DA 231o: Data Engineering at Scale

Stock Market Analysis and Prediction

The Bull Runners

Sandeep Pulavarthi : sandeeppulav@iisc.ac.in
S Varun : varuns1@iisc.ac.in
Arun Kumar A : aruna@iisc.ac.in
Akhzar Farhan : akhzarfarhan@iisc.ac.in

Problem Definition

Stock price prediction is a complex and challenging task due to the inherent volatility and unpredictability of the stock market.

Our Approach

- This project aims to predict future stock prices or trends using historical stock prices, trading volumes, and related features, a machine learning model will forecast the next day's closing price or future price movements.
- The model is built with PySpark to efficiently process large datasets, ensuring scalability and fast training.

Key Factors

1. Handling Large Datasets
2. Improved Accuracy Over Traditional Methods
3. Forecasting Stock Trends and Market Behavior
4. Scalability and Parallel Processing

The Dataset

- Historical daily price of NIFTY 100 (Top 100 Indian Stocks data) from Jan 2015 to Feb 2022.
- 55 technical indicators.
- Data is recorded for every 5 minutes timestamp.

Data Preprocessing

- Removed rows with null values.
- Discarded columns with no correlation with our target column.

Features Required

- Open Price: The price of the stock at the beginning of 5 min window.
- Close Price: The price of the stock at the end of 5 min window.
- High Price: The highest price the stock reached during a 5 min window.
- Low Price: The lowest price the stock reached during a 5 min window.
- SMA 20: SImple moving average of the stock in the last 20 days.

Data Visualization

Feature Engineering

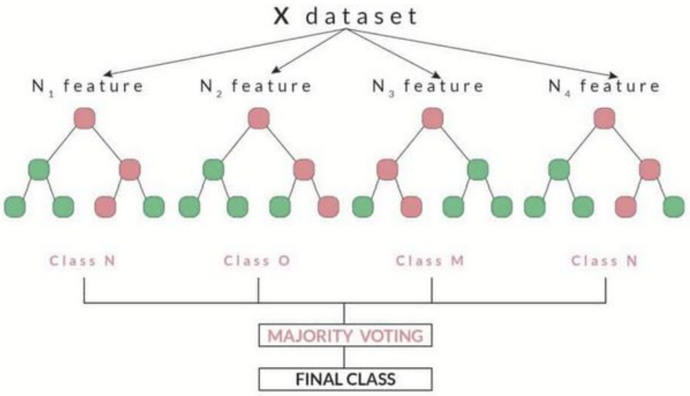
- For Random Regressor: Created five lag features using the closing price of last 5 data points. These lag features captured the temporal dependencies in the stock's price movements.
- For ARIMA: The data was aggregated to a daily timeframe, where rolling mean and rolling standard deviation were computed over a defined window. This process smooths out short-term fluctuations and capture underlying trend and volatility of the data.
- Train-Test Split: 80% and 20% split was performed.
- Dataset Fractions: [0.1, 0.2, 0.5, 0.8, 1.0] fractions of data was taken for training and evaluation purposes.

Random Forest

Random forest regression is a supervised machine learning algorithm. It uses an ensemble of decision trees to predict continuous target variables

Why?

- Handles non-linear patterns in stock prices
- Scalable and efficient with large datasets
- Generalizes well to unseen data

