# Stock Sentiment Analysis Using Machine Learning Techniques

June 19,2024


Name: Bhavya Shah

Enrollment No:23115032

Electrical Engineering


FINANCE CLUB
IIT ROORKEE

# REPORT

## Overview

**This project aims to develop a sentiment analysis model for predicting stock price movements based on textual data from various sources like news report, articles and social media posts or magazine. I have selected 3 companies for analysis 'APPLE','BOEING' and 'META'. Machine Learning Model used in the project analyze the news titles for the mentioned and generates features such as negative, positive and neutral, polarity, subjectivity, Word2Vec Feature and Fin BERT Features using some of the advanced NLP (natural language processing) techniques. These features are further used to establish connection between news sentiment and variation in stock prices. From the ML model and output variation of stock prices, we can form some trading strategy which is best for profitability.**

# Trajectory of Project

## 1.Data Acquisition

In order to get new headlines of 3 mentioned companies, we need to scrape pages of certain social media pages, news aggregators or few finance websites. I have used market insider for getting the news titles and yahoo finance for obtaining the numerical data about the stocks of the company.

Here is the code:

```
#CODE FOR WEB SCRAPPING NEWS TITLE OF APPLE COMPANY FROM MARKET INSIDER
url = 'https://markets.businessinsider.com/news/aapl-stock'
bh=[]

#I have scrapped 390 pages of information for the Apple stock
for i in range(1,390) :
  url2=url+'?p='+str(i)
  res = requests.get(url2)
  html = res.text
  soup = BeautifulSoup(html,'lxml')
  columns = ['datetime','title']
  df=pd.DataFrame(columns=columns)

#From the appropriate class,i have extracted useful textual data
  articles= soup.find_all('div',class_='latest-news__story')
  for article in articles:
    datetime = article.find('time',class_='latest-news__date').get('datetime')
    title = article.find('a',class_='news-link').text
    bh.append([datetime,title])
    df=pd.DataFrame(bh,columns=df.columns)
df['date'] = pd.to_datetime(df['datetime']).dt.date
drive.mount('drive', force_remount = True)

#Csv sheet is created to add these data in respective column and saved in drive

df.to_csv('df.csv')
!cp df.csv "drive/My Drive/"
```

## 2.Textual Data Cleaning and Preparation

Now, for collected news headings we start cleaning process which includes removing stop words and punctuations. Here we also perform lemmatization and tokenization.

Here is the code:

```
#This is written for tokenising and cleaning new title of Apple Stock

lemmatizer = WordNetLemmatizer()
def clean_text(text):
    words = word_tokenize(text)
    words = [word.lower() for word in words]
    #It removes all all stop words from news title

    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word.isalpha() and word not in stop_words]
    words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(words)
#Formation of a new column for cleaned title text which has final outcome of cleaning

df2['cleaned_title'] = df2['title'].apply(clean_text)
print(df2)
```

## 3.Annotation and Labeling

Merging cleaned news title data with the numerical data of respective date in one data frame. Labeling it as 1 if increase is detected else 0 would be the label.

## Here is the code:

```
#Below code is for transforming textual cleaned title text into vector by running it on Word2vec

sentences = df2['cleaned_title'].apply(lambda x: x.split()).tolist()
model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)
#Defining of word2vec function for transformation

def word2vec(words, model, num_features):
    feature_vec = np.zeros((num_features,), dtype="float32")
    nwords = 0
    index2word_set = set(model.wv.index_to_key)
    for word in words:
        if word in index2word_set:
            nwords += 1
            feature_vec = np.add(feature_vec, model.wv[word])
    if nwords > 0:
        feature_vec = np.divide(feature_vec, nwords)
    return feature_vec
num_features = 100
#Added column named features which has vectors obtained via above defined function

df2['Word2Vec_Features'] = df2['cleaned_title'].apply(lambda x: word2vec(x, model, num_features))
print(df2[['title', 'Word2Vec_Features']])
```

