

Project Report

Group 8: Benjamin Wright, Jasmine Jensen, Nyla Hussain, Anmol Matharu

Introduction:

With this project we wanted to determine whether we could predict the direction of the stock market based on news articles. To do this, we vectorized a set of articles using the TD-IDF vectorization model which vectorizes based on the words and lowers the weights of those that are recurring (this can help prevent the weights being affected by filler words).

The S&P 500 Stock Data dataset (<https://www.kaggle.com/datasets/camnugent/sandp500/data>) stores the historical stock prices for companies found on the S&P 500 index from 2013 to 2018. It includes the date, the highest price reached in a day, the lowest price reached in a day, the price of the stock at market open, the number of shares traded, and the stock ticker's name.

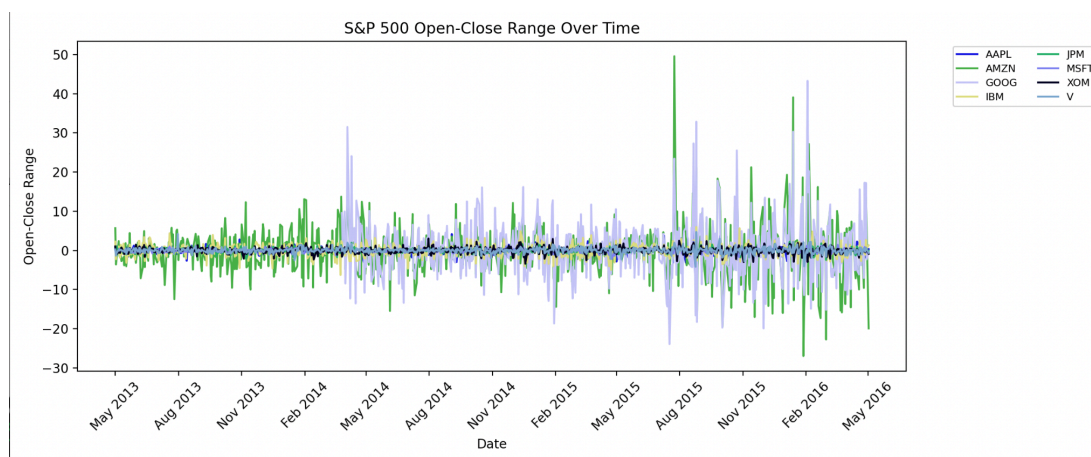
The Daily News for Stock Market Prediction dataset (<https://www.kaggle.com/datasets/aaron7sun/stocknews>) stores the date, whether the DJIA rose or dropped that day, and the top 25 news articles from /r/worldnews. It includes data from 2008-08-08 to 2016-06-30

Preliminary Analysis:

<pre>ticker IBM high low range statistics mean: 2.39 median: 2.14 variance: 0.95 standard deviation: 0.98 open close range statistics mean: -0.03 median: -0.03 variance: 2.36 standard deviation: 1.54</pre>	<pre>ticker AMZN high low range statistics mean: 9.3 median: 7.61 variance: 32.03 standard deviation: 5.66 open close range statistics mean: -0.0 median: 0.0 variance: 43.03 standard deviation: 6.56</pre>	<pre>ticker GOOG high low range statistics mean: 11.33 median: 9.79 variance: 40.67 standard deviation: 6.38 open close range statistics mean: 0.39 median: 0.25 variance: 58.31 standard deviation: 7.64</pre>
---	--	---

<p>ticker V</p> <p>high low range statistics</p> <p>mean: 1.03 median: 0.88 variance: 0.5 standard deviation: 0.71</p> <p>open close range statistics</p> <p>mean: -0.0 median: -0.01 variance: 0.52 standard deviation: 0.72</p>	<p>ticker XOM</p> <p>high low range statistics</p> <p>mean: 1.32 median: 1.17 variance: 0.39 standard deviation: 0.62</p> <p>open close range statistics</p> <p>mean: -0.03 median: -0.01 variance: 0.71 standard deviation: 0.84</p>	<p>ticker JPM</p> <p>high low range statistics</p> <p>mean: 0.97 median: 0.86 variance: 0.37 standard deviation: 0.61</p> <p>open close range statistics</p> <p>mean: -0.0 median: -0.04 variance: 0.38 standard deviation: 0.62</p>
---	---	--

