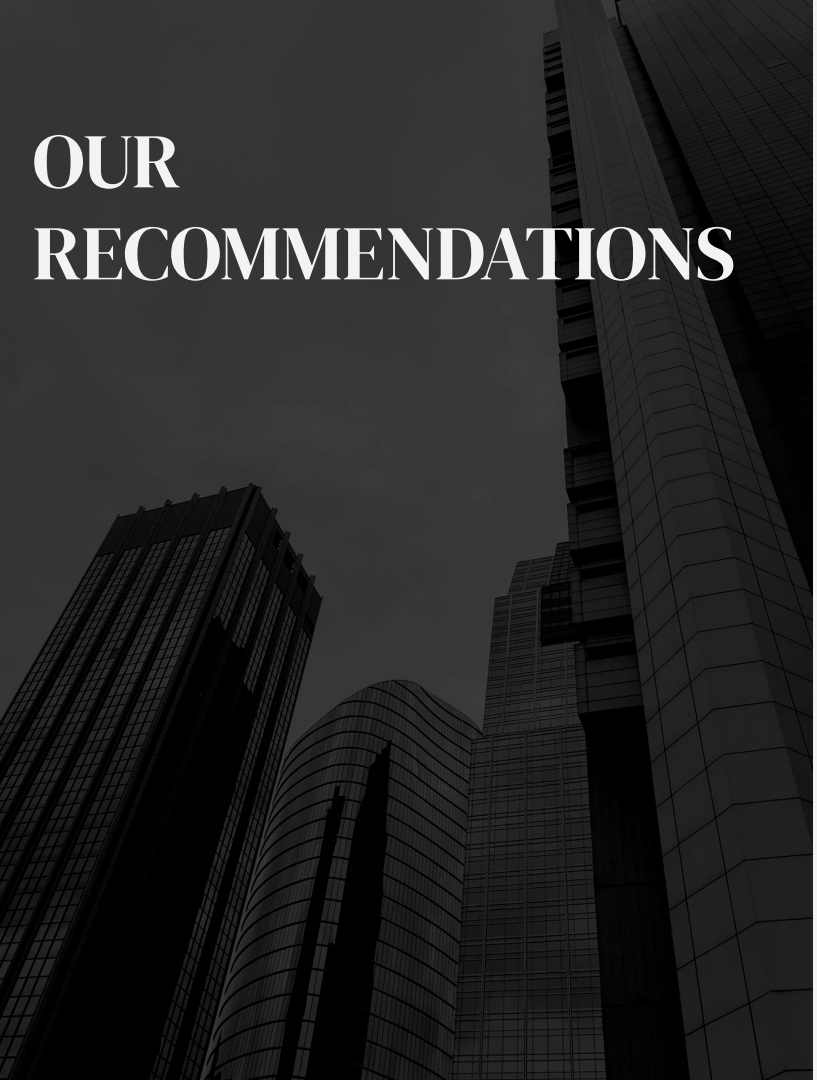
```
model = Sequential()
model.add(Embedding(max_words, 300)) #can change 100 for how many datapts
model.add(LSTM(64, activation = 'relu', return_sequences=True,
               kernel_regularizer=regularizers.l1(0.001)))

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)
```

53% accuracy: visualizations in Appendix A

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


```
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
_________________________________________________________________
```