## 4.Feature Extraction

**Sentiment Score are well evaluated using inbuilt function present in Vader. Also, numerical vector was obtained from the textual data with the help of NLP models like Word2Vec and FinBERT.**

## Here is the code:

```
#It is important to set hugging face token as an Environment Variable
#below is code for it (authenticating access to Hugging Face Model)

os.environ['HF_TOKEN'] = 'collab'
#Defining a function to add more feature with the help of FinBERT model

def FinBERT_Feature(text, model, tokenizer):
    #Transforms text into tensors (tokenize)

    inputs = tokenizer(text, return_tensors='pt', max_length=512, truncation=True, padding='max_length')
    with torch.no_grad():
        outputs = model(**inputs)
    last_hidden_state = outputs.last_hidden_state
    feature_vector = last_hidden_state.mean(dim=1).cpu().numpy().flatten()
    return feature_vector
finbert_model = BertModel.from_pretrained('yiyanghkust/finbert-tone')
finbert_tokenizer = BertTokenizer.from_pretrained('yiyanghkust/finbert-tone')
#Adding column to save FinBERT feature extracted from cleaned title text

df2['FinBERT_Features'] = df2['cleaned_title'].apply(lambda text: FinBERT_Feature(text, finbert_model, finbert_tokenizer))
finbert_feature = pd.DataFrame(df2['FinBERT_Features'].to_list(), columns=[f'FinBERT_{i}' for i in range(len(df2['FinBERT_Features'][0]))])
word2vec_feature = pd.DataFrame(df2['Word2Vec_Features'].to_list(), columns=[f'Word2Vec_{i}' for i in range(len(df2['Word2Vec_Features'][0]))])
final_data4 = pd.concat([df2, finbert_feature, word2vec_feature], axis=1)
final_data4.drop(['FinBERT_Features', 'Word2Vec_Features'], axis=1, inplace=True)
print(final_data4)
```

## 5.Machine Learning Model Development

Thereafter we train our ML model by providing appropriate set of data which we get from feature extraction. I used 3 model to train my dataset, Logistic Regression, Light Gradient Boosting Model and Linear Discriminant Analysis. Then fine-tuned them using respective parameters in order to lower individual error and increase individual accuracy and robustness (Grid Search CV). Then applying ensemble on them would help us better generalization of data and improve overall accuracy and error compensation.

## Here is the code:

```
                                    # MODEL-3 #

# Linear Discriminant Analysis Model

X = merged_df2.drop(['Label'], axis=1)
y = merged_df2['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lda = LinearDiscriminantAnalysis()
#For fine tuning i am using hyper parameter tuning with grid search

param_grid = {'solver': [ 'lsqr', 'eigen'],'shrinkage': [None, 'auto'] + list(np.linspace(0, 1, 10))}
grid_search = GridSearchCV(lda, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_lda = grid_search.best_estimator_
best_params = grid_search.best_params_
print(f"Best parameters found: {best_params}")
y_pred = best_lda.predict(X_test)
#Below is the code for printing results of Model-3

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

## 6.Perfomance Evaluation

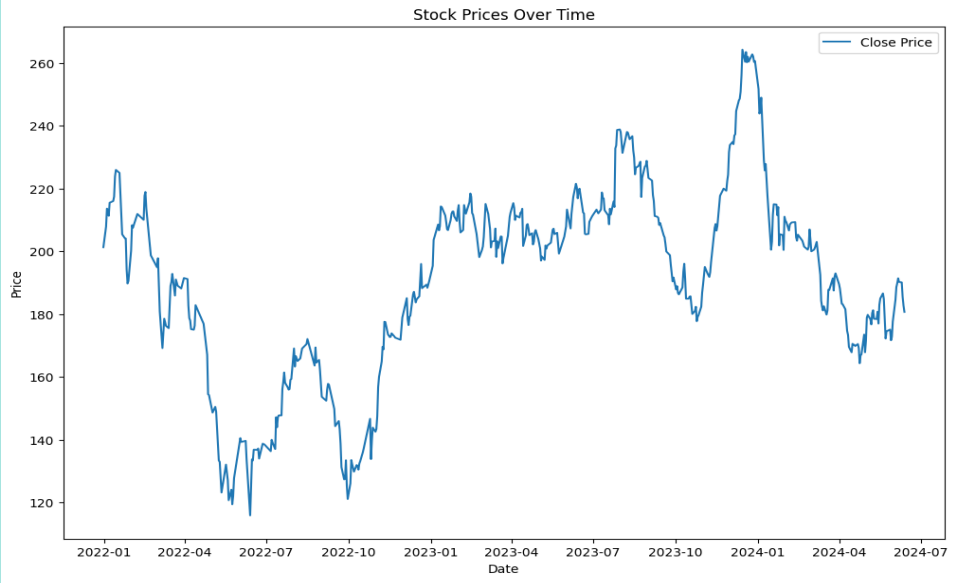
After ensembling ML models, we get to evaluate their performance. (get final result here)