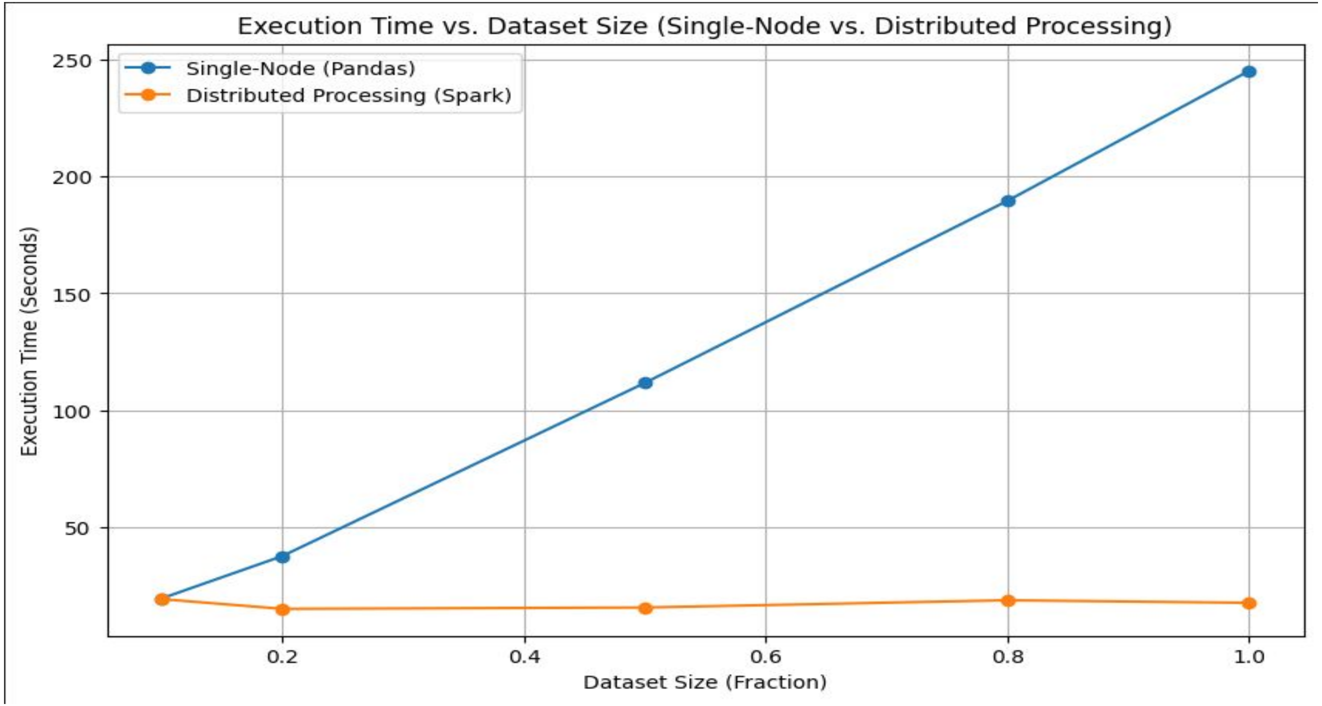


Random Forest Machine Learning Methods

1. Distributed Computing (actual model):
   a. Algorithm: Random Forest from PySpark's MLlib
   b. Data Storage: Stored and processed using PySpark dataframes
2. Single Node (for comparison):
   a. Algorithm: Random Forest from scikit-learn
   b. Data Storage: Stored and processed using Python Pandas dataframes