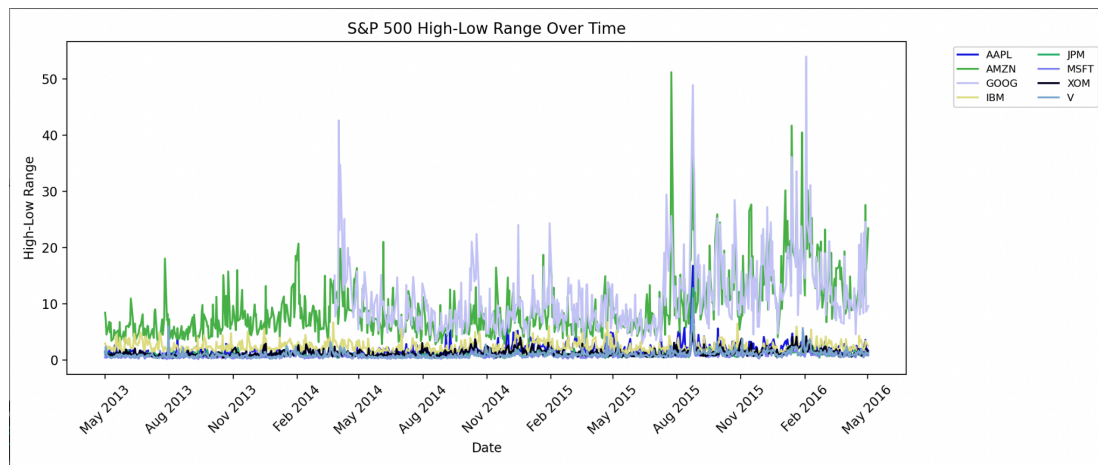
We ended up graphing and displaying the summary statistics for the high low range and the open close range for six popular companies across different industries. The mean represents the average change in price for a single day between the two values, the median is the middle value (doesn't take into account outliers), variance represents the proximity of values in comparison to the mean, and the standard deviation is the variance square rooted to get a more applicable value for the proximity of the values.



The Open-Close Range Over Time analyzes the data using the calculations for the range of the price of the stock at open and close (open - close) for the companies Apple, Amazon, Google, IBM, JPMorgan, Microsoft, and Exxon from May 2013 to May 2016. Not all companies were graphed to avoid a graph that is too full to gather meaningful information from. From the figure above, you can see the range tends to be more drastic after May 2015. In the table below, you can see the maximum and minimum ranges for four companies that seemed to have the largest differences; Amazon, Google, Apple, and IBM. It is interesting that Apple and IBM have the same date where they have the maximum open/close range; August 25, 2015; and two of the top 25 news headlines on that day discuss the stock market; "Shanghai stock market opens down 6.1%" and "The Richest Guy in Asia Loses \$3.6 Billion in the Market Rout." Since it is the maximum, it means that the close date was much lower than the open date, which aligns with the news stating that one of the largest stock markets was down. It is also intriguing that Apple's

minimum open/close date range is the day before its maximum; August 24, 2015 and August 25, 2015 respectively. There is also one headline discussing the stock market on that day which reads “The Australian Stock Market has plunged 2.5% in the first 20 minutes of opening for the week, losing \$20 billion. The Chinese Stock Market has dropped 7% just an hour after opening, and the ripple effects are starting to be seen worldwide as markets open.” This does not completely make sense with our data, because on the day that our stock goes up most from open to close, the stock market in different areas of the world dropped. None of the other dates’ news headlines directly discussed the stock market, but some have discussed money and the economy of mostly Russia and India.

