Project Report on

IntelliTrade Stock Price Prediction

as part of the Course

Applied Data Science

offered in the

Smart Internz Externship Programme

By

Team 409

Team members:
Vattem Karthik | 20BCE1627 | VIT Chennai |
vattem.karthik2020@vitstudent.ac.in
Anirudh Kaushik | 20BKT0038 | VIT Vellore |
anirudh.kaushik2020@vitstudent.ac.in
Amit rathore | 20BCE11115 | VIT Bhopal |
amit.rathore2020@vitbhopal.ac.in
Sahaj Dargan | 20BEC0737 | VIT Vellore |
sahaj.dargan2020@vitstudent.ac.in

Table of Contents

Problem Statement:

The project aims to address the lack of a comprehensive and integrated system that combines web scraping, data analysis, sentiment analysis, and investment suggestion generation. The existing tools are fragmented, lacking the ability to provide a holistic view of the stock market and limiting users' ability to make informed investment decisions. Therefore, the project seeks to develop a user-friendly platform that gathers and analyses historical data, extracts sentiment from news articles, and generates accurate investment suggestions based on real-time stock prices.

Abstract:

This project aims to develop a comprehensive web-based system that combines web scraping, data analysis, sentiment analysis, and investment suggestion generation for stock market analysis. The system collects historical data for multiple years for all stocks in the current Nifty 50 index using web scraping techniques. Data analysis techniques are applied to calculate daily log returns and remove influences caused by stock splits. LSTM models are trained to predict future stock prices based on historical patterns. Sentiment analysis is performed on news articles from reputable sources, assigning sentiment scores to stocks. The system generates investment suggestions by combining sentiment scores with user data, filtering stocks with positive rates of change. A user-friendly front-end interface provides access to estimated close prices and visualizations to aid decision-making. The project addresses the need for a comprehensive tool that integrates various techniques to support investors and traders in making informed investment decisions in the stock market.

Introduction:

The project aims to develop a comprehensive web-based system that combines various techniques such as web scraping, data analysis, sentiment analysis, and investment suggestion generation for stock market analysis. With the rapid advancement of technology and the increasing complexity of financial markets, there is a growing need for tools that can provide valuable insights and support decision-making in the stock market.

The stock market is influenced by a multitude of factors, including historical trends, market sentiment, and news events. By leveraging data analysis techniques and incorporating sentiment analysis of news articles, our project seeks to provide investors and traders with a holistic view of the market and actionable investment suggestions.

The project's foundation lies in web scraping, enabling the extraction of valuable data from reliable sources such as the NSE India platform. By collecting historical data for multiple years for all stocks in the current Nifty 50 index, we can build a comprehensive database that serves as the basis for analysis and prediction.

Data analysis is a crucial component of our system, as it enables us to derive meaningful insights from the collected data. By calculating daily log returns and applying cleaning techniques to eliminate influences from stock splits, we ensure accurate analysis and

prediction. Additionally, the use of LSTM models allows us to predict future stock prices based on historical patterns and trends.
Sentiment analysis plays a vital role in our project, as it provides valuable insights into market sentiment by analyzing news articles from reputable sources such as Business Today and Economic Times. By employing machine learning algorithms and preprocessing techniques, we can predict the sentiment of the articles and assign sentiment scores to individual stocks. This allows us to gauge market sentiment and its potential impact on stock prices.

The system's front-end interface, developed using Flask, provides users with an intuitive and interactive platform to access the analysis and recommendations. Users can view estimated close prices for specific stocks, analyze graphical representations of historical data and model performance, and make informed investment decisions based on the system's recommendations.

Our project aims to empower investors and traders with a comprehensive system that combines web scraping, data analysis, sentiment analysis, and investment suggestions. By leveraging historical data, sentiment analysis of news articles, and real-time stock prices, our system provides users with valuable insights to navigate the complex stock market and make informed investment decisions.