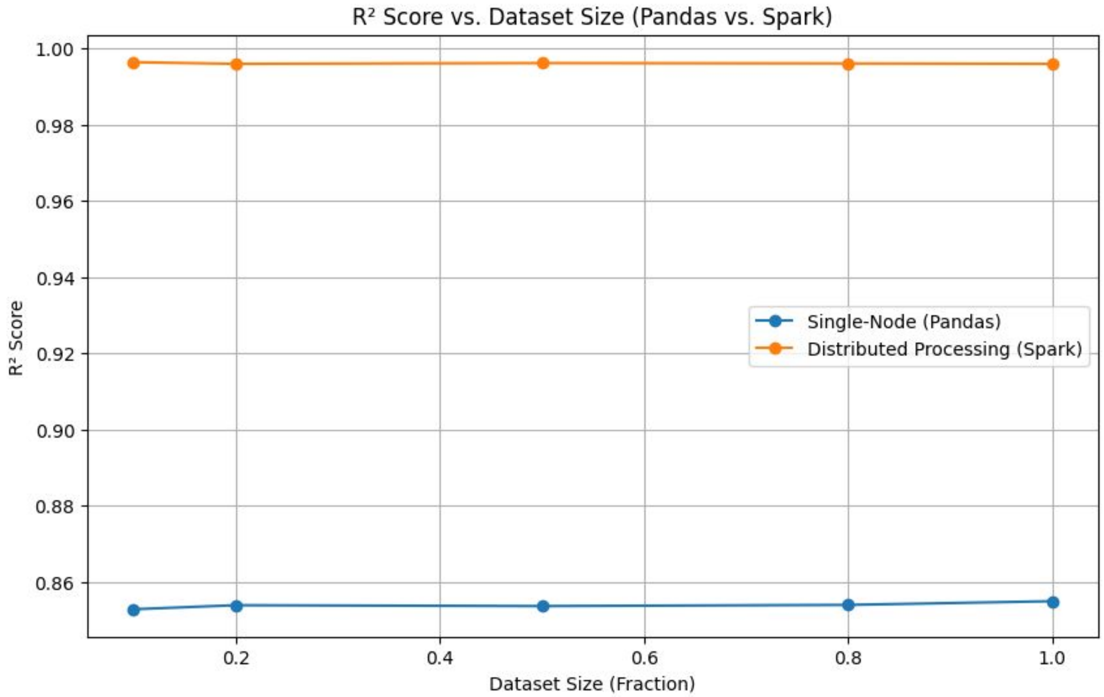
Scalability and Parallelization

Plot 1: Execution Time vs Dataset Size (Single-Node vs Distributed Processing)



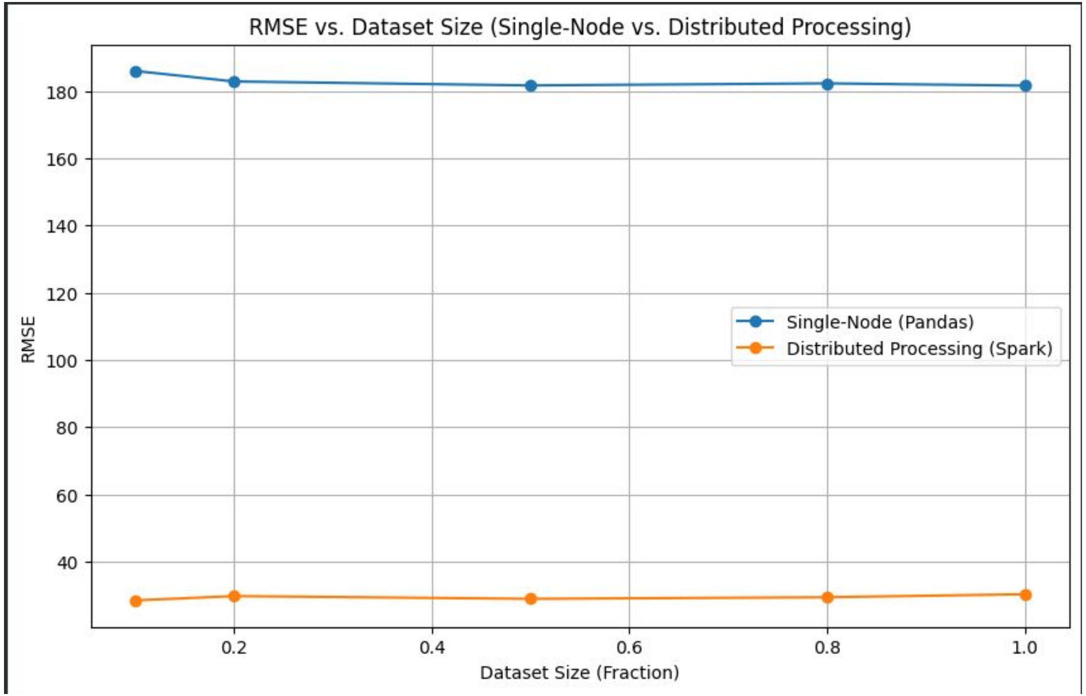
Scalability and Parallelization

Plot 2: R2 Score vs Dataset Size (Single-Node vs Distributed Processing)



