Microsoft Stock Price Prediction

Introduction

- The stock market is a complex and dynamic environment where shares of public companies are traded. It is a place where buyers and sellers come together to exchange ownership of stocks or other financial assets.
- One of the biggest challenges in navigating the stock market is the uncertainty and volatility that comes with it.
- The prices of stocks can fluctuate rapidly due to various factors, such as changes in the economy, political instability, company news, or even social media trends.
- The application of data mining and machine learning (ML) in stock price prediction can help investors and traders navigate the complex and volatile stock market with more confidence and accuracy.
- In this research, I will use some ML algorithms to predict the closing price of Microsoft stocks.
- The ML algorithms used in the experiment are Linear Regression(LR), Support Vector Regressor (SVR), Random Forest Regressor (RFR), Gradient Boosting Regressor (GBR), KNeighbors Regressor (KNR), and Deep Neural Networks (DNNs).
- The tools that have used in the implementation are Python, Scikit-Learn, Keras, Pandas, NumPy, Matplotlib, GridSearchCV, and Keras Tuner.

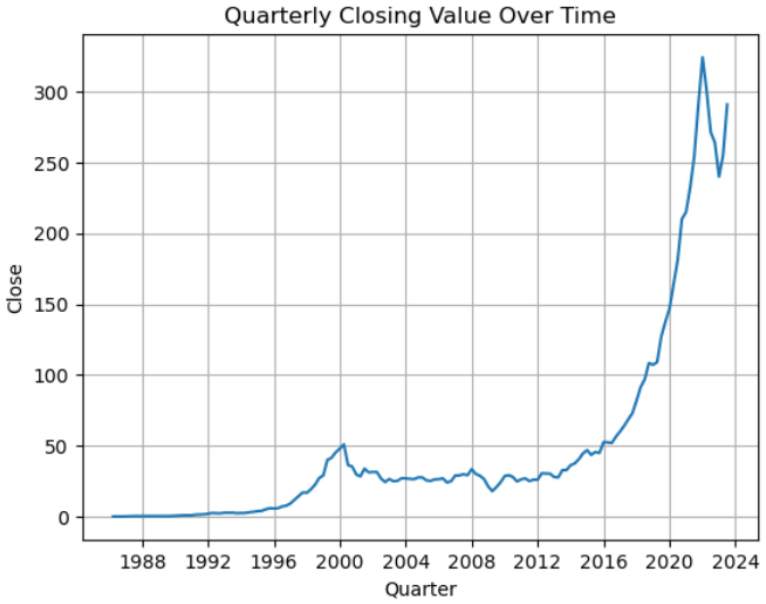
Data Collection

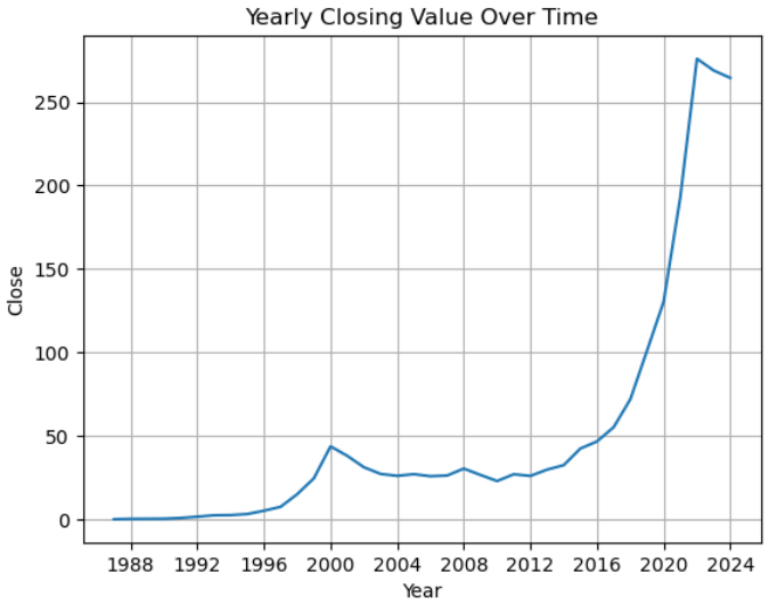
- Microsoft stock price data is collected from Yahoo Finance website (https://finance.yahoo.com/quote/MSFT/history/)
- The dataset has stock data from March 1989 to May 2023
- It includes 9361 rows and 7 columns

	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.060657	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.062823	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.063907	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.062281	67766400
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.061198	47894400
...	...	...	...	...	...	...	...
9356	2023-04-27	295.970001	305.200012	295.250000	304.829987	304.829987	46462600
9357	2023-04-28	304.010010	308.929993	303.309998	307.260010	307.260010	36446700
9358	2023-05-01	306.970001	308.600006	305.149994	305.559998	305.559998	21294100
9359	2023-05-02	307.760010	309.179993	303.910004	305.410004	305.410004	26404400
9360	2023-05-03	306.619995	308.609985	304.089996	304.399994	304.399994	22303300