Literature Review:

Paper Title: "Stock Market Prediction Using LSTM Recurrent Neural Network" by Singh, S. K., et al. (2018)
In this paper, the authors propose using LSTM recurrent neural networks for stock market prediction. They demonstrate the effectiveness of LSTM models in capturing long-term dependencies in time series data, such as stock prices. The study provides insights into the application of LSTM models in stock market analysis and highlights the importance of considering historical trends for accurate predictions.

Paper Title: "Sentiment Analysis in Stock Market Prediction: A Review" by Zhang, Y., et al. (2019)
This review paper focuses on sentiment analysis in stock market prediction. It discusses the use of natural language processing techniques and machine learning algorithms to extract sentiment from news articles and social media data. The study highlights the potential impact of sentiment analysis on stock market predictions and provides an overview of different approaches and methodologies used in sentiment analysis for stock market analysis.

Paper Title: "Stock Market Prediction using Support Vector Machines" by Zou, L., et al. (2019)
The paper investigates the use of support vector machines (SVM) for stock market prediction. It explores the predictive power of SVM algorithms in analyzing stock price movements and identifies key features for accurate predictions. The study provides insights into the application of SVMs and their effectiveness in stock market analysis, emphasizing the importance of feature selection and model optimization.

Paper Title: "Web Scraping and Data Extraction Techniques: A Survey" by Gohil, S., et al. (2020)

This survey paper provides an overview of web scraping techniques for data extraction. It discusses various approaches, tools, and challenges associated with web scraping, highlighting its importance in gathering data for stock market analysis. The study explores different methods for handling dynamic web content, dealing with anti-scraping mechanisms, and ensuring data quality.

Paper Title: "Visualization Techniques for Financial Data Analysis: A Review" by Li, H., et al. (2017)
This review paper focuses on visualization techniques for financial data analysis. It discusses the importance of visualizations in understanding and interpreting complex financial data, including stock market data. The study provides an overview of different visualization methods and tools, highlighting their strengths and limitations. It emphasizes the significance of effective visualizations in aiding decision-making and improving user understanding of financial data.

Methodology:

1. Data Collection:

- Web scraping techniques are employed to collect historical data for multiple years for all stocks in the Nifty 50 index from the NSE India platform. This data includes stock prices, volume, and other relevant information.

2. Data Preprocessing:

- The collected data is subjected to preprocessing steps to ensure its quality and suitability for analysis.
- Inconsistencies and outliers are identified and removed from the data to maintain data integrity.
- Daily log returns are calculated based on the closing prices of stocks. Log returns capture the percentage change in price over time and provide a measure of relative performance.
- Adjustments are made to account for stock splits, which can introduce biases in the data. By removing the influence of such events, the model focuses on the actual price movement of the stocks.

3. LSTM Model Training:

- LSTM (Long Short-Term Memory) models are built for each of the 50 stocks in the Nifty 50 index. LSTM models are a type of recurrent neural network (RNN) that can capture long-term dependencies in time series data.
- The historical data is pre-processed to remove large deviations due to stock splits/bonuses, the daily log return which is calculated using the close price of the stock is then used to train the LSTM models. The models learn from past patterns and relationships in the data to make predictions about future stock prices.
- Hyperparameters of the LSTM models are tuned and the model architecture is adjusted as needed to optimize their performance.
- Model performance is evaluated using MAE, RMSE and residual plots

4. Sentiment Analysis:

- News articles are scraped from reputable sources such as Business Today and Economic Times. These articles contain information that can influence stock market sentiment.
- The scraped news articles undergo preprocessing steps, including the removal of stop words, stemming, and tokenization. These steps help extract the core meaning from the text.
- Machine learning algorithms, such as Logistic Regression, Naive Bayes, SVM, etc., are applied for sentiment analysis on the pre-processed news articles. These algorithms classify the sentiment of the articles as positive, negative, or neutral.
- Sentiment scores are assigned to individual stocks based on the sentiment analysis results. These scores reflect the market sentiment associated with each stock.