```
# Here i am ensembling the 3 ML models for getting better Output

#Ensemble Model

X = merged_df2.drop(['Label'], axis=1)
y = merged_df2['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=46)
ensemble=VotingClassifier(estimators=[('lr',best_logreg),('ada',best_ada_model),('lda',lda)],voting='soft')
ensemble.fit(X_train, y_train)
y_pred = ensemble.predict(X_test)

#Below is the code for printing results of Ensemble Model
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

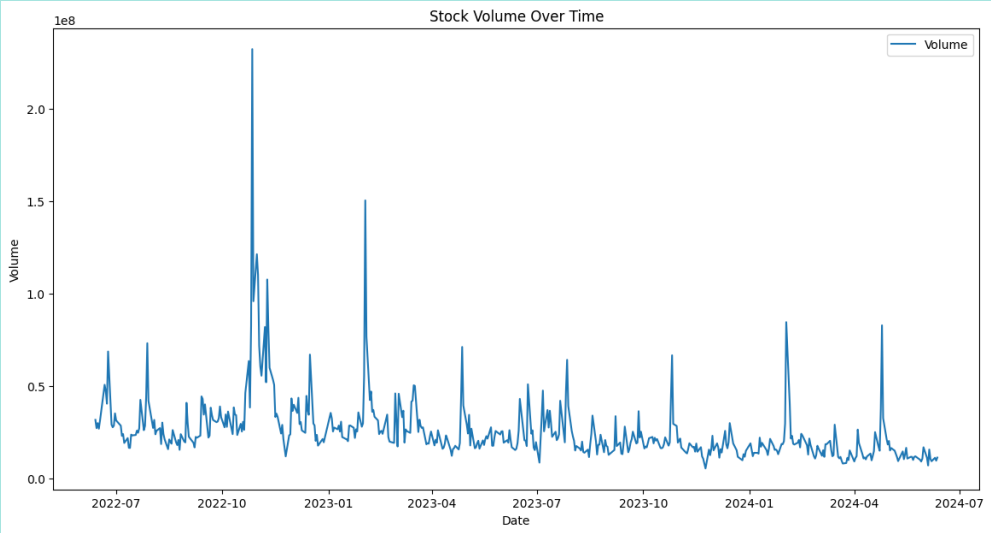
# STOCK PRICE OVER TIME FOR THE THREE COMPANIES

