finance club

STOCK SENTIMENT ANALYSIS

PROJECT REPORT

KARAN SARDAR
22118035
BTECH (MT) III YR

About the Project

The project aimed at building a sentiment analysis model that drives through news articles to understand the overall sentiment related to a particular stock. Some technical parameters such as open, high and low prices of the different days were also taken as features to train the model.

What is Sentiment Analysis ?

Sentiment analysis is a natural language processing (NLP) technique used to determine the emotional tone behind a body of text. It involves analyzing text to identify the sentiment expressed — whether it's positive, negative, or neutral.

In context of the stock market, this project aims to utilise news related to a particular stock, major geopolitical events and industry specific developments to predict bullish or bearish trends

Approaches used

In this project, Machine learning models such as Support Vector Machne, Linear Discriminant, Random forest and Logistic Regression are used as these are the Classification algorithms.

Also the ARIMA (Autoregressive Integrated Moving Average) model was taken into account. ARIMA models are typically applied to time series data to make predictions based on historical patterns and relationships within the data.

Flow of Project

1.Data Collection

- Web Scraping using Beautiful Soup library was done to fetch the data from New York times Archives. I used the NY times developer API to fetch news articles from 2012 to end of 2023.
- The prediction dataset for news articles was also fetched from New York Times for the entire Year 2024 till date.
- SP500 stock data was also fetched using Yfinance library.
- The Sp 500, often referred to simply as the "S&P," is a stock market index that measures the performance of 500 large companies listed on stock exchanges in the United States.

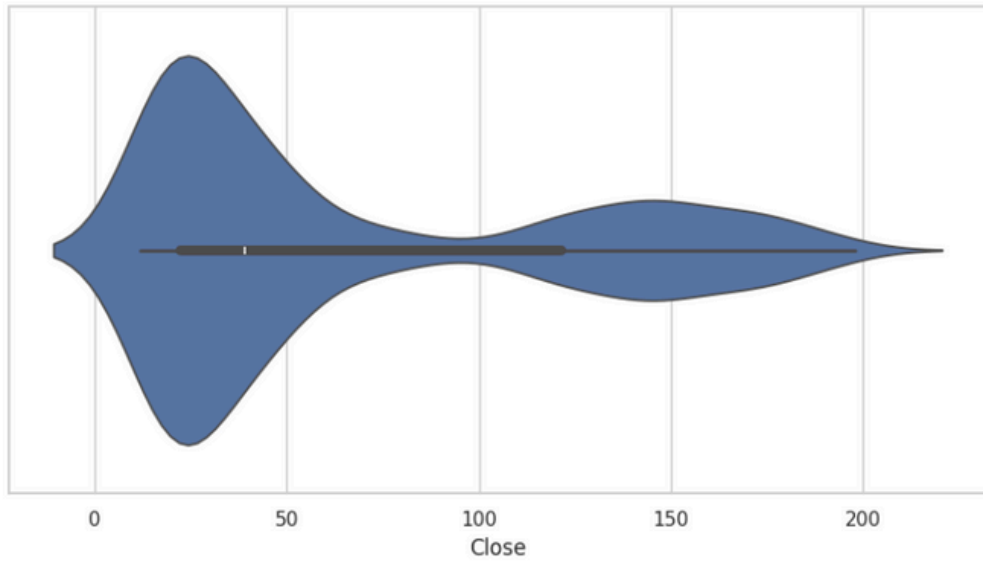
2.Data Labelling and Sentiment Scores

- Data was analysed using VADER to determine positivity, negativity, neutrality, and compound scores. Due to the predominantly factual nature of the stock price-related data, detecting nuanced sentiments like irony or sarcasm posed minimal challenges.
- getSubjectivity and getPolarity functions were used to calculated the subjectivity score and polarity score of the text using TextBlob.
- getSIA function was used as the SentimentIntensityAnalyzer class to get a more detailed sentiment breakdown.
- Polarity measures how positive or negative a text is. It's a float value within the range [- 1.0, 1.0], where -1.0 indicates highly negative sentiment, 0.0 indicates neutral sentiment, and 1.0 indicates highly positive sentiment.
- Subjectivity measures how subjective or opinionated the text is. It's also a float value within the range [0.0, 1.0], where 0.0 is very objective (factual) and 1.0 is very subjective (opinionated).
- Sample dataset used below:

