Sentiment Analysis on News Headlines to Predict Stock Market Fluctuations

Prepared by:
Jahnavi Yandapalli

Agenda

- Introduction
- Exploration
- Analysis
- Insights

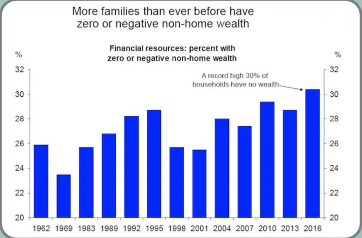
Investments over the Years

In 1989, 32% of US families invested in the stock market

In 2019, 53% of US families invested in the stock market

Majority of investments are from retirement accounts

Investments over the Years



According to **The Federal Reserve's Survey of Consumer Finances**, **30% of households** had **no wealth** in 2016. Although there have been an **increase** in the number of families **investing in the stock market**, **fewer families** have been able to **secure wealth**.

Introduction

Exploration

Analysis

Insights

Sentiment Analysis

Sentiment analysis (opinion mining) is a **natural language processing technique** that ascertains whether the language in a text is **positive, negative, or neutral**.

Types: rule-based, **automatic**, hybrid

Sentiment Metrics

Polarity: Measure of **positive** and **negative** language

Subjectivity: Measure of **personal opinion**

Compound

Sum of the **valence scores** and then **normalization** of them between **[-1,1]**

Dataset: Reddit

Dataset: Reddit WorldNews Channel (r/worldnews)
Timeline: June 8, 2008 to July 1, 2016
Information: Daily news headlines (Top 25 daily headlines)

Date	Headline	Label
Monday - Friday: 9:30am - 4pm **1,989 days**	Sentiment Analysis	Dow Jones Industrial Average 0: Decrease 1: Same/Increase

Dataset: DJIA

Dataset: Dow Jones Industrial Average (DJIA)
Timeline: June 8, 2008 to July 1, 2016
Information: Prices of the DJIA

Open/Close

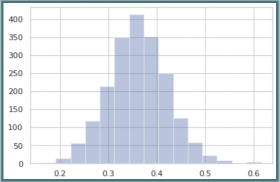
Adj. Close

High/Low

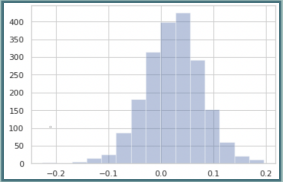
Volume

Polarity and Subjectivity

Subjectivity

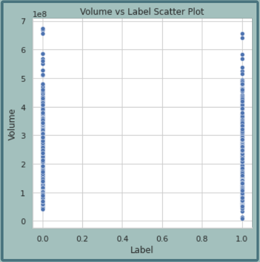


Polarity



Normal Distribution: Unimodal, No Cluster, No Outliers

Volume of Stocks Traded

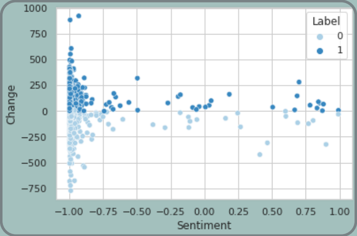


Statistics

For Labels 0 and 1, the **volume of stocks traded** are **uniformly distributed**. The volume traded is slightly **lower with Label 1**, which could be attributed to **retainment of stocks due to a positive mindset.**

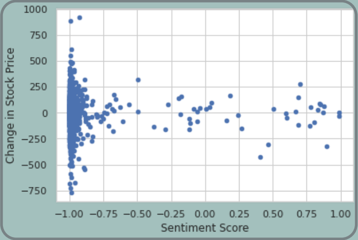
Introduction | **Exploration** | Analysis | Insights

Linear Regression



There is a **large cluster** with both Labels 0 and 1 to the leftmost negative value of close to -1.00. Both labels showing a cluster around the same negative value suggests that sentiment **may not play a significant impact** on the fluctuation of the stocks.

Linear Regression



Statistics

Equation:
Predicted Change in Stock Price = -10.341 - 14.892(Sentiment Score)

R^2: 0.044%

MSE: 19926.4650

Analysis: Linear Regression

Equation

Coefficient: -14.892

The negative value suggests that an **increase** in **sentiment** leads to a **decrease** in the **DJIA**.

Correlation

The coefficient of determination (r^2) suggests that **0.044%** of the variation in **stock prices** can be determined by the **sentiment score.**

Mean Squared Error

MSE: 19926.4650

The **large error** indicates the model's **incorrectedness** in predicting most of the fluctuation in prices.

Analysis: Linear Regression

Conclusion

Since the coefficient of determination is **low** at **0.044%** and the **MSE is high**, there is **insufficient evidence** to conclude existence of a **strong linear relationship** between sentiment scores and changes in stock prices.

Next Steps

Since a linear regression model **cannot** accurately predict the relationship, the data will be **trained** and **tested** to create a linear discriminant analysis prediction model.

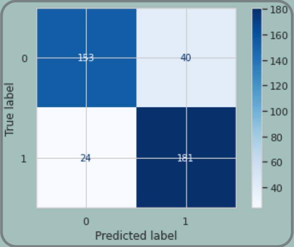
Linear Discriminant Analysis

Model

The dataset was split into training and testing data (**80:20 ratio**). Once the fit of the model was created, the **accuracy** of the **predictions** was calculated to be at **84%**.

	precision	recall	f1-score	support
0	0.86	0.79	0.83	193
1	0.82	0.88	0.85	205
accuracy			$\boxed{0.84}$	398
macro avg	0.84	0.84	0.84	398
weighted avg	0.84	0.84	0.84	398

Linear Discriminant Analysis



Confusion Matrix

The **confusion matrix** describes the number of times the prediction model **guesses wrong** for each label. Evidently, the wrong guesses are **rather low** for both labels.