## APPLE

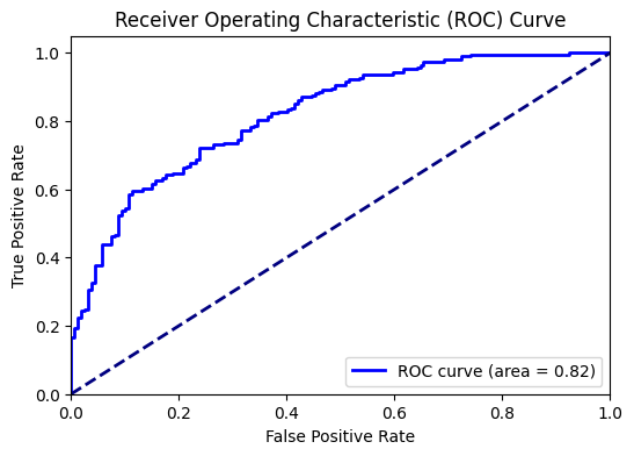

## BOEING

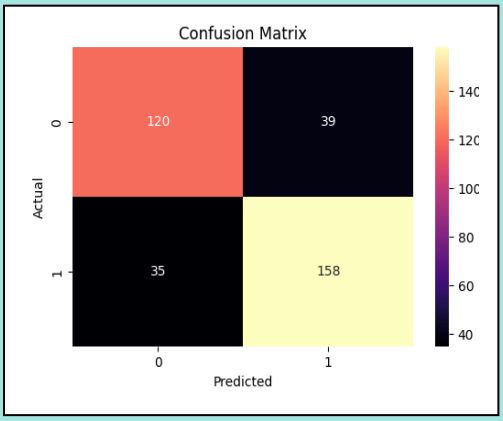


# META

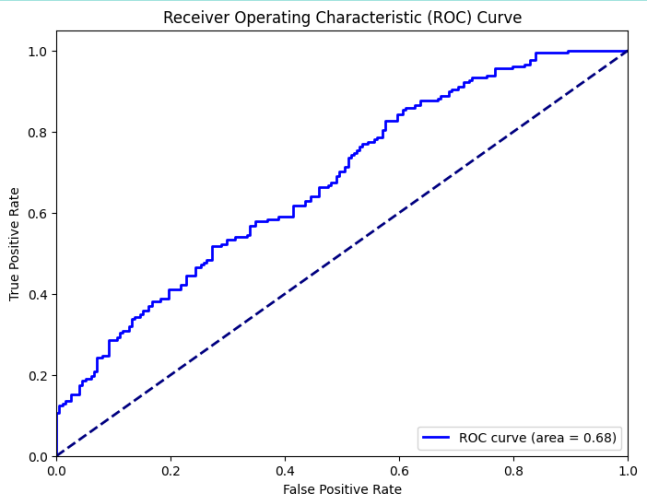


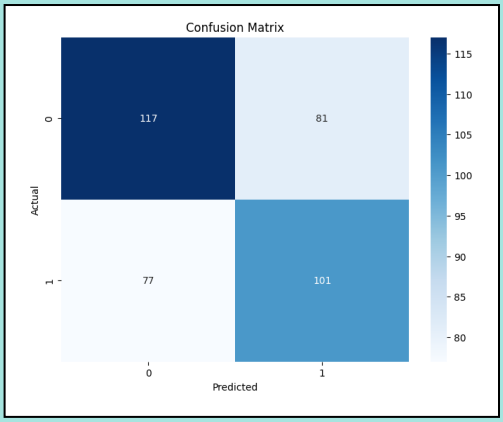

## Metrics for Apple



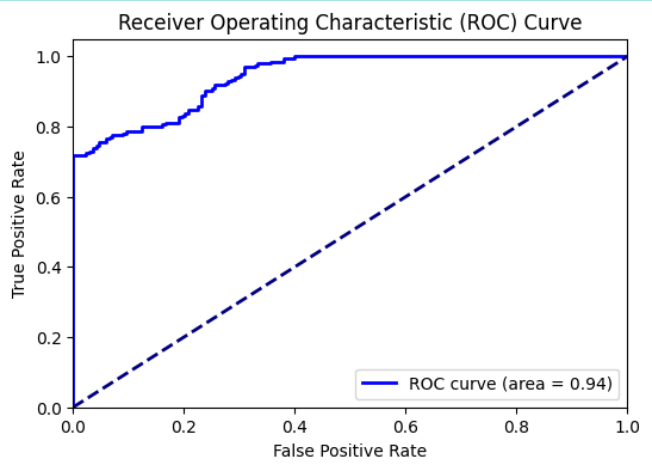



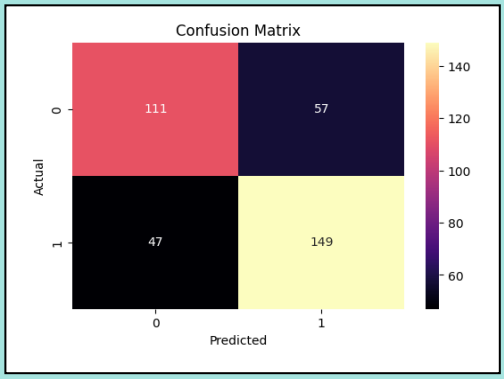
## Metrics for Boeing




## Metrics for Meta



## 7.Trading Strategy Back testing

**Based upon the ensemble results I formed a column where Predicted Label is entered which has