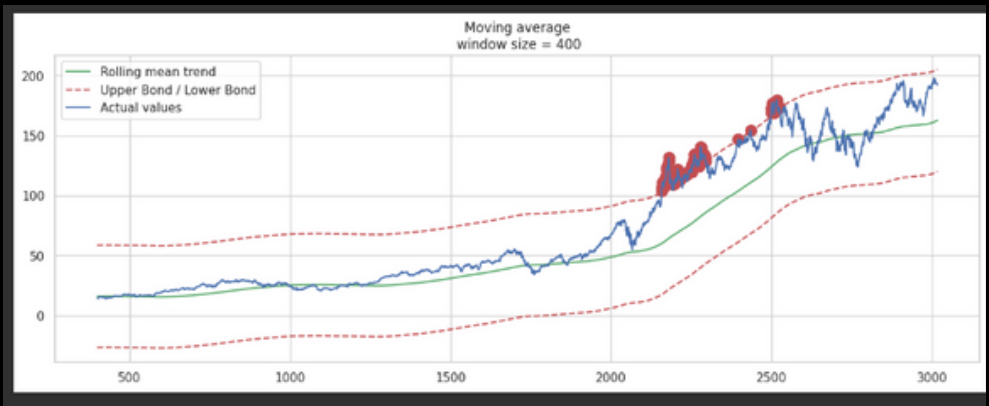
	Date	Headlines	Open	High	Low	Close	Volume	Yesterday	Subjectivity	Polarity
0	2012-01-03	democrat becom target occupi protest european ...	12.378494	12.472224	12.366399	12.433825	302220800	12.245458	0.443480	0.112125
1	2012-01-04	decod iowa caucu coverag explain tonight poll ...	12.396637	12.538141	12.374867	12.500648	260022000	12.433825	0.409932	0.119099
2	2012-01-05	bachmann next decis whether run reelect questi...	12.546302	12.655149	12.477363	12.639427	271269600	12.500648	0.440050	0.147485

3. EDA (Exploratory Data Analysis)

- The EDA was done on “Close” parameter of the dataset.
  I plotted a violin plot for analysis



- Closing price of the stock was also plotted along with percentage change to analyze the price deviation from previous day
- Moving Average was also plotted with rolling mean trends.



The moving average window clashes with the actual data at some points in the plot because of the drastic peak and trough pattern.

- Stationarity of the data was checked using statsmodels.tsa.stattools module
- The data turned out to be non stationary and was made stationary.
- ADF statistic was checked.

```
ADF Statistic: -12.594009309510227
n_lags: 1.7953102145124775e-23
p-value: 1.7953102145124775e-23
Critial Values:
   1%, -3.4325330913621452
Critial Values:
   5%, -2.862504548608965
Critial Values:
   10%, -2.5672834546224057
========================================
The data is stationary
```

4. Cleaning and Preprocessing was done by making the headline _cleaning_func function.
It iterates through each headline in the 'Headlines' column of the DataFrame df, applying the headline_cleaning_func function to each headline, and then assigns the cleaned headlines back to the 'Headlines' column in df. It assumes that emoji is the correct library for handling emojis, which converts them to text-based representations.

5.Feature Extraction and Data Splitting

- Required features to train the model were stored in feature variable as predictors and the Label(Up/Down) feature was stored as the target feature
- Train test split was applied through sklearn's module.

```
X_train, X_test, y_train, y_test = train_test_split(feature, target, test_size=0.3,
qt = QuantileTransformer(output_distribution='normal')
X_train = qt.fit_transform(X_train)
X_test = qt.transform(X_test)
```