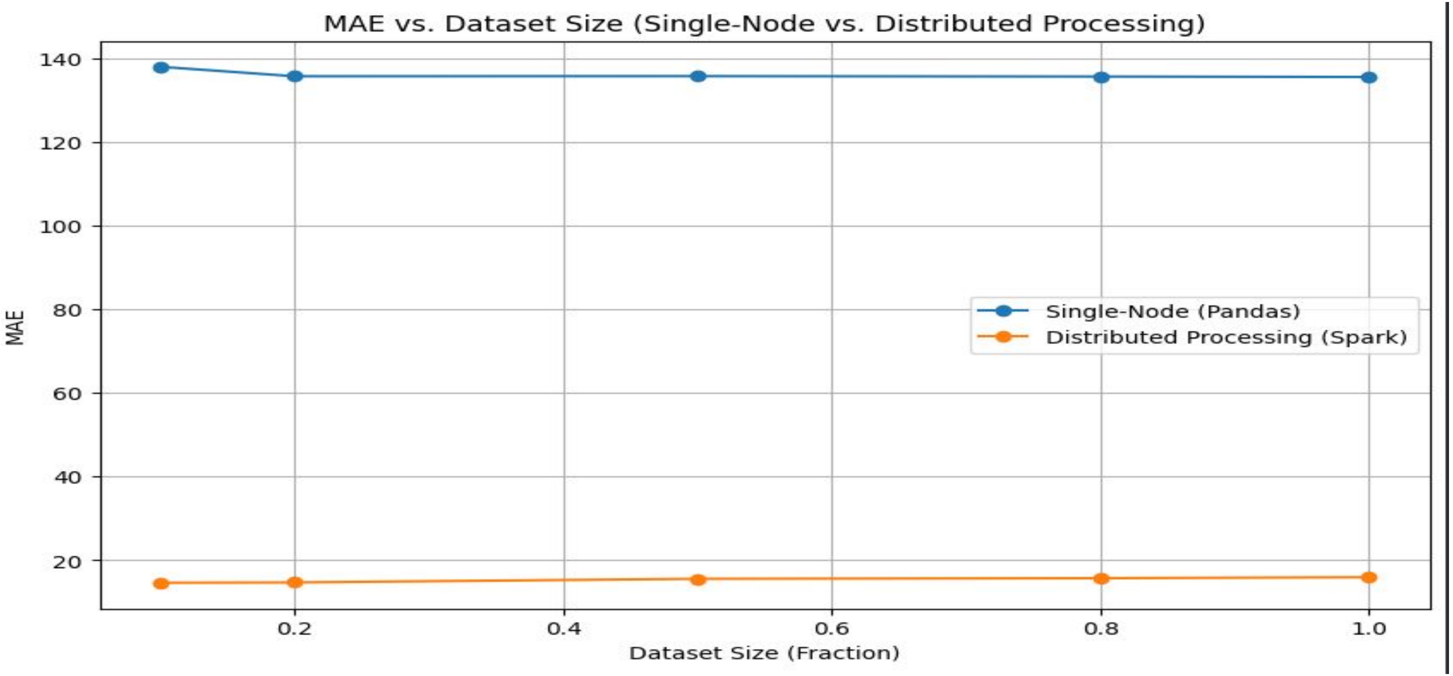
Scalability and Parallelization

Plot 3: RMSE vs Dataset Size (Single-Node vs Distributed Processing)