Company	Maximum Open/Close Range	Maximum Date	Minimum Open/Close Range	Minimum Date
Amazon	49.57	07-24-2015	-26.98	01-28-2016
Google	43.57	02-03-2016	-23.93	07-17-2015
Apple	7.37	08-25-2015	-8.25	08-24-2015
IBM	5.98	08-25-2015	-6.28	03-25-2014



Another graph, High-Low Range Over Time, is used to analyze the data using the calculations for the range of the highest price reached in the day and the lowest price reached in the day (high - low) for the same companies in the open-close graph. In the figure above, there is a spike for all companies right after August 2015. Amazon and Google also tend to move together, while most of the other companies seem to be more stable below them. In the table below, you can see the maximum and minimum ranges for four companies that seemed to have the largest differences; Amazon, Google, Apple, and IBM. Apple’s maximum range between the highest price and the lowest price falls on August 24, 2015, which is the same day that Apple had

its minimum open-close date. This is interesting because that means Apple had its largest stock price on the same day that its price had increased most from open to close. That is the day in which the top news headlines reads “The Australian Stock Market has plunged 2.5% in the first 20 minutes of opening for the week, losing \$20 billion. The Chinese Stock Market has dropped 7% just an hour after opening, and the ripple effects are starting to be seen worldwide as markets open”, showing the stock markets around the world had dropped. Amazon and Google were also interesting, since their maximum high-low range date was on the same date as the maximum open-close range date; July 24, 2015 and February 3, 2016 respectively. This means that on the date where they had their highest stock price, they also had the most decrease in stock market price from open to close. Both of these dates’ top news headlines do not discuss the stock market.

Company	Maximum High/Low Range	Maximum Date	Minimum High/Low Range	Minimum Date
Amazon	51.22	07-24-2015	2.80	08-19-2014
Google	54	02-03-2016	2.67	03-20-2015
Apple	16.8	08-24-2015	0.46	04-02-2014
IBM	6.77	04-23-2015	0.77	12-24-2015

Models:

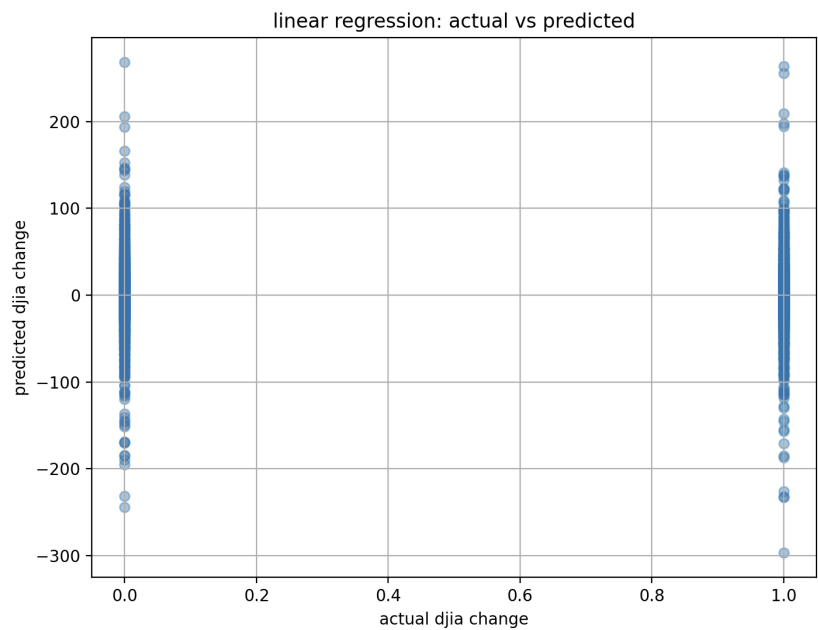
Linear Regression Model

We used the linear regression model as a baseline model to see whether there was a linear relationship between the daily news headlines and the DJIA. This regression provided us with a clear foundation for whether the texts had any predictive signals regarding the market. We trained the model on TF-IDF vectorized news headlines that encode word frequency and importance, but it does not really capture any contextual meaning.