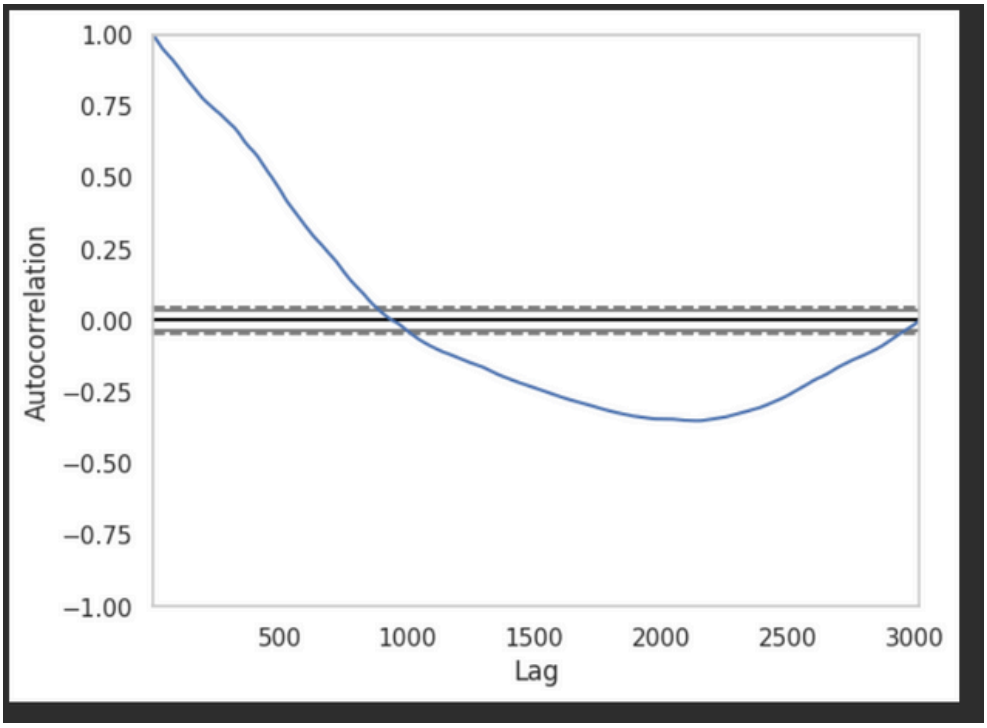
- Sklearn's Quantile Transformer scaling method was used to scale the values.
- QuantileTransformer applies a non-linear transformation to make data conform more closely to a uniform or Gaussian distribution, depending on the output distribution parameter.

6.Applying classification models and training

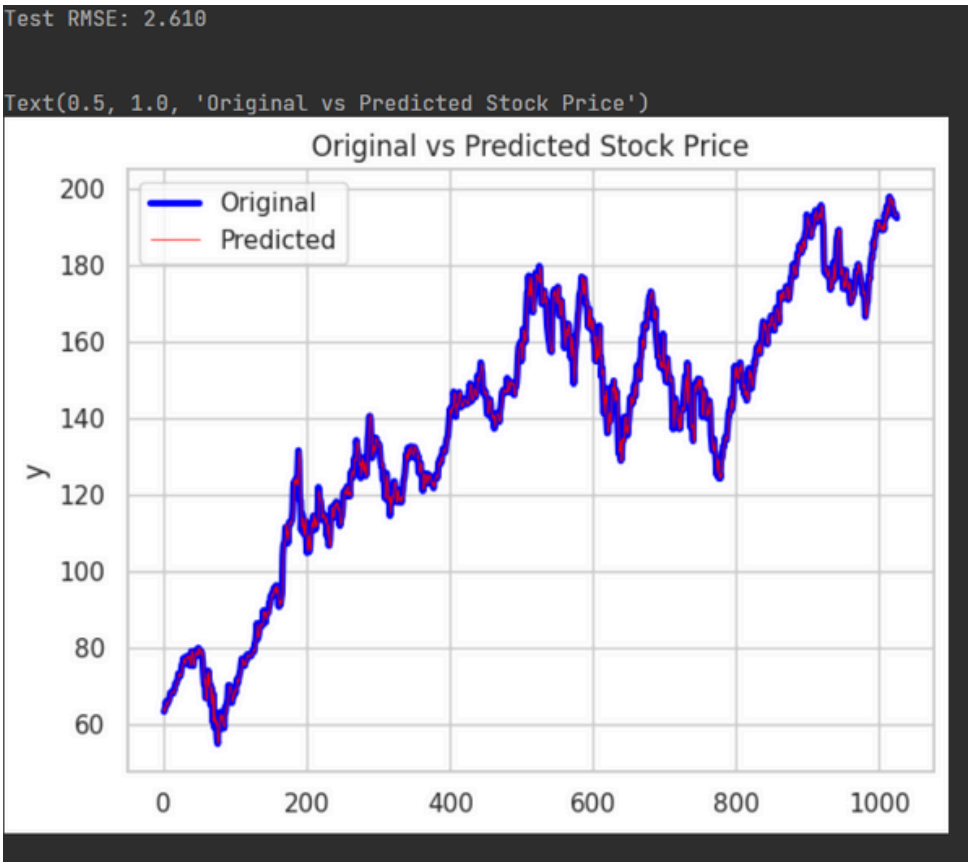
- Each stock was trained and tested with different algorithms (LDA, Random Forest, Support Vector Machine and Logistic Regression).
- The model with best accuracy and precision was considered. Moreover, the area under the ROC curve was also studied and the model giving the maximum area for both train and test sets.

7.ARIMA

- autocorrelation plot was plotted giving valuable insights



- Time series forecasting using ARIMA, evaluating the forecast accuracy with RMSE, and visualization of the results using matplotlib. was done
- RMSE was quite satisfactory.