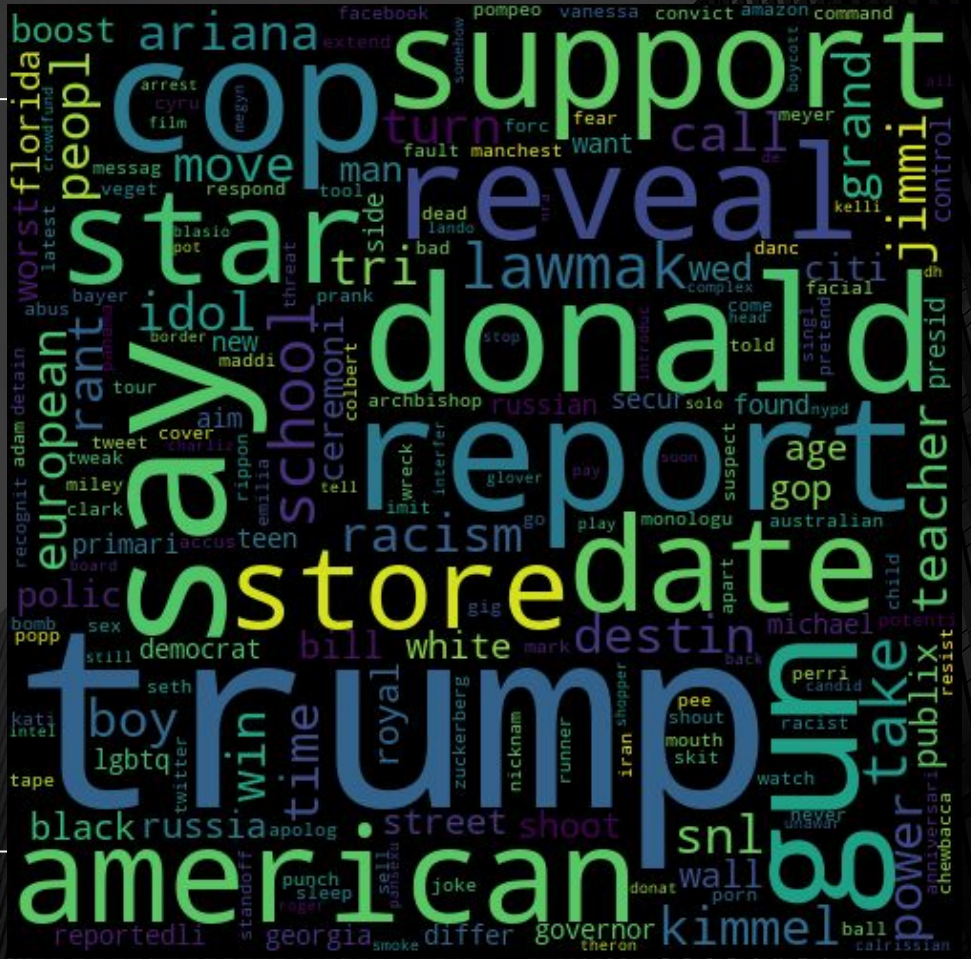


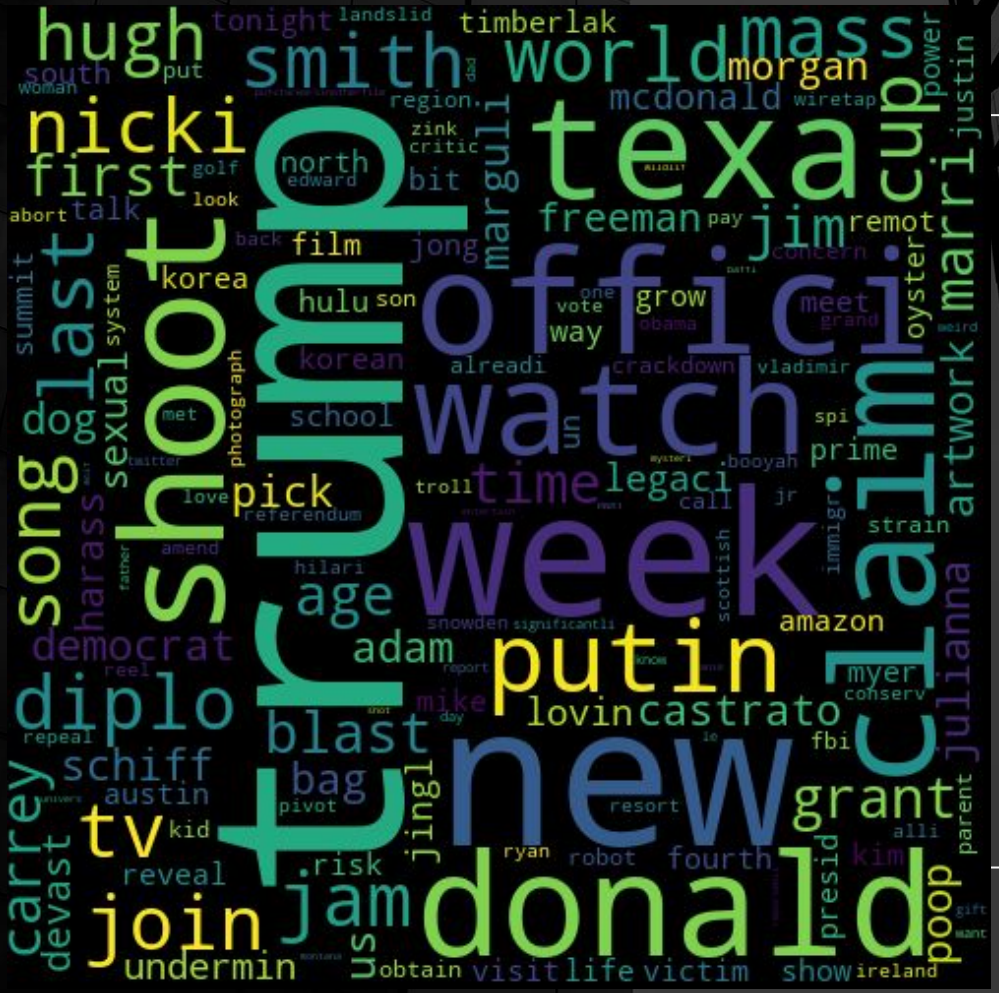
OUR RECOMMENDATIONS

1. Look at positive word list
2. Avoid words on negative list
3. Make your model more specific
4. Follow same path

01

POSITIVE WORDS LIST



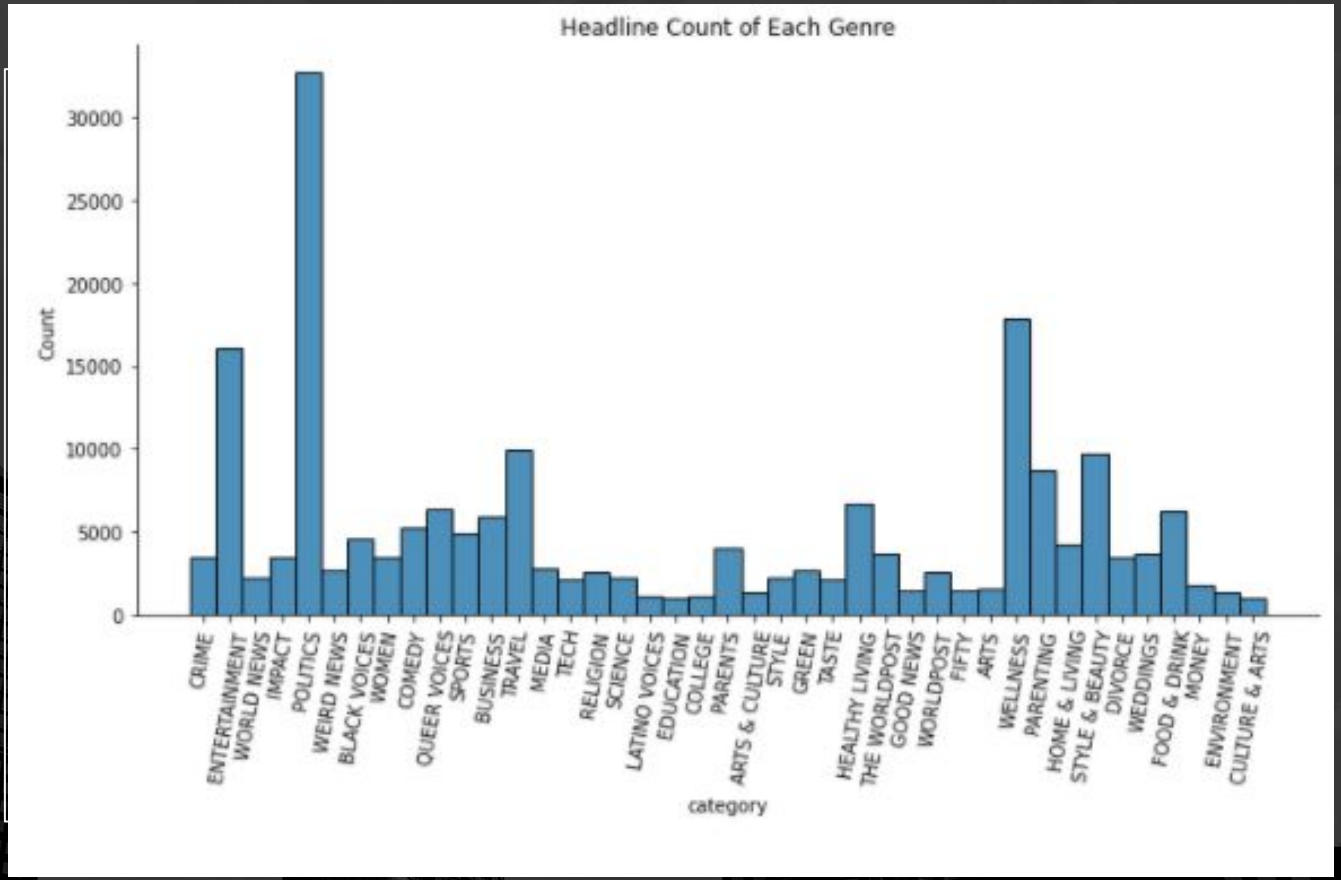


NEGATIVE WORDS LIST

02

03