When we ran the model, we could see the limitations of linear regression very quickly. The training R^2 of 0.998 shows us that there is extreme overfitting, as this model was just memorizing the training data, which is to be expected. The classification accuracy with the test data shows that the model is performing the same as essentially random guessing, with an accuracy of 50%. Finally, the testing R^2 was a negative value, indicating that the linear regression performed worse than a dummy predictor. Overall, Linear Regression is extremely unreliable and unstable for this type of data set.

We can also see these in the scatter plot, as there is no visible trend or relationship. There are just vertical bands with lots of noise. Linear regression does not work for this text prediction because the headlines have a sparse feature space that makes the coefficients both unstable and

overfitted. The lack of context and complexity within the data also makes it hard to predict market behaviour.



```
linear regression results
training r^2: 0.9978023877237148
testing r^2: -4453.355161504077

training classification report:
              precision    recall  f1-score   support

     0           1.00       1.00       1.00       6653
     1           1.00       1.00       1.00       7540

 accuracy              1.00       1.00       1.00      14193
 macro avg              1.00       1.00       1.00      14193
weighted avg              1.00       1.00       1.00      14193

training confusion matrix:
[[6652   1]
 [  2 7538]]
```

```

testing classification report:
              precision    recall  f1-score   support

         0       0.47      0.49      0.48       2222
         1       0.53      0.50      0.52       2510

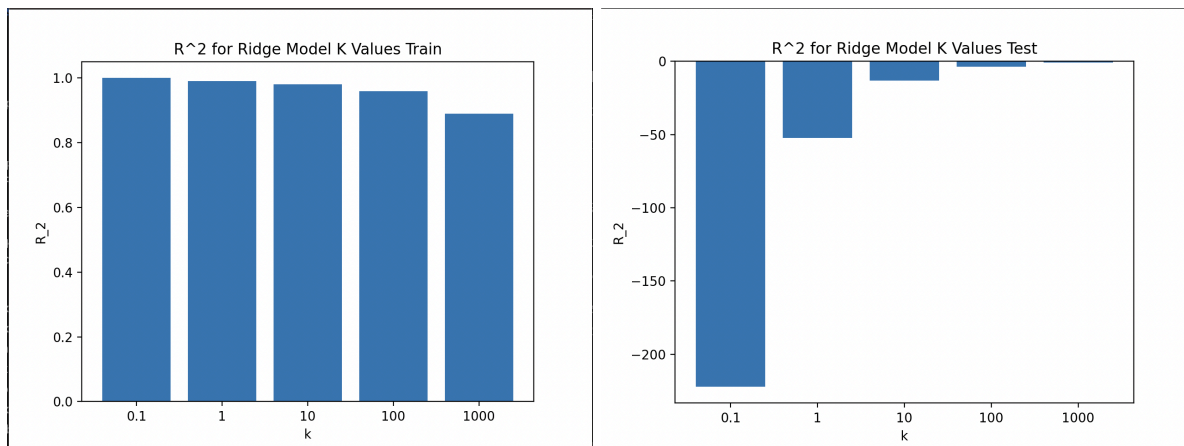
 accuracy      0.50
 macro avg     0.50      0.50      0.50
weighted avg     0.50      0.50      0.50

testing confusion matrix:
[[1088 1134]
 [1243 1267]]

```

Ridge Regression Model

We chose to use the ridge regression model as a comparison to the linear regression model. We wanted to show that the ridge regression model would not be a good model for this data set due to the fact that it is best used when there are more factors than datapoints in the dataset. This is because it has an alpha value (k values in this case) that acts as a penalty to the coefficients. As the alpha value rises, it leads to a stabilization of the model due to driving the coefficients towards zero. This can be seen in the R^2 values going towards zero in the testing data as the k value increases.