predicted increase or decrease. Along with that we also provide Open and Close's Actual value in a column and save a new file which consist Open, Close and Predicted Label as column. Now I used RSI (Relative Strength Index) trading strategy on my data.**

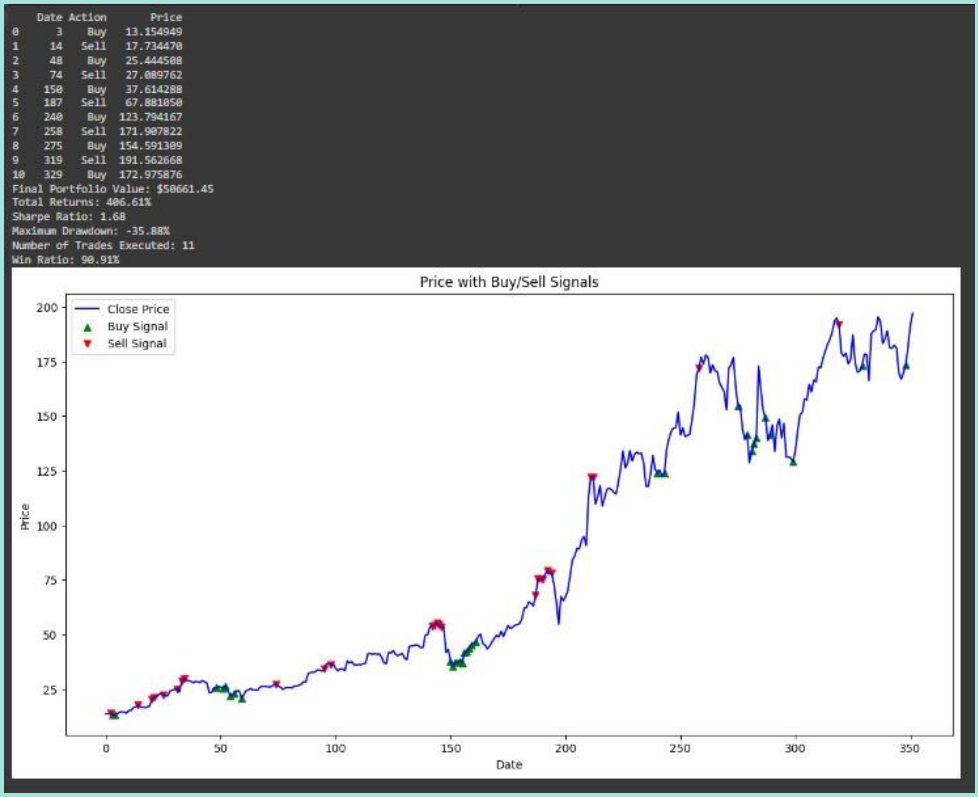
**STRATEGY:**

**The RSI is an oscillator-based momentum trading strategy that moves from 0 to 100. It is generally bound to the extreme major levels of 70(overbought) and 30 (oversold).**