7. Model Prediction

- The scrapped predicition model was loaded and merged with the sp500 stock data.
- getSIA, getSubjectivity and getPolarity were used for sentiment scores on this dataset.
- Sample dataset

	Date	Headlines	Subjectivity	Polarity	Compound	Negative	Neutral	Positive
0	2024-01-01	pga tour saudiback liv extend deadlin final de...	0.404672	0.120707	-0.9490	0.181	0.711	0.108
1	2024-01-02	south korean opposit leader stab yearold die s...	0.422998	0.175054	-0.9953	0.198	0.740	0.061
2	2024-01-03	epstein document name promin figur expect rele...	0.447047	0.136802	-0.9979	0.222	0.707	0.071
3	2024-01-04	unseal document shed light epstein misde littl...	0.405678	0.083816	-0.9946	0.174	0.746	0.079
4	2024-01-05	town empti farm languish war stalk israelileba...	0.349820	0.137964	-0.9981	0.196	0.748	0.056

- Same feature extraction and train_test_split_methods were followed.
- LDA, SVM, Random forest and Logistic Regression were again used to predict the data as all of them had same accuracies and ROC curve areas.
- Weighted Average/Precision score of all classifiers was calculated

```
Weighted Average of All Classifiers
              precision    recall  f1-score   support

           0       0.46      0.33      0.38        52
           1       0.48      0.62      0.54        52

    accuracy                           0.47       104
   macro avg       0.47      0.47      0.46       104
weighted avg       0.47      0.47      0.46       104

Weighted Precision Score: 0.46853569987898347
```

8.Generating and Plotting Buy/Sell Signal Chart

- The following strategy was used to generate trading signals

```
def generate_signal(row):
    if row['Diff'] > 0 and row['Cumu_Label(Up/Down)'] ==1:
        return "Buy"
    elif row['Diff'] <= -1 and row['Cumu_Label(Up/Down)'] ==-1:
        return "Sell"
    else:
        return "Hold"
```

- As a result the following chart came and here is the visual.

9. Evaluation Metrics

- **Sharpe Ratio**
  It measures the return of an investment compared to its risk.
  A Sharpe ratio >1 is considered good.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

Where:

- R_p= The expected return of the portfolio (investment),
- R_f = The risk-free rate of return,
- $R_p - R_f$ = Portfolio's excess return
- σ_p = The standard deviation of the portfolio's excess return

- **Maximum Drawdown**
  Represents the maximum loss from the peak values of an investment to its lowest point before a new peak is reached.

$$MDD\ \% = \frac{peak\ value - trough\ value}{peak\ value} * 100$$

- **Number of trades executed**

  It is the total number of buy and sell transactions of different stocks made within a specific time period. It helps traders understand volatility and market trends better.

- **Win Ratio**
  It is the proportion of profitable trades or investments compared to the total number of trades or investments made over a period of time

$$win\ ratio = \frac{Total\ number\ of\ winning\ trades}{Total\ number\ of\ trades} * 100$$

- Above are the parameters on which the model was tested.
- Following code was used:

```
num_trades = buy_signals.shape[0]

min_trades = min(len(buy_signals), len(sell_signals))
buy_signals = buy_signals.iloc[:min_trades]
sell_signals = sell_signals.iloc[:min_trades]

trades = pd.DataFrame({
    'Buy_Date': buy_signals['Date'].values,
    'Sell_Date': sell_signals['Date'].values,
    'Buy_Price': buy_signals['Close'].values,
    'Sell_Price': sell_signals['Close'].values,
    'Profit/Loss': sell_signals['Close'].values - buy_signals['Close'].values
})
win_ratio = (trades['Profit/Loss'] > 0).mean()


trades['Return'] = trades['Profit/Loss'] / trades['Buy_Price']

mean_return = trades['Return'].mean()
std_return = trades['Return'].std()

risk_free_rate = 0.0

sharpe_ratio = (mean_return - risk_free_rate) / std_return
cumulative_return = (trades['Return'] + 1).prod() - 1
trades['Investment'] = trades['Buy_Price']
total_investment = trades['Investment'].sum()
trades['Weighted_Return'] = trades['Profit/Loss'] / trades['Investment']
portfolio_total_return = trades['Profit/Loss'].sum() / trades['Investment'].sum()
```

10.Nvidia Stock Predictions

- Nvidia is now the world's most valuable company as of 20th Jun 2024.
- The evaluation metrics were used to judge the stock predictions on Nvidia from the time period of : 2024-01-02 to 2024-05-31.