The R^2 value shows a good prediction rating for the training data, but in the testing data it performs very poorly. The R^2 being negative shows that it performs worse than if you were to guess based on the percentage of ones or zeros in the original dataset and as it reached zero, due to the stabilization, the model performed slightly worse than if you guessed based on the percentage of ones and zeros.

Model with k = 0.1

Classification Report

	precision	recall	f1-score	support
0	0.47	0.46	0.46	2222
1	0.53	0.54	0.53	2510
accuracy			0.50	4732
macro avg	0.50	0.50	0.50	4732
weighted avg	0.50	0.50	0.50	4732

Confusion Matrix Report

[[1015 1207]
[1159 1351]]

Model with k = 1

Classification Report

	precision	recall	f1-score	support
0	0.47	0.42	0.44	2222
1	0.53	0.59	0.56	2510
accuracy			0.51	4732
macro avg	0.50	0.50	0.50	4732
weighted avg	0.51	0.51	0.51	4732

Confusion Matrix Report

[[927 1295]
[1028 1482]]

Model with k = 10

Classification Report

	precision	recall	f1-score	support
0	0.47	0.36	0.41	2222
1	0.53	0.64	0.58	2510
accuracy			0.51	4732
macro avg	0.50	0.50	0.49	4732
weighted avg	0.50	0.51	0.50	4732

Confusion Matrix Report

[[797 1425]
[901 1609]]

Model with k = 100

Classification Report

	precision	recall	f1-score	support
0	0.49	0.28	0.36	2222
1	0.54	0.75	0.63	2510
accuracy			0.53	4732
macro avg	0.52	0.51	0.49	4732
weighted avg	0.52	0.53	0.50	4732

Confusion Matrix Report

[[626 1596]
[640 1870]]

Model with k = 1000

Classification Report

	precision	recall	f1-score	support
0	0.51	0.15	0.23	2222
1	0.54	0.87	0.67	2510
accuracy			0.53	4732
macro avg	0.53	0.51	0.45	4732
weighted avg	0.53	0.53	0.46	4732

Confusion Matrix Report

[[332 1890]
[314 2196]]

The classification report essentially shows us that the best accuracy given from all of the k values of the ridge model was at a k value of 1000. This is most likely due to the

overstabilization of the coefficients. The most accurate for this model is 53 percent and therefore is not a good model to predict the stock market based on.

Linear Support Vector Classifier

The model we used that was not discussed in class is the Linear Support Vector Classifier model. This model works by finding the best line that separates the two classes with the most margin, or distance, between the classes. Its goal is to make all of the data points fall on the correct side of the margin. The Scikit-learn's LinearSVC model was used to model the data. At first, the model was very overfitted to the training data, as seen in the figure below.

```
Training Confusion Matrix & Classification Report:
[[5930 723]
 [ 507 7033]]
      precision    recall  f1-score   support

     0       0.92      0.89      0.91      6653
     1       0.91      0.93      0.92      7540

 accuracy      0.91      0.91      0.91      14193
 macro avg     0.91      0.91      0.91      14193
weighted avg     0.91      0.91      0.91      14193


Testing Classification Report:
[[ 991 1231]
 [1054 1456]]
      precision    recall  f1-score   support

     0       0.48      0.45      0.46      2222
     1       0.54      0.58      0.56      2510

 accuracy      0.52      0.52      0.52      4732
 macro avg     0.51      0.51      0.51      4732
weighted avg     0.51      0.52      0.52      4732
```

To reduce overfitting, I added 2 parameters to the model: $C=0.3$ and $\text{loss}=\text{"hinge."}$ The C parameter reduces overfitting by allowing more regularization, which would in turn allow for more training data to be classified incorrectly, which would then let it generalize new data better. The loss parameter being set to hinge allows the model to be more robust to outliers and generalizes better on high-dimensional text data. After adding these parameters, the model became less overfitted, but still not a great model, as seen in the figure below.