**But these key values could be changed in occur dance to accuracy of prediction and risk-taking factor. Keeping that as a variable with the fall below the user defined thresholds, the RSI gives the sell signal, showing that the security might be overbought and hence could be due for a correction. On the other hand, a buy signal is generated when the RSI crosses back above a user**

**defined value from below, indicating that the stock may have been oversold and could rebound Signals should be checked with other indicators, and a good risk management system must be in place to minimize your losses.**

## APPLE

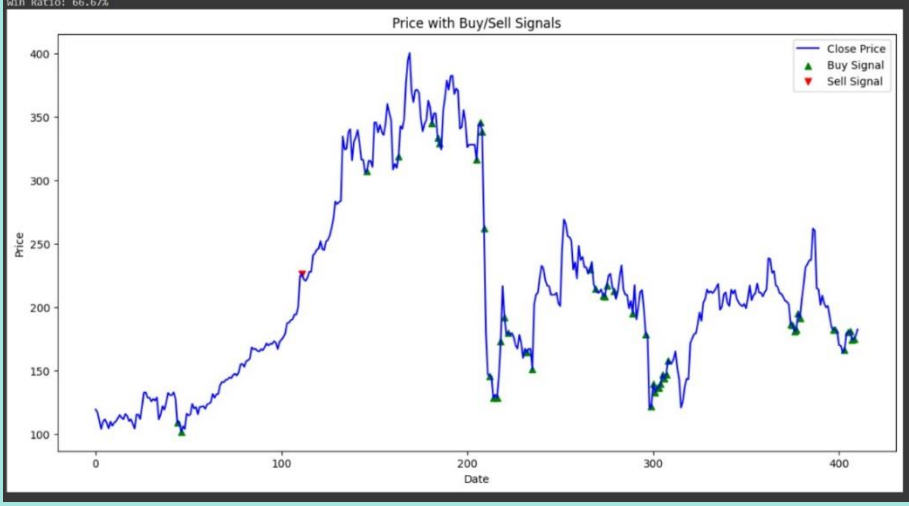


| | Date | Action | Price |
|---|---|---|---|
| 0 | 3 | Buy | 13.154949 |
| 1 | 14 | Sell | 17.734470 |
| 2 | 48 | Buy | 25.444508 |
| 3 | 74 | Sell | 27.089762 |
| 4 | 150 | Buy | 37.614288 |
| 5 | 187 | Sell | 67.881050 |
| 6 | 240 | Buy | 123.794167 |
| 7 | 258 | Sell | 171.907822 |
| 8 | 275 | Buy | 154.591309 |
| 9 | 319 | Sell | 191.562668 |
| 10 | 329 | Buy | 172.975876 |

Final Portfolio Value: $50661.45
Total Returns: 406.61%
Sharpe Ratio: 1.68
Maximum Drawdown: -35.88%
Number of Trades Executed: 11
Win Ratio: 90.91%

# BOEING

```
   Date Action       Price
0    44    Buy  109.297104
1   111   Sell  226.363235
2   146    Buy  307.420807
Final Portfolio Value: $12294.71
Total Returns: 22.95%
Sharpe Ratio: 0.75
Maximum Drawdown: -69.70%
Number of Trades Executed: 3
Win Ratio: 66.67%
```




# META

```
   Date Action       Price
0    67    Buy  152.867798
1    83   Sell  197.150818
2    95    Buy  160.409805
3   123   Sell  169.989639
4   158    Buy  183.355453
5   181   Sell  207.879440
6   195    Buy  169.320343
7   215   Sell  292.889252
8   285    Buy  333.386261
9   355   Sell  242.922241
Final Portfolio Value: $19467.21
Total Returns: 94.67%
Sharpe Ratio: 0.89
Maximum Drawdown: 0.00%
Number of Trades Executed: 10
Win Ratio: 80.00%
```