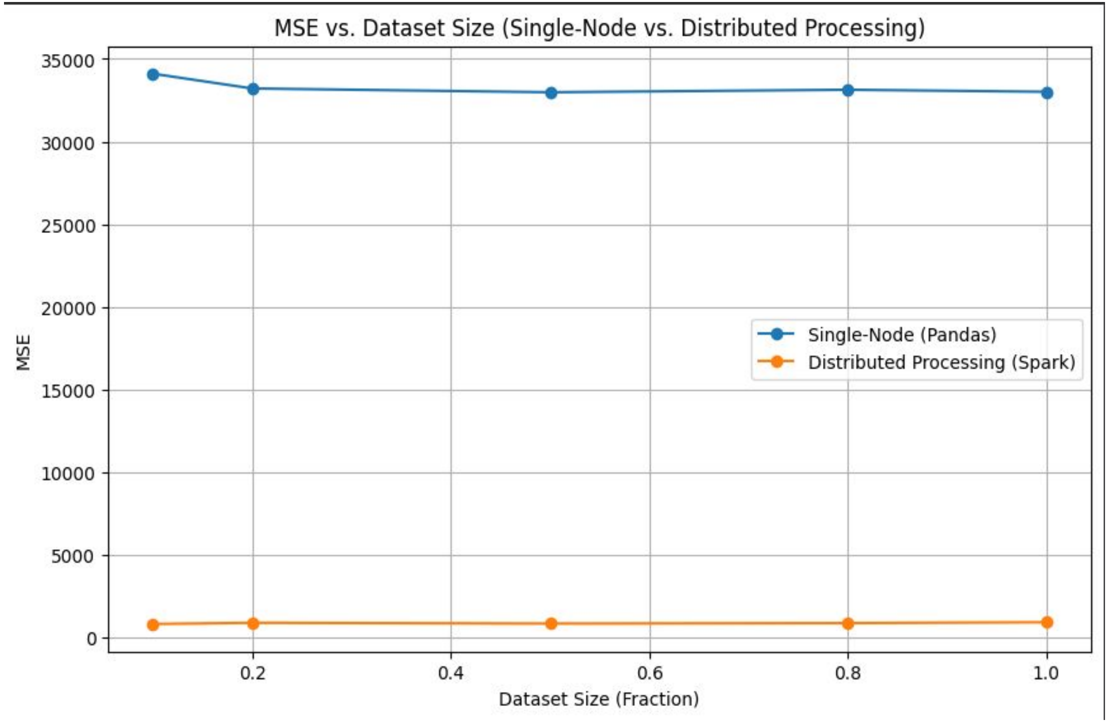
Scalability and Parallelization

Plot 4: MAE vs Dataset Size (Single-Node vs Distributed Processing)



Scalability and Parallelization

Plot 5: MSE vs Dataset Size (Single-Node vs Distributed Processing)