Training Confusion Matrix & Classification Report:

```
[[1638 5015]
 [ 240 7300]]
```

	precision	recall	f1-score	support
0	0.87	0.25	0.38	6653
1	0.59	0.97	0.74	7540
accuracy			0.63	14193
macro avg	0.73	0.61	0.56	14193
weighted avg	0.72	0.63	0.57	14193

Testing Confusion Matrix & Classification Report:

```
[[ 216 2006]
 [ 229 2281]]
```

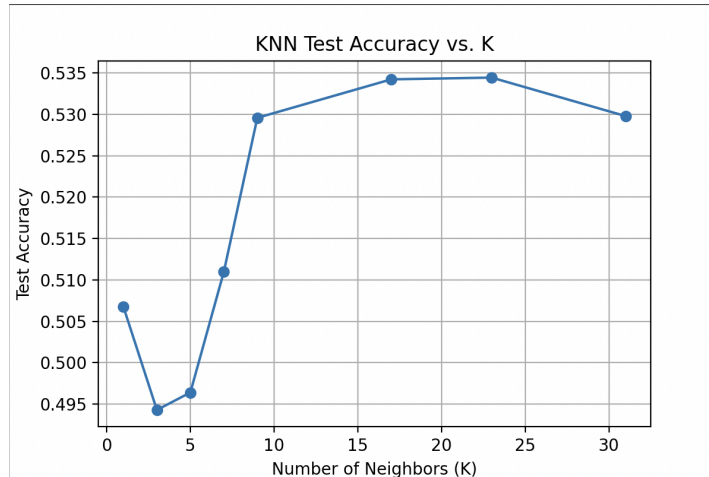
	precision	recall	f1-score	support
0	0.49	0.10	0.16	2222
1	0.53	0.91	0.67	2510
accuracy			0.53	4732
macro avg	0.51	0.50	0.42	4732
weighted avg	0.51	0.53	0.43	4732

This model is much less overfitted, but the accuracy is low on both the testing and training data, 53% and 63% accurate respectively. This model is not a very good model for predicting the data, with the accuracy being only slightly better than a random guess.

K-Nearest Neighbors Classifier

The K-Nearest Neighbors model works by looking at the most similar data points in the past and using them to make a prediction. KNN makes decisions based on distance between neighbors/data points. For our analysis, KNN is used as a classification model to predict whether the DJIA will go up (1) or down (0) using daily news data. On a specific day, KNN finds the k most similar days in the past and looks at how the market was on those particular days. The most common label among those neighbors then becomes the prediction.