9361 rows × 7 columns

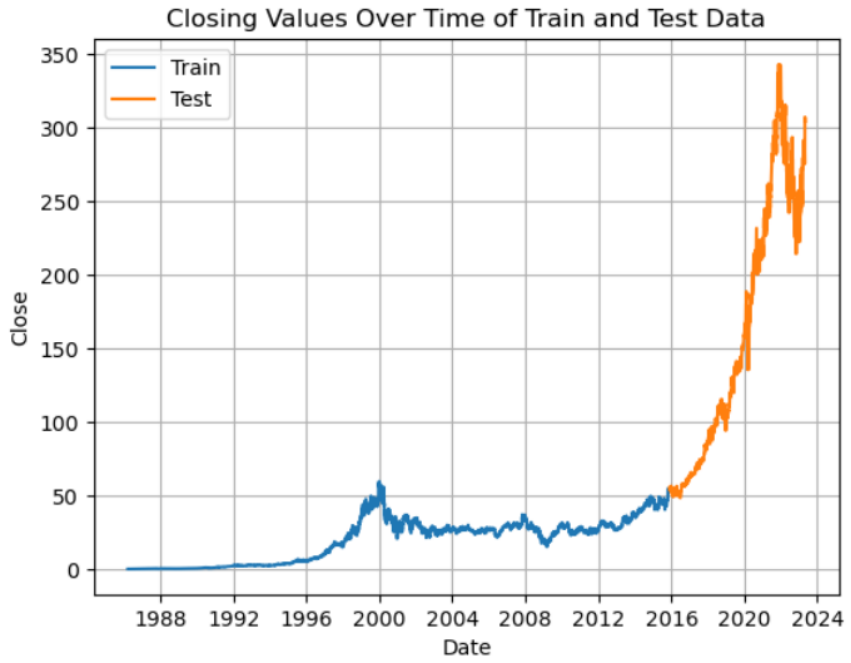
Data Exploration





Data Preprocessing

- Data cleaning
  - Check missing values for Date and Close columns
  - Remove redundant columns
  - Convert date string to datetime data type and set as index
- Feature engineering
  - Create windows by using different five lagged variables based on closing price
- Splitting data into train, validation and test sets
  - 80 percent of data has been used for train set and 20 percent for test set
  - For DNNs, 20 percent of train data has been used as validation set
- Normalization
  - Standardize all features



Model Selection

- Searching hyperparameters of LR, SVR, RFR, GBR, KNR and DNNs using different lagged values (ranged from 1 to 5) as features by the help of GridSearchCV, Keras tuner and time series split
- Comparing the performance of the models by using mean absolute error (MAE)
- In addition to proposed models, we can also use other machine learning and deep learning models

Model Training and Evaluation

- For LR, SVR, RFR, GBR, KNR, I used time series split to split the train data into train and validation sets. For DNNs, I split the data into train, validation and test sets.
- The best parameters got from hyperparameters tuning has been set to the models
- Mean absolute error (MAE), mean squared error (MSE), Performance metrics have been used to evaluate the model

Model Training and Evaluation (Cont...)