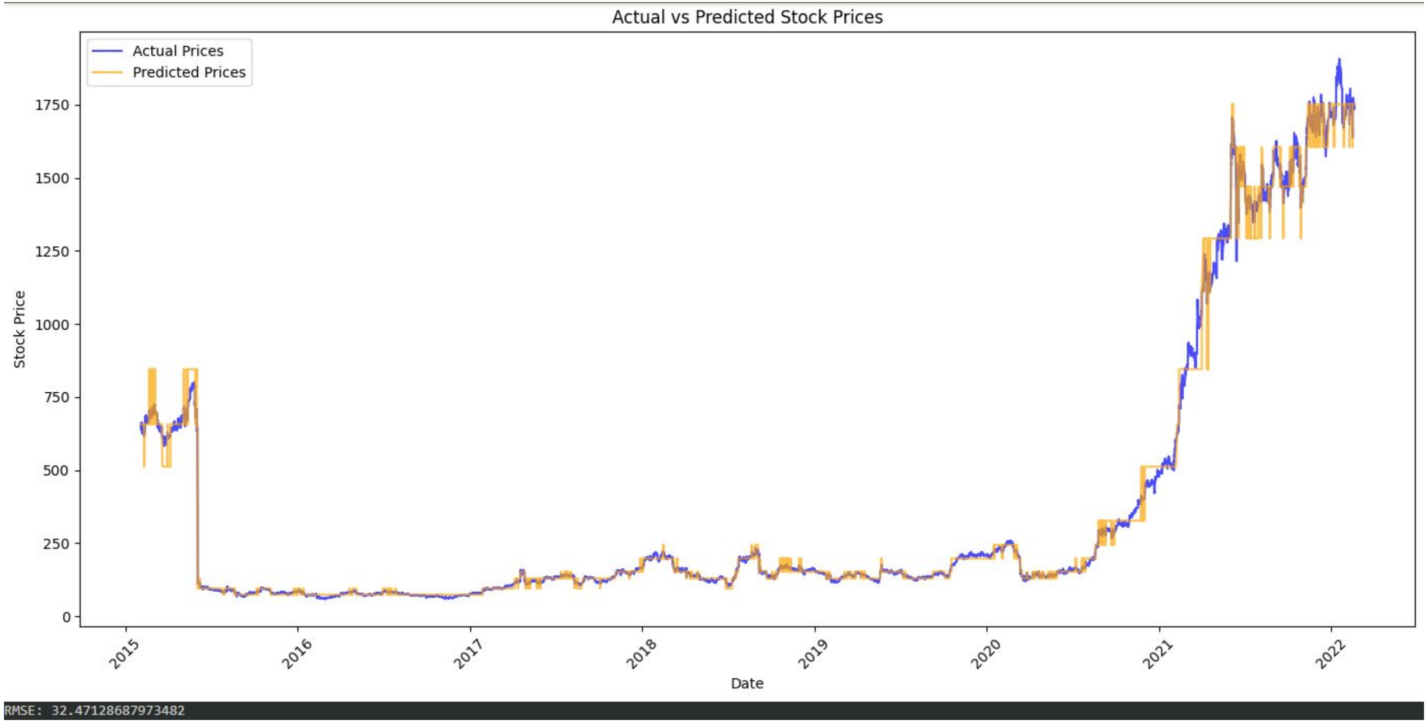
Performance Metrics

Random Forest

Processing type	Execution time	R-squared	RMSE	MAE	MSE
Single Node	~250 sec	~0.86	~180	~130	~35000
Distributed	~40 sec	~0.99	~40	~10	~1000

Result

Plot 7: Actual Stock Price vs Predicted Stock prices for the Test Data



ARIMA (AutoRegressive Integrated Moving Average)

AutoRegression (AR):

- Relies on the relationship between an observation and a certain number of lagged observations (previous values).
- Represented by the parameter *p*, which indicates the number of lagged observations included in the model.

Integration (I):

- Represents the differencing of the data to make the time series stationary (i.e., to remove trends or seasonality).
- Represented by the parameter *d*, which is the number of differencing operations applied.

Moving Average (MA):

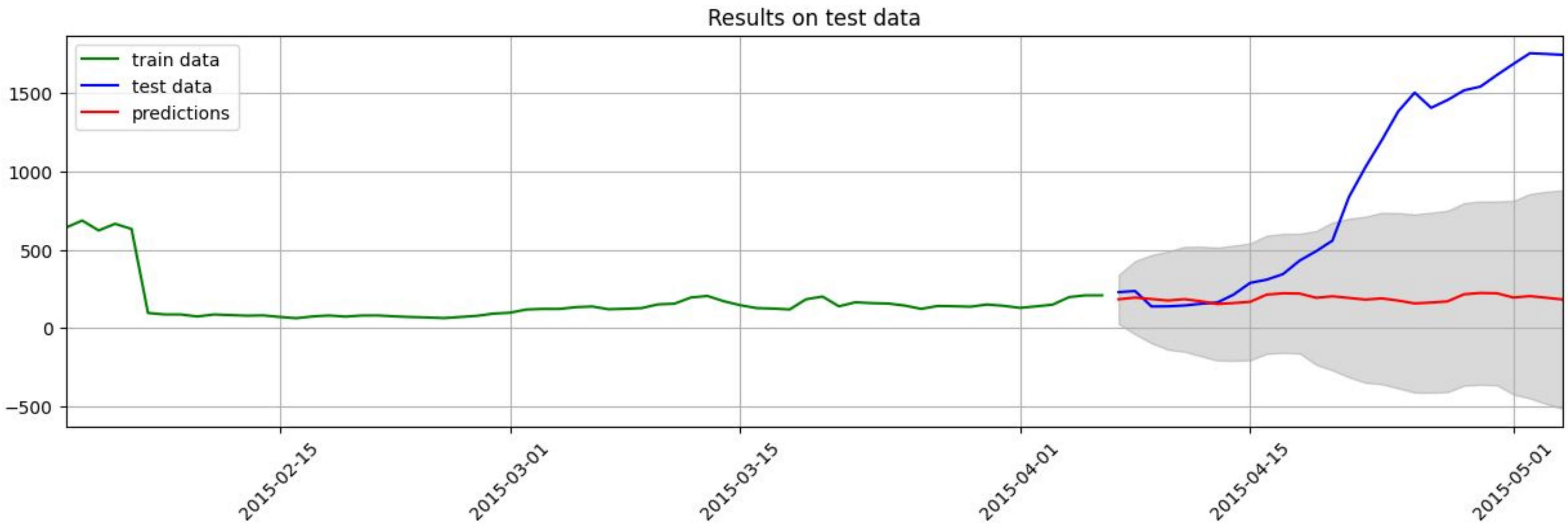
- Models the relationship between an observation and a residual error from a moving average model applied to lagged observations.
- Represented by the parameter *q*, which indicates the size of the moving average window.