As seen in the graph below, the best k value was 23 as it has a test accuracy of about 53.44%. After k = 23, the accuracy begins to decline, potentially showing that too many neighbors can lead to extra noise instead of more accurate predictions. The rest of the figures below depict the training and testing metrics and accuracy for each of the k values I used.



```
K Value: 1
-----
Train Accuracy: 0.9999
Test Accuracy: 0.5068

Train Classification Report:
      precision    recall  f1-score   support

     0       0.9998       0.9998       0.9998        6653
     1       0.9999       0.9999       0.9999        7540

 accuracy          0.9999          0.9999          0.9999       14193
 macro avg          0.9999          0.9999          0.9999       14193
 weighted avg          0.9999          0.9999          0.9999       14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.4813       0.6503       0.5532        2222
     1       0.5509       0.3797       0.4495       2510

 accuracy          0.5161          0.5150          0.5068        4732
 macro avg          0.5161          0.5150          0.5068        4732
 weighted avg          0.5161          0.5150          0.5068        4732

Confusion Matrix (Test):
[[1445  777]
 [1557  953]]
```

```
K Value: 3
-----
Train Accuracy: 0.6927
Test Accuracy: 0.4943

Train Classification Report:
      precision    recall  f1-score   support

     0       0.6145       0.9247       0.7383        6653
     1       0.8802       0.4881       0.6279        7540

 accuracy          0.7473          0.7064          0.6927       14193
 macro avg          0.7473          0.7064          0.6927       14193
 weighted avg          0.7473          0.7064          0.6927       14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.4772       0.8038       0.5988        2222
     1       0.5592       0.2203       0.3161       2510

 accuracy          0.5182          0.5120          0.4943        4732
 macro avg          0.5182          0.5120          0.4943        4732
 weighted avg          0.5182          0.5120          0.4943        4732

Confusion Matrix (Test):
[[1786  436]
 [1957  553]]
```

```
K Value: 5
-----
Train Accuracy: 0.7352
Test Accuracy: 0.4964

Train Classification Report:
      precision    recall  f1-score   support

     0       0.6488       0.9487       0.7706        6653
     1       0.9236       0.5468       0.6869        7540

 accuracy          0.7862          0.7478          0.7352       14193
 macro avg          0.7862          0.7478          0.7352       14193
 weighted avg          0.7862          0.7478          0.7352       14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.4757       0.7102       0.5698        2222
     1       0.5449       0.3072       0.3929       2510

 accuracy          0.5183          0.5087          0.4964        4732
 macro avg          0.5183          0.5087          0.4964        4732
 weighted avg          0.5183          0.5087          0.4964        4732

Confusion Matrix (Test):
[[1578  644]
 [1739  771]]
```

```
K Value: 7
-----
Train Accuracy: 0.7609
Test Accuracy: 0.5110

Train Classification Report:
      precision    recall  f1-score   support

     0       0.6966       0.8679       0.7729        6653
     1       0.8511       0.6664       0.7475        7540

 accuracy          0.7739          0.7672          0.7609       14193
 macro avg          0.7739          0.7672          0.7609       14193
 weighted avg          0.7739          0.7672          0.7609       14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.4830       0.5891       0.5308        2222
     1       0.5485       0.4418       0.4894       2510

 accuracy          0.5157          0.5155          0.5110        4732
 macro avg          0.5157          0.5155          0.5110        4732
 weighted avg          0.5157          0.5155          0.5110        4732

Confusion Matrix (Test):
[[1309  913]
 [1401 1109]]
```

```

=====
K Value: 9
-----
Train Accuracy: 0.7663
Test Accuracy: 0.5296

Train Classification Report:
      precision    recall  f1-score   support

     0       0.7421    0.7685    0.7551     6653
     1       0.7891    0.7643    0.7765     7540

   accuracy        0.7656    0.7664    0.7663    14193
  macro avg       0.7656    0.7664    0.7658    14193
 weighted avg     0.7671    0.7663    0.7665    14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.4990    0.4446    0.4703     2222
     1       0.5516    0.6048    0.5770     2510

   accuracy        0.5253    0.5247    0.5236     4732
  macro avg       0.5253    0.5247    0.5236     4732
 weighted avg     0.5269    0.5296    0.5269     4732

Confusion Matrix (Test):
[[ 988 1234]
 [ 992 1510]]
=====

```

```

=====
K Value: 17
-----
Train Accuracy: 0.6209
Test Accuracy: 0.5342

Train Classification Report:
      precision    recall  f1-score   support

     0       0.7731    0.2709    0.4012     6653
     1       0.5910    0.9298    0.7227     7540

   accuracy        0.6821    0.6003    0.6209    14193
  macro avg       0.6821    0.6003    0.6209    14193
 weighted avg     0.6764    0.6209    0.5720    14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.5136    0.1526    0.2353     2222
     1       0.5376    0.8721    0.6651     2510

   accuracy        0.5256    0.5123    0.5342     4732
  macro avg       0.5256    0.5123    0.5342     4732
 weighted avg     0.5263    0.5342    0.4633     4732

Confusion Matrix (Test):
[[ 339 1883]
 [ 321 2109]]
=====