5. Integration and Recommendation:

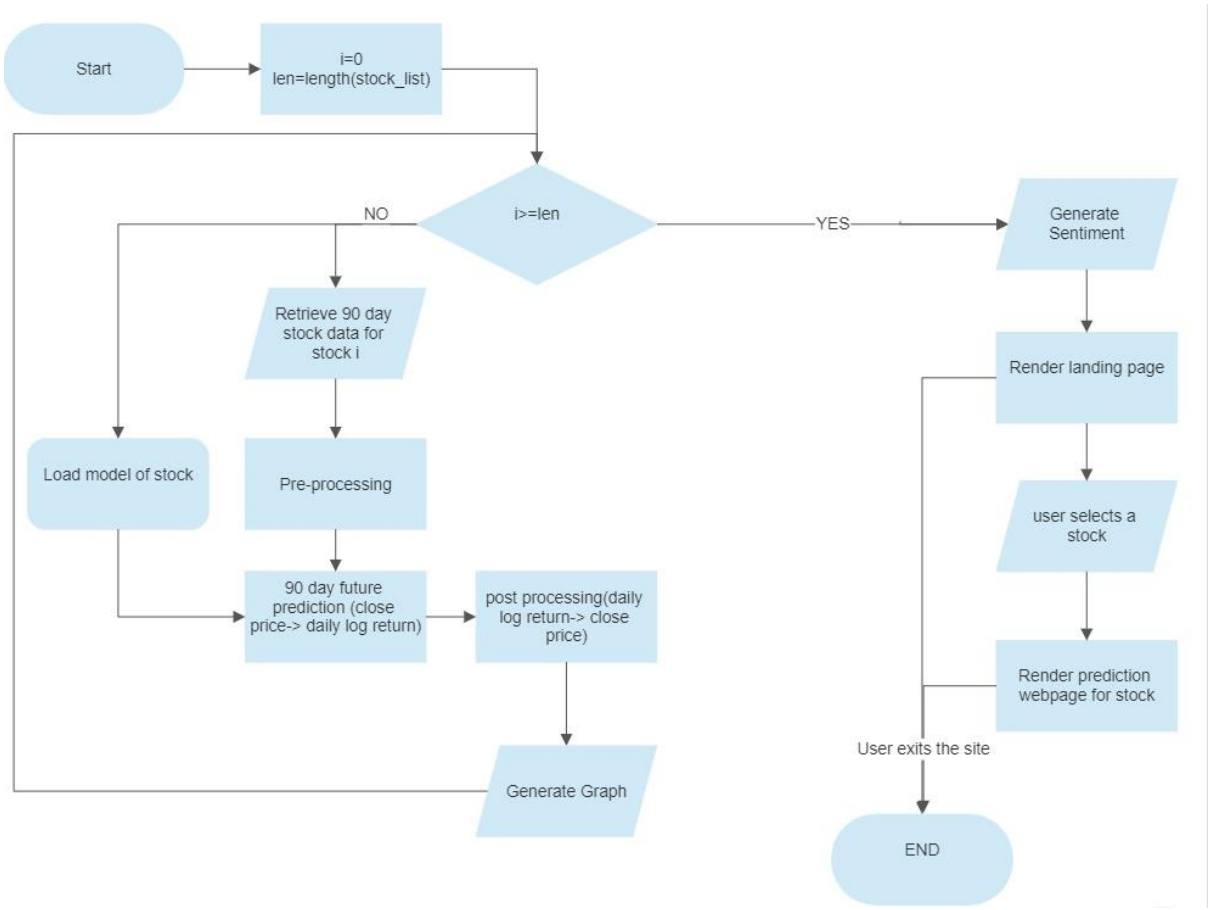
- The sentiment scores from news articles are combined with user input data, including current stock prices and rates of change.
- Stocks with positive rates of change are filtered, indicating stocks with upward price momentum.
- The filtered stocks are ranked based on their potential profitability, considering factors such as sentiment scores and historical performance.
- Investment suggestions are generated based on the filtered and ranked stocks, taking into account the available deposit amount.
- The system presents these investment suggestions to the user.