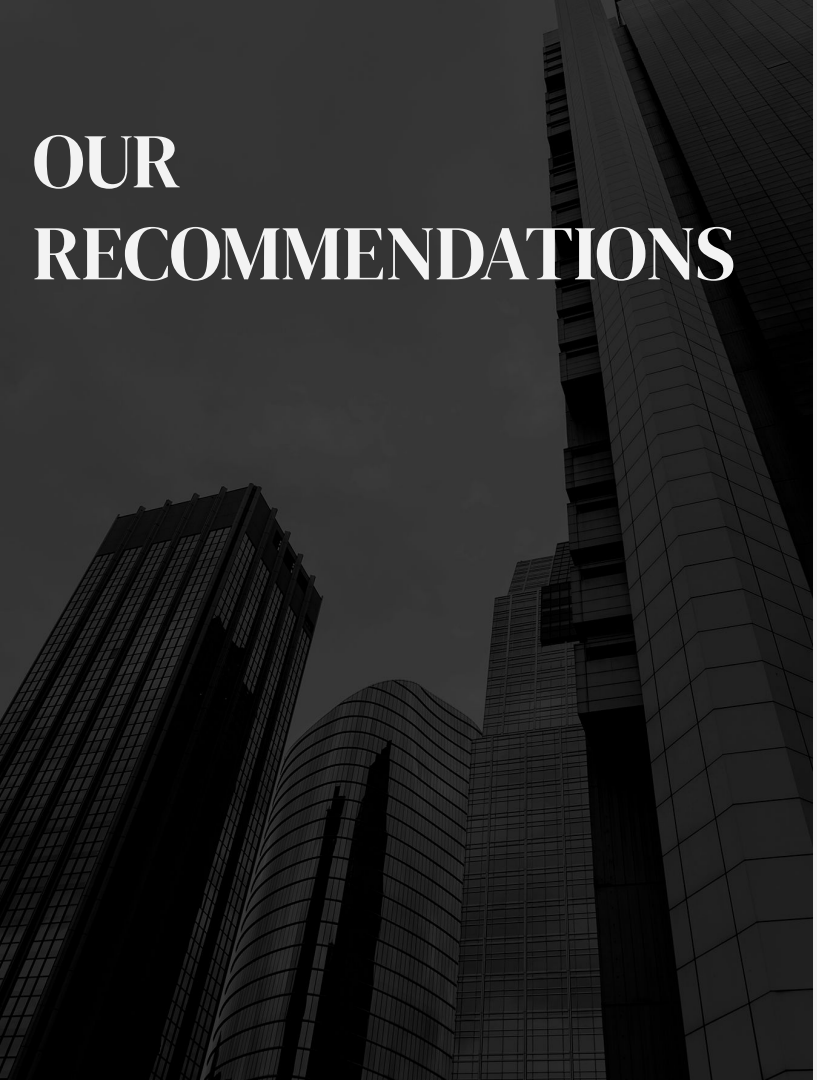
SAME MODEL, MORE SPECIFIC HEADLINES



1. Build a simple Time Series Model
   a. We recommend using a Neural Network
2. Create a NLP Model to see what words cause the stock to increase or decrease
   a. We recommend a LSTM Neural Network or Random Forest