Testing the Model

Dataset

	Label	Open	High	Low	Volume	Polarity	Subjectivity	compound	positive	negative	neutral
2	0	11781.700195	11782.349609	11601.519531	173590000	-0.044302	0.536234	-0.9715	0.056	0.128	0.816
1985	1	17190.509766	17409.720703	17190.509766	112190000	0.046560	0.352649	-0.9571	0.102	0.132	0.767

	Actual	Prediction	
Row 2	0	0	✓
Row 1985	1	1	✓

Limitations

Confounding Variables/Influences

The stock market is **not solely affected** by news articles and headlines, which **reduces the impact** of the prediction model. Others factors can include interest rates, politics, and inflation.

Components of Model

The model **requires many components** that may not always be available when investors make trading decisions. **Removing** such factors **reduces the accuracy of the model**, however.

Conclusion

A **linear regression model** does not accurately predict the fluctuation in stock prices based on sentiment as evident from the **low coefficient of determination**. However, creating a **linear discriminant model is rather accurate** at 84% in predicting the direction of the fluctuation in stock prices.

By providing the model the relevant information, **investors can more informatively** make trading decisions after seeing the publication of certain news. The model is restricted to solely providing the **direction of the movement** and not the **extent**, so conclusions cannot be made **too broad**.

Questions?

Appendix: Cleaning the Data

```
Headline = []

for topnews in range(0, len(MergedData.index)):
    Headline.append(" ".join(str(x) for x in MergedData.iloc[topnews, 2:27]))

Cleaned_Headline = []
for i in range(0, len(Headline)):
  Cleaned_Headline.append(re.sub("b'", '', Headline[i]))
  Cleaned_Headline[i] = re.sub('b"', '', Cleaned_Headline[i])
  Cleaned_Headline[i] = re.sub("\'", '', Cleaned_Headline[i])

MergedData['Daily News'] = Cleaned_Headline
MergedData
```

Appendix: Sentiment Analysis

```
def polarity_score(text):
  return TextBlob(text).sentiment.polarity

#Obtaining the Subjectivity Scores
def subjectivity_score(text):
  return TextBlob(text).sentiment.subjectivity

MergedData['Polarity'] = MergedData['Daily News'].apply(polarity_score)
MergedData['Subjectivity'] = MergedData['Daily News'].apply(subjectivity_score)
MergedData
```

Appendix: Sentiment Analysis

```python
compound = []
pos = []
neg = []
neu = []
SIA = 0

for i in range (0, len(MergedData['Daily News'])):
  SIA = getSIA(MergedData["Daily News"][i])
  compound.append(SIA['compound'])
  pos.append(SIA['pos'])
  neg.append(SIA['neg'])
  neu.append(SIA['neu'])

MergedData['compound'] = compound
MergedData['positive'] = pos
MergedData['negative'] = neg
MergedData['neutral'] = neu

MergedData
```

Appendix: Linear Regression

```
SentimentScore = ['Sentiment']

X = Data[SentimentScore]
y = Data.Change

linreg = LinearRegression()
linreg.fit(X, y)

# coefficents
print("The y intercept: ", linreg.intercept_)
print("The single coefficient:", list(zip(SentimentScore,linreg.coef_)))


# r^2

y_pred = linreg.predict(X)
print("R^2: ", metrics.r2_score(y, y_pred))

# Evaluate MSE
print("MSE: ", metrics.mean_squared_error(y, y_pred))
```

Appendix: Data Set for Model

	Label	Open	Close	High	Low	Volume	Polarity	Subjectivity	Sentiment	positive	negative	neutral	Change
0	0	11432.089844	11734.320312	11759.959961	11388.040039	212830000	-0.048568	0.267549	-0.9982	0.041	0.235	0.724	302.230468
1	1	11729.669922	11782.349609	11867.110352	11675.530273	183190000	0.121956	0.374806	-0.9858	0.089	0.191	0.721	52.679687
2	0	11781.700195	11642.469727	11782.349609	11601.519531	173590000	-0.044302	0.536234	-0.9715	0.056	0.128	0.816	-139.230468
3	0	11632.809570	11532.959961	11633.780273	11453.339844	182550000	0.011398	0.364021	-0.9809	0.066	0.146	0.788	-99.849609
4	1	11532.070312	11615.929688	11718.280273	11450.889648	159790000	0.040677	0.375099	-0.9882	0.094	0.189	0.717	83.859376
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1984	0	17355.210938	17140.240234	17355.210938	17063.080078	138740000	-0.014015	0.352311	-0.9644	0.094	0.148	0.758	-214.970704
1985	1	17190.509766	17409.720703	17409.720703	17190.509766	112190000	0.046560	0.352649	-0.9571	0.102	0.132	0.767	219.210937
1986	1	17456.019531	17694.679688	17704.509766	17456.019531	106380000	0.052622	0.389617	-0.9975	0.091	0.225	0.684	238.660157
1987	1	17712.759766	17929.990234	17930.609375	17711.800781	133030000	0.011243	0.382566	-0.9977	0.061	0.202	0.738	217.230468
1988	1	17924.240234	17949.369141	18002.380859	17916.910156	82160000	-0.035458	0.320261	-0.9983	0.059	0.212	0.729	25.128907

1989 rows × 13 columns

Appendix: Creating the Model

```
A = Testing

B = np.array(NewDataSet['Label'])

#Spitting the data into testing and training groups
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size=0.2, random_state=0)

#Creating the model
Model = LinearDiscriminantAnalysis().fit(A_train, B_train)

#Testing the model
PredictFluctuation = Model.predict(A_test)
PredictFluctuation

print(classification_report(B_test, PredictFluctuation))
```