6. Front-End Development:

- A web-based front-end interface is developed using Flask, a Python web framework.
- The interface allows users to access the system and interact with its features.
- Estimated close prices for specific stocks are displayed based on the 90 most recent trading days' data, providing insights into the predicted future prices for 90 days into the future when the app is run by integrating with a web scraper.
- Various graphical representations, such as charts and graphs, are incorporated to inform users about model performance, sentiment analysis results, and other relevant information.

Methodology Diagrams:

Flask Application Flowchart:



Experimental Results & Discussions

1. Sentiment analysis

The dataset contains, the company name, the news regarding that particular company and, sentiment of the news either positive or negative sentiment. Over here, for pre-processing, we remove the stop words, & perform vectorisation on the text (news). Then, we convert the matrix into TF-IDF matrix using the TfidfTransformer class in sklearn package. Then, we have applied six algorithms for the data, they are Multilinear regression, Decision tree, Random Forest, Naive Bayes algorithm and SVM classifier, K-Nearest Neighbour algorithms

Naive Bayes Classification

The supervised learning algorithm known as the Naive Bayes, which is based on the Bayes theorem, is used to resolve classification issues. It provides predictions based on the likelihood that an object will occur because it is a probabilistic classifier. It depends on the conditional probability.
Here we were able to predict the data 97.64% accurately for the text data.

```
# Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train_features, y_train)
nb_pred = nb_model.predict(X_test_features)
nb_accuracy = accuracy_score(y_test, nb_pred)
print("Naive Bayes Accuracy:", nb_accuracy)
metrics.confusion_matrix(y_test,nb_pred)
print(metrics.classification_report(y_test,nb_pred))
```

```
Naive Bayes Accuracy: 0.9764150943396226
              precision    recall  f1-score   support

    Negative       0.97      0.98      0.98       110
    Positive       0.98      0.97      0.98       102

    accuracy                           0.98       212
   macro avg       0.98      0.98      0.98       212
weighted avg       0.98      0.98      0.98       212
```

SVM Classifier

In order to swiftly categories new data points in the future, the SVM algorithm aims to determine the optimum line or decision boundary that can divide n-dimensional space into classes. The name of this best decision boundary is a hyperplane. The extreme vectors and points that help create the hyperplane are chosen via SVM.
Using the linear kernel for SVM classification, we get accuracy of 99.52% for the text data.

```
# Support Vector Machine
svm_model = SVC()
svm_model.fit(X_train_features, y_train)
svm_pred = svm_model.predict(X_test_features)
svm_accuracy = accuracy_score(y_test, svm_pred)
print("Support Vector Machine Accuracy:", svm_accuracy)
metrics.confusion_matrix(y_test,svm_pred)
print(metrics.classification_report(y_test,svm_pred))
```

```
Support Vector Machine Accuracy: 0.9952830188679245
              precision    recall  f1-score   support

    Negative       0.99      1.00      1.00       110
    Positive       1.00      0.99      1.00       102

    accuracy                           1.00       212
   macro avg       1.00      1.00      1.00       212
weighted avg       1.00      1.00      1.00       212
```

Logistic Regression

Logistic regression is a statistical method used for binary classification, which means predicting the outcome of a binary variable based on one or more predictor variables.
The accuracy obtained for logistic regression is 99.52% for the text data.

```
# Logistic Regression
logreg_model = LogisticRegression()
logreg_model.fit(X_train_features, y_train)
logreg_pred = logreg_model.predict(X_test_features)
logreg_accuracy = accuracy_score(y_test, logreg_pred)
print("Logistic Regression Accuracy:", logreg_accuracy)
metrics.confusion_matrix(y_test,logreg_pred)
print(metrics.classification_report(y_test,logreg_pred))
```

```
Logistic Regression Accuracy: 0.9952830188679245
              precision    recall  f1-score   support

    Negative       1.00      0.99      1.00       110
    Positive       0.99      1.00      1.00       102

    accuracy                           1.00       212
   macro avg       1.00      1.00      1.00       212
weighted avg       1.00      1.00      1.00       212
```