3. Build a Time Series Model to see which news headlines better predict the market.

FOLLOW SAME PATH

04



OUR RECOMMENDATIONS

1. Look at positive word list
2. Avoid words on negative list
3. Make your model more specific
4. Follow same path

FUTURE WORK

1. Separate News Headlines by Category to see which Category impacts the stock market more.
2. Test different Newspapers (ex. Wall Street Journal, New York Times, 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.

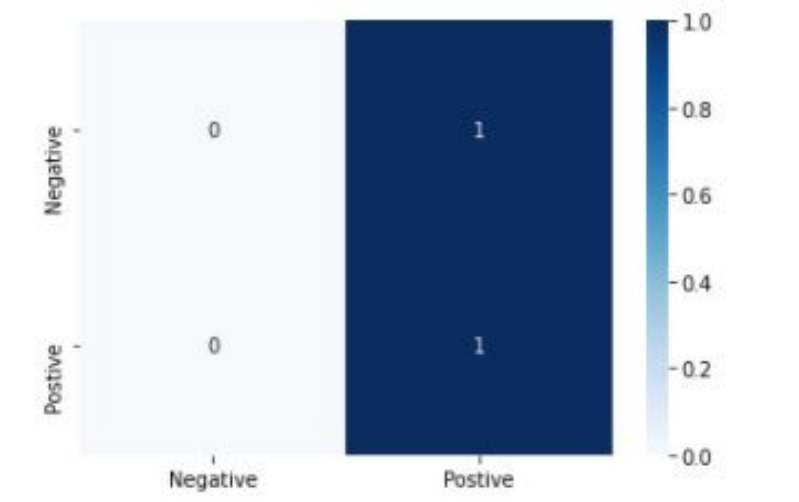
Thank you for your time and consideration!

QUESTIONS?

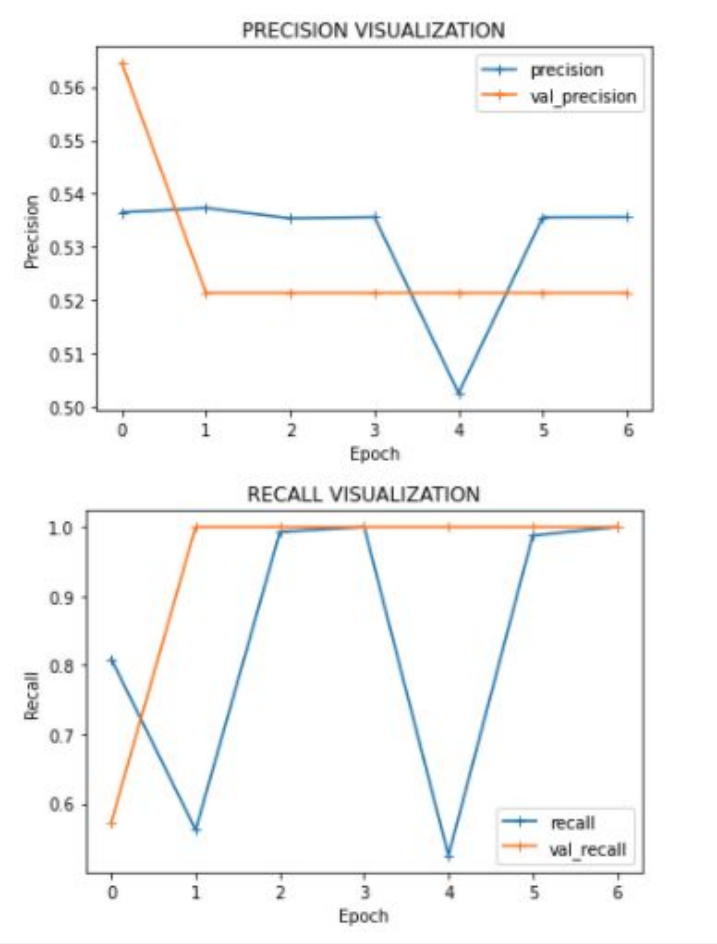
Feel free to e-mail me at Lauren.Esser02@gmail.com or reach out via LinkedIn