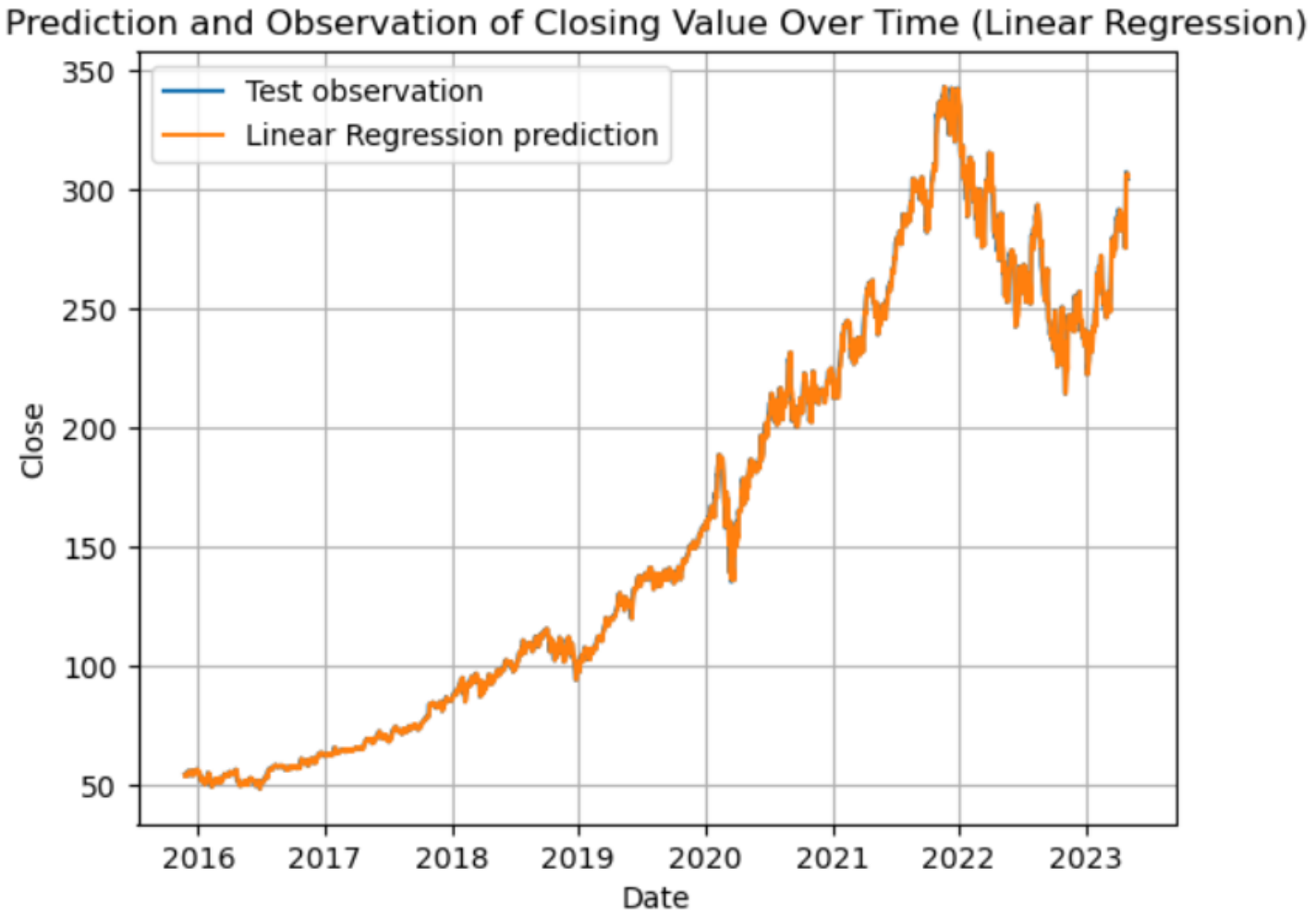
Parameters (Lag 1)	Parameters (Lag 2)	Parameters (Lag 3)	Parameters (Lag 4)	Parameters (Lag 5)
Linear Regression fit_intercept='True' **Support Vector Regressor** kernel='linear' C=0.1 epsilon=0.1 **Random Forest Regressor** n_estimators=50 max_depth=None **Gradient Boosting Regressor** n_estimators=50 learning_rate=0.1 max_depth=3 **KNeighbors Regressor** n_neighbors=3 weights='uniform' **Neural Networks** Epoch=100 best_hps = {'num_layers': 23, 'units_0': 416, 'units_1': 160, 'learning_rate': 1e-05, 'units_2': 192, 'units_3': 608, 'units_4': 608, 'units_5': 416, 'units_6': 960, 'units_7': 608, 'units_8': 128, 'units_9': 896, 'units_10': 160, 'units_11': 864, 'units_12': 544, 'units_13': 704, 'units_14': 224, 'units_15': 1024, 'units_16': 224, 'units_17': 800, 'units_18': 32, 'units_19': 32, 'units_20': 32, 'units_21': 32, 'units_22': 32}	**Linear Regression** **fit_intercept**='True' Support Vector Regressor kernel='linear' C=0.1 epsilon=0.1 Random Forest Regressor n_estimators=50 max_depth=None Gradient Boosting Regressor n_estimators=50 learning_rate=0.1 max_depth=3 KNeighbors **Regressor** **n_neighbors**=3 weights='uniform' **Neural Networks** Epoch=100 best_hps = {'num_layers': 49, 'units_0': 576, 'units_1': 224, 'learning_rate': 1e-05, 'units_2': 288, 'units_3': 64, 'units_4': 992, 'units_5': 128, 'units_6': 64, 'units_7': 448, 'units_8': 128, 'units_9': 448, 'units_10': 288, 'units_11': 576, 'units_12': 672, 'units_13': 800, 'units_14': 864, 'units_15': 896, 'units_16': 800, 'units_17': 960, 'units_18': 96, 'units_19': 32, 'units_20': 960, 'units_21': 768, 'units_22': 416, 'units_23': 512, 'units_24': 640, 'units_25': 928, 'units_26': 320, 'units_27': 32, 'units_28': 480, 'units_29': 288, 'units_30': 960, 'units_31': 64, 'units_32': 896, 'units_33': 320, 'units_34': 832, 'units_35': 352, 'units_36': 512, 'units_37': 96, 'units_38': 416, 'units_39': 1024, 'units_40': 736, 'units_41': 672, 'units_42': 608, 'units_43': 736, 'units_44': 608, 'units_45': 32, 'units_46': 32, 'units_47': 32, 'units_48': 32}	**Linear Regression** **fit_intercept**='True' Support Vector Regressor kernel='linear' C=0.1 epsilon=0.1 Random Forest Regressor n_estimators=50 max_depth=None Gradient Boosting Regressor n_estimators=50 learning_rate=0.1 max_depth=3 KNeighbors **Regressor** **n_neighbors**=3 weights='uniform' **Neural Networks** Epoch=100 best_hps = {'num_layers': 24, 'units_0': 864, 'units_1': 288, 'learning_rate': 0.001, 'units_2': 480, 'units_3': 800, 'units_4': 576, 'units_5': 384, 'units_6': 64, 'units_7': 512, 