ARIMA Machine Learning Methods

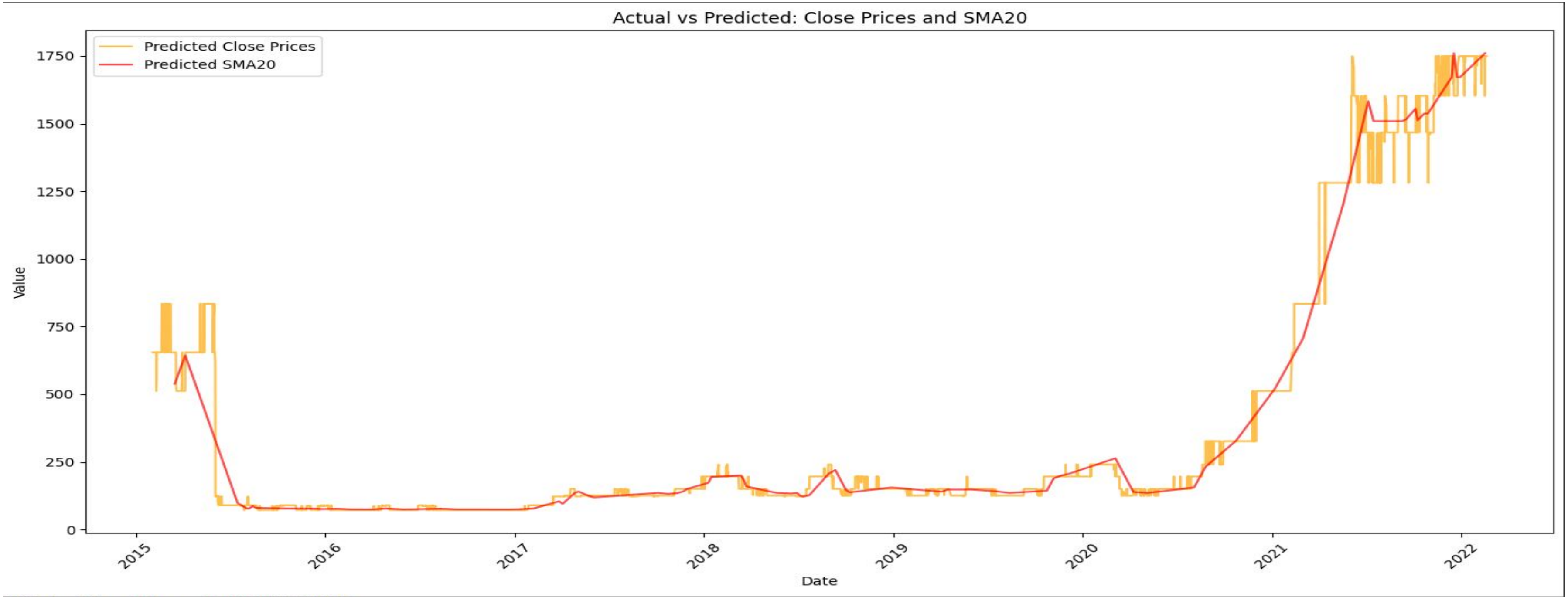
1. Distributed Computing model:
   a. Algorithm: Auto Arima from StatsForecast and FugueBackend libraries.
   b. Data Storage: Stored and processed using PySpark dataframes in a grouped structure.
2. Single Node model
   a. Algorithm: Auto Arima from pmdarima
   b. Data Storage: Stored and processed using Python Pandas dataframes

ARIMA Predictions

Plot 6: Predictions on ARIMA model



Stock Buy/Sell Recommendation




```
RMSE for Close Prices: 32.18982963155974
RMSE for SMA20: 38.43535182212123
```

Future Enhancements

- Implement LSTM model using PySpark
- Train the model on more stocks data to improve model's accuracy.
- Deploy the model with user interface.
- Real time data integration.
- Addition of alternative data such as sentiment analysis and macroeconomic factors to enhance model predictions.

References

Dataset: https://www.kaggle.com/datasets/debashis74017/stock-market-data-nifty-100-stocks-5-min-data/data

Arima model: https://pypi.org/project/pmdarima/

Random forest: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.regression.RandomForestRegressor.html

Thank You