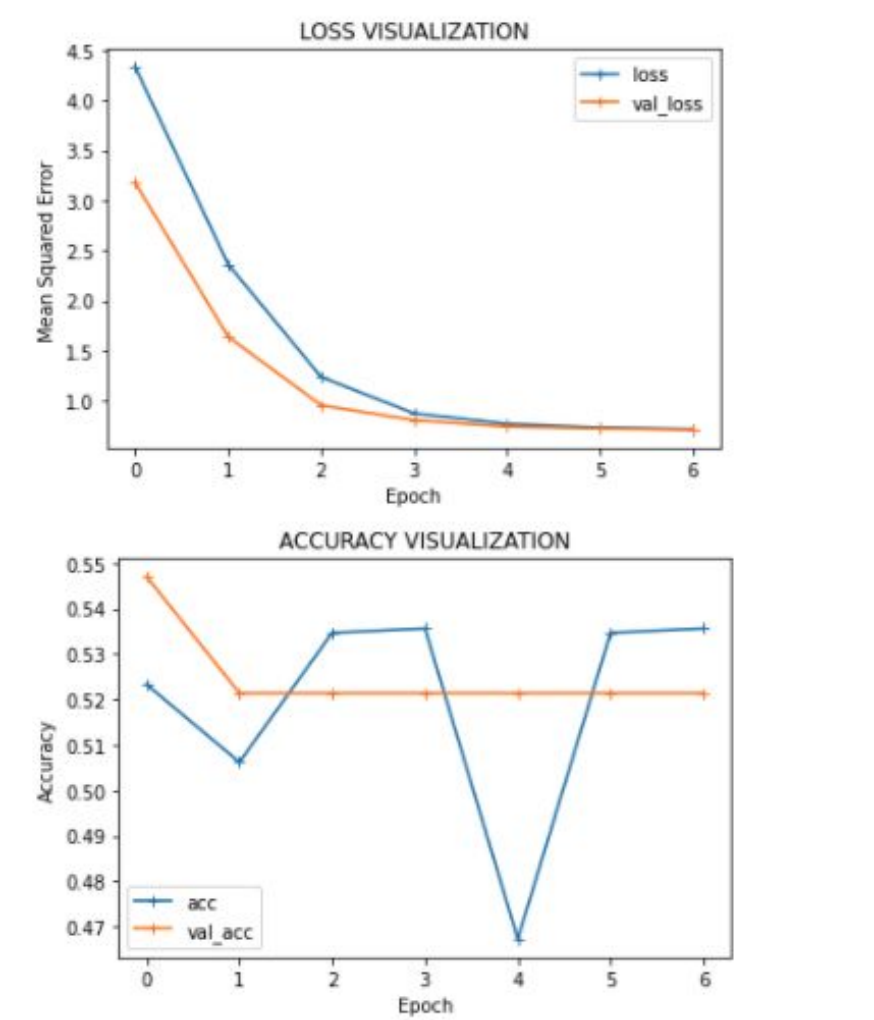
APPENDIX A1

```
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 A2


PRECISION VISUALIZATION
precision
val_precision
Precision
Epoch
RECALL VISUALIZATION
recall
val_recall
Recall
Epoch


LOSS VISUALIZATION
loss
val_loss
Mean Squared Error
Epoch
ACCURACY VISUALIZATION
acc
val_acc
Accuracy
Epoch

APPENDIX B: 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

----------------------------------------------------------------
```