Results

```
Sharpe Ratio: 1.02
Maximum Drawdown: -39.81%
Number of Trades: 13
Win Ratio: 92.31%

Portfolio Total Return: 6.01%
NVDA total return is: 4.48%
SP500 total return is: 127.62%

      Buy_Date   Sell_Date   Buy_Price  Sell_Price  Profit/Loss    Return  \
0   2024-01-08  2024-01-19  185.323517  191.315872     5.992355  0.032335
1   2024-02-01  2024-02-05  186.621872  187.440811     0.818939  0.004388
2   2024-02-08  2024-03-01  188.080017  179.660004    -8.420013 -0.044768
3   2024-02-21  2024-05-08  182.320007  182.740005     0.419998  0.002304
4   2024-02-26  2024-05-14  181.160004  187.429993     6.269989  0.034610
5   2024-03-07  2024-05-29  169.000000  190.289993    21.289993  0.125976
6   2024-03-12  2024-05-31  173.229996  192.250000    19.020004  0.109796
7   2024-03-21  2024-05-31  171.369995  192.250000    20.880005  0.121842
8   2024-03-27  2024-05-31  173.309998  192.250000    18.940002  0.109284
9   2024-04-18  2024-05-31  167.039993  192.250000    25.210007  0.150922
10  2024-04-29  2024-05-31  173.500000  192.250000    18.750000  0.108069
11  2024-05-10  2024-05-31  183.050003  192.250000     9.199997  0.050259
12  2024-05-22  2024-05-31  190.899994  192.250000     1.350006  0.007072

    Investment  Weighted_Return
0   185.323517         0.032335
1   186.621872         0.004388
2   188.080017        -0.044768
3   182.320007         0.002304
4   181.160004         0.034610
5   169.000000         0.125976
6   173.229996         0.109796
7   171.369995         0.121842
8   173.309998         0.109284
9   167.039993         0.150922
10  173.500000         0.108069
11  183.050003         0.050259
12  190.899994         0.007072
```

Final Trades Dataframe

Financial Dataframe:

bjectivity	Polarity	Compound	Negative	Neutral	Positive	Yesterday	Tomorrow	Cumu_Label(Up/Down)	Diff	Signal
05678	0.083816	-0.9946	0.174	0.746	0.079	184.015198	180.949097	0	0.0	Sell
49820	0.137964	-0.9981	0.196	0.748	0.056	181.678177	185.323517	0	0.0	Sell
85290	0.100302	-0.9972	0.176	0.782	0.042	180.949097	184.904053	1	1.0	Buy
47812	0.053901	-0.9971	0.181	0.761	0.058	185.323517	185.952713	0	-1.0	Sell
01869	0.108188	-0.9968	0.148	0.799	0.053	184.904053	185.353485	0	0.0	Sell
	...	...	...	...	...	...	...	...	...	...
25880	0.152737	-0.2874	0.115	0.768	0.117	191.289993	194.029999	0	2.0	Sell

	Buy_Date	Sell_Date	Buy_Price	Sell_Price	Profit/Loss	Return	Investment	Weighted_Return
0	2024-01-08	2024-01-19	185.323517	191.315872	5.992355	0.032335	185.323517	0.032335
1	2024-02-01	2024-02-05	186.621872	187.440811	0.818939	0.004388	186.621872	0.004388
2	2024-02-08	2024-03-01	188.080017	179.660004	-8.420013	-0.044768	188.080017	-0.044768
3	2024-02-21	2024-05-08	182.320007	182.740005	0.419998	0.002304	182.320007	0.002304
4	2024-02-26	2024-05-14	181.160004	187.429993	6.269989	0.034610	181.160004	0.034610
5	2024-03-07	2024-05-29	169.000000	190.289993	21.289993	0.125976	169.000000	0.125976
6	2024-03-12	2024-05-31	173.229996	192.250000	19.020004	0.109796	173.229996	0.109796
7	2024-03-21	2024-05-31	171.369995	192.250000	20.880005	0.121842	171.369995	0.121842
8	2024-03-27	2024-05-31	173.309998	192.250000	18.940002	0.109284	173.309998	0.109284
9	2024-04-18	2024-05-31	167.039993	192.250000	25.210007	0.150922	167.039993	0.150922
10	2024-04-29	2024-05-31	173.500000	192.250000	18.750000	0.108069	173.500000	0.108069
11	2024-05-10	2024-05-31	183.050003	192.250000	9.199997	0.050259	183.050003	0.050259
12	2024-05-22	2024-05-31	190.899994	192.250000	1.350006	0.007072	190.899994	0.007072

THANK

YOU