Decision Tree Algorithm

A decision tree is a supervised machine learning model used for classification and regression. Each internal node in this structure, which resembles a flowchart, stands for a test on an attribute.
The accuracy obtained by this technique is 98.11% for the text data.

```
# Decision Tree
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train_features, y_train)
dt_pred = dt_model.predict(X_test_features)
dt_accuracy = accuracy_score(y_test, dt_pred)
print("Decision Tree Accuracy:", dt_accuracy)
metrics.confusion_matrix(y_test,dt_pred)
print(metrics.classification_report(y_test,dt_pred))
```

```
Decision Tree Accuracy: 0.9811320754716981
              precision    recall  f1-score   support

    Negative       0.99      0.97      0.98       110
    Positive       0.97      0.99      0.98       102

    accuracy                           0.98       212
   macro avg       0.98      0.98      0.98       212
weighted avg       0.98      0.98      0.98       212
```

Random Forest Algorithm

Random Forest Classifier is a combines the predictions of multiple decision trees to improve accuracy and mitigate overfitting.
By this algorithm we achieved 99.05% accuracy for the text data.

```
# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train_features, y_train)
rf_pred = rf_model.predict(X_test_features)
rf_accuracy = accuracy_score(y_test, rf_pred)
print("Random Forest Accuracy:", rf_accuracy)
metrics.confusion_matrix(y_test,rf_pred)
print(metrics.classification_report(y_test,rf_pred))
```

```
Random Forest Accuracy: 0.9905660377358491
              precision    recall  f1-score   support

    Negative       0.98      1.00      0.99       110
    Positive       1.00      0.98      0.99       102

    accuracy                           0.99       212
   macro avg       0.99      0.99      0.99       212
weighted avg       0.99      0.99      0.99       212
```

K-Nearest Neighbours Algorithm

K-Nearest Neighbours (KNN) algorithm is a non-parametric classification algorithm that makes predictions based on the majority class of its k nearest neighbors in the feature space.
By this algorithm we achieved 94.81% accuracy for the text data.

```
# K-Nearest Neighbors
knn_model = KNeighborsClassifier()
knn_model.fit(X_train_features, y_train)
knn_pred = knn_model.predict(X_test_features)
knn_accuracy = accuracy_score(y_test, knn_pred)
print("K-Nearest Neighbors Accuracy:", knn_accuracy)
metrics.confusion_matrix(y_test,knn_pred)
print(metrics.classification_report(y_test,knn_pred))
```

```
K-Nearest Neighbors Accuracy: 0.9481132075471698
              precision    recall  f1-score   support

    Negative       0.94      0.96      0.95       110
    Positive       0.96      0.93      0.95       102

    accuracy                           0.95       212
   macro avg       0.95      0.95      0.95       212
weighted avg       0.95      0.95      0.95       212
```

Comparison of Results & Interpretation

By executing all the algorithms, we have listed down all the accuracies of each algorithm and is displayed below:

S.no.	Algorithm	Accuracy (in percentage)
1.	Naive Bayes Algorithm	97.76
2.	SVM Linear Classifier	99.52
3.	Logistic Regression Algorithm	99.52
4.	Decision Tree Algorithm	98.11
5.	Random Forest Algorithm	99.05
6.	K – Nearest Neighbor Algorithm	94.81

2. LSTM model performance metrics

Mean Absolute Error (MAE):

Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. It is calculated by taking the mean value of the absolute difference between the predicted and actual values of the response variable.
MAE is equally sensitive to outliers as compared to non-outliers.