```

```

=====
K Value: 23
-----
Train Accuracy: 0.5650
Test Accuracy: 0.5344

Train Classification Report:
      precision    recall  f1-score   support

     0       0.7910    0.0979    0.1742     6653
     1       0.5511    0.9772    0.7047     7540

   accuracy        0.6710    0.5375    0.5650    14193
  macro avg       0.6710    0.5375    0.4394    14193
 weighted avg     0.6635    0.5650    0.4560    14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.5473    0.0495    0.0988     2222
     1       0.5339    0.9637    0.6071     2510

   accuracy        0.5406    0.5066    0.3890     4732
  macro avg       0.5406    0.5066    0.3890     4732
 weighted avg     0.5402    0.5344    0.4071     4732

Confusion Matrix (Test):
[[ 110 2112]
 [  91 2419]]
=====

```

```

=====
K Value: 31
-----
Train Accuracy: 0.5397
Test Accuracy: 0.5298

Train Classification Report:
      precision    recall  f1-score   support

     0       0.8191    0.0231    0.0450     6653
     1       0.5360    0.9955    0.6968     7540

   accuracy        0.6776    0.5093    0.5397    14193
  macro avg       0.6776    0.5093    0.3709    14193
 weighted avg     0.6687    0.5397    0.3913    14193

Test Classification Report:
      precision    recall  f1-score   support

     0       0.4595    0.0077    0.0151     2222
     1       0.5304    0.9920    0.6912     2510

   accuracy        0.4949    0.4998    0.3531     4732
  macro avg       0.4971    0.5298    0.3737     4732
 weighted avg     0.4971    0.5298    0.3737     4732

Confusion Matrix (Test):
[[  17 2285]
 [  20 2490]]
=====

```

Best k = 23 with Test Accuracy = 0.5344

We chose the KNN classifier because it is fairly straightforward and doesn't make assumptions about the data. However, KNN can have trouble when the feature space is very large because the distance between points becomes less meaningful. This helps to explain why our accuracy is close to random guessing. The model has trouble finding "similar" days in a high-dimensional space. However, KNN is a useful classifier to compare against the other models we used and it proved to be the model that performed best for our project.

Further Work:

As seen by the models prediction data and statistics, our methods of prediction led to poor results. Although we used the methods that we did, looking into further work on the subject, it may be worthwhile to take certain considerations and efforts going forward.

One method of making the models better would be to add weights to the articles that have bigger changes in the market. So, therefore, the more change an article has, the more weight that article's words would have. This could be coupled with analyzing which markets got affected by certain words and making different datasets and prediction models based on each industry.

Furthermore, you could take into account the possible offset of the articles affect (ie. some articles may lead to a change within one or two days of its posting).

We could also make this better by using a vectorization model, such as Doc2Vec, that takes into account grammatical and semantic values of the article. This would allow the vectorization to be more accurate for the tone of the article that may have a better ability to predict the change in the market.

Lastly, we could make it better by trying out more models with these different metrics to see if there is a better model than the ones used in this analysis. Possibly, we could add more values to help the prediction outside of only looking at the article values and a simple metric such as the DJIA going up or down.

Conclusion:

All models performed about the same, with the SVC, Ridge Regression, and K-Nearest Neighbors having an accuracy of ~53%. However, we found the K-Nearest Neighbors model performed slightly better than the rest with an accuracy of 53.44%. Linear Regression performed slightly worse with an accuracy of ~50%.

As seen by this data, we cannot really accurately predict whether the DJIA score of a company will go up or down based on the top headlines for that day with the TF-IDF vectorization model and the models that we have chosen. Although if the things noted in further work were implemented, it may increase the possibility of being able to do so.