Root Mean Square Error (RMSE):

Root-Mean-Square Error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. It is calculated by taking the root of the mean value of difference between the square of the actual value and the square of the predicted value of the response variable.
RMSE is more sensitive to outliers.

Residual plots:

A residual plot is a graph in which the residuals are displayed on the y axis and the independent variable is displayed on the x-axis. A linear regression model is appropriate for the data if the dots in a residual plot are randomly distributed across the horizontal axis.

The values of MAE, RMSE were calculated for each stock and the residual plots for both train and test datasets of the respective stock model were plotted.

3. Investment Recommendations: Maximizing Returns

We have implemented a robust algorithm that takes into account the available amount and stock prices to suggest an optimized allocation strategy. By leveraging historical stock data and employing advanced calculations, our system identifies stocks with potential for growth and recommends investing in them.

```
Please kindly enter the desired investment amount for your consideration: 7000
Available Amount: 7000
Invest 4.0 stocks in ADANIENT at a price of 1464.8849400000001
Invest 1.0 stocks in INDUSINDBK at a price of 781.0832999999999
Invest 3.0 stocks in NTPC at a price of 112.748256
Leftover amount: 21.132171999999514
```

```
Please kindly enter the desired investment amount for your consideration: 30000
Available Amount: 30000
Invest 2.0 stocks in NESTLEIND at a price of 13806.2274
Invest 1.0 stocks in ADANIENT at a price of 1464.8849400000001
Invest 1.0 stocks in INDUSINDBK at a price of 781.0832999999999
Invest 1.0 stocks in NTPC at a price of 112.748256
Leftover amount: 28.828704000000442
```

Interpretation:

The Naive Bayes Algorithm achieved an accuracy of 97.76%. This suggests that it is effective in making accurate predictions on the dataset, indicating a good ability to capture the underlying patterns and relationships.

Both the SVM Linear Classifier and Logistic Regression Algorithm demonstrated an impressive accuracy of 99.52%. These algorithms leverage linear decision boundaries to classify the data, and their high accuracy indicates their strong discriminatory power.

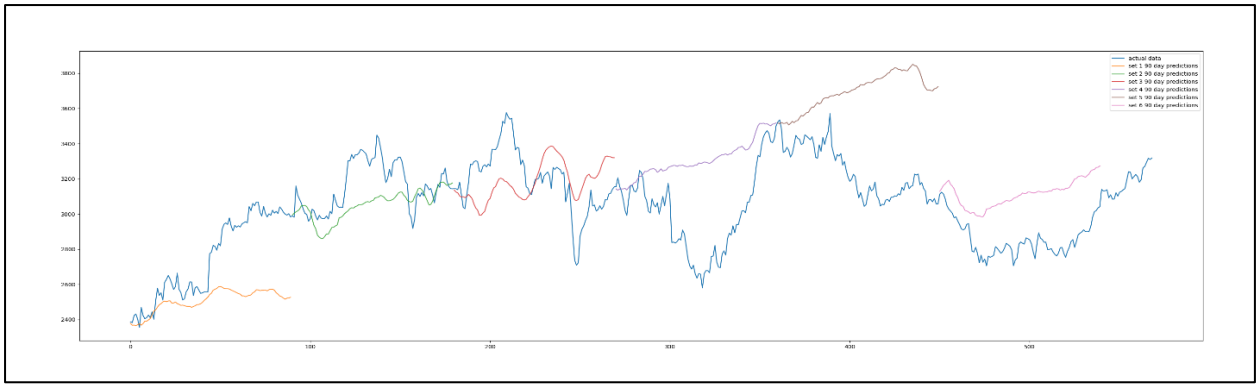
The Decision Tree Algorithm achieved an accuracy of 98.11%. This result suggests that decision trees were able to capture complex relationships within the dataset and make accurate predictions.

The Random Forest Algorithm achieved an accuracy of 99.05%. By combining multiple decision trees, the Random Forest Algorithm improves the accuracy and robustness of predictions, as reflected in this high accuracy result.

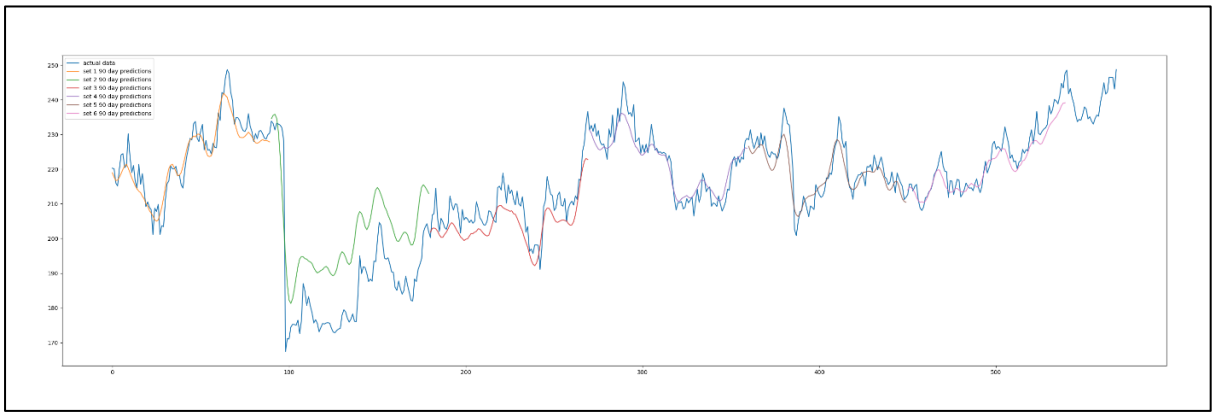
The K-Nearest Neighbor Algorithm achieved an accuracy of 94.81%. While this algorithm is intuitive and simple to implement, it exhibited relatively lower accuracy compared to other algorithms in this study. This may indicate that the dataset's characteristics or the choice of parameters for KNN could potentially impact its predictive performance.

The model performed quite well with predicting price movements for some stocks and poorly for some others; For example,

The model performed quite poorly on some stocks such as ASIANPAINT whose Share Value is around 3016, with an MAE of 131.46 and an RMSE of 157.61.



From the above image, one can infer that the model fails at predicting the major movements of the stock(blue)



The model performs very well on some stocks such as POWERGRID whose Share Value is around 210 with an MAE of 3.17 and an RMSE of 3.78.

From the above image, one can notice that the model performed well at predicting the major price movements(blue). The large dip in price that is present is due to a stock split; this data is not fed into the model and is imputed for as the rapid decline of stock price due to a split is not indicative of actual market interest in the stock and therefore introduces a difference between the 90 day predicted value(green) and the actual value(blue).

From this varying set of results, one can infer that daily log returns are not enough to predict the 90 day future close prices using a LSTM model; more data such as number of trades, sentiment news is required for a more robust prediction for some of the stocks.

Appendix

GitHub repo on top of which the NSE web scraper was developed: https://github.com/Sampad-Hegde/NSE-India-Web-Scraping

Demonstration Video Link: https://www.youtube.com/playlist?list=PL3XFXiYp65qwdWj0CNAF-0eb-hL_TWOOq

Conclusion:

This project report explored the application of sentiment analysis and LSTM networks in stock market prediction. By leveraging sentiment analysis techniques, we extracted valuable insights from social media data and news articles to capture the market sentiment surrounding current nifty 50 stocks. Using a web scraper, daily log returns of the stocks were calculated and fed into a LSTM network which predicted 90-day future close prices. However, it is important to note that stock market prediction is a challenging task due to the inherent complexity, volatility, and randomness of financial markets. While sentiment analysis and daily log returns alone can provide valuable insights, it is also required to look at other factors surrounding the stock such as its sector, volatility, activities taken by the company whose stock is released publicly etc. for more accurate and robust predictions.