'units_8': 352, 'units_9': 672, 'units_10': 64, 'units_11': 224, 'units_12': 448, 'units_13': 352, 'units_14': 768, 'units_15': 640, 'units_16': 832, 'units_17': 992, 'units_18': 224, 'units_19': 800, 'units_20': 64, 'units_21': 896, 'units_22': 512, 'units_23': 1024, 'units_24': 224, 'units_25': 160, 'units_26': 800}	**Linear Regression** **fit_intercept**='True' Support Vector Regressor kernel='linear' C=0.1 epsilon=0.1 Random Forest Regressor n_estimators=50 max_depth=None Gradient Boosting Regressor n_estimators=50 learning_rate=0.1 max_depth=3 KNeighbors **Regressor** **n_neighbors**=3 weights='uniform' **Neural Networks** Epoch=100 best_hps = {'num_layers': 35, 'units_0': 288, 'units_1': 960, 'learning_rate': 0.0001, 'units_2': 288, 'units_3': 128, 'units_4': 288, 'units_5': 256, 'units_6': 64, 'units_7': 192, 'units_8': 672, 'units_9': 416, 'units_10': 896, 'units_11': 352, 'units_12': 704, 'units_13': 736, 'units_14': 672, 'units_15': 384, 'units_16': 608, 'units_17': 224, 'units_18': 544, 'units_19': 256, 'units_20': 288, 'units_21': 320, 'units_22': 992, 'units_23': 544, 'units_24': 768, 'units_25': 576, 'units_26': 128, 'units_27': 96, 'units_28': 928, 'units_29': 544, 'units_30': 864, 'units_31': 544, 'units_32': 32, 'units_33': 864, 'units_34': 192, 'units_35': 352, 'units_36': 608, 'units_37': 320, 'units_38': 1024, 'units_39': 352, 'units_40': 192}	**Linear Regression** **fit_intercept**='True' Support Vector Regressor kernel='linear' C=0.1 epsilon=0.1 Random Forest Regressor n_estimators=50 max_depth=None Gradient Boosting Regressor n_estimators=50 learning_rate=0.1 max_depth=3 KNeighbors **Regressor** **n_neighbors**=3 weights='uniform' **Neural Networks** Epoch=100 best_hps = {'num_layers': 7, 'units_0': 352, 'units_1': 480, 'learning_rate': 0.001, 'units_2': 896, 'units_3': 640, 'units_4': 736, 'units_5': 768, 'units_6': 576, 'units_7': 576, 'units_8': 672, 'units_9': 32, 'units_10': 128, 'units_11': 256, 'units_12': 96, 'units_13': 928, 'units_14': 64, 'units_15': 672, 'units_16': 736,'units_17': 224, 'units_18': 960, 'units_19': 768, 'units_20': 224, 'units_21': 608, 'units_22': 96,'units_23': 832, 'units_24': 704, 'units_25': 352, 'units_26': 736, 'units_27': 704, 'units_28': 832,'units_29': 512, 'units_30': 992, 'units_31': 544, 'units_32': 640, 'units_33': 576}

Model Training and Evaluation (Cont...)

Loss (Lag 1)	Loss (Lag 2)	Loss (Lag 3)	Loss (Lag 4)	Loss (Lag 5)
Linear Regression MAE: 2.10 MSE: 11.67 RMSE: 3.42	Linear Regression MAE: 2.10 MSE: 11.64 RMSE: 3.41	Linear Regression MAE: 2.10 MSE: 11.64 RMSE: 3.41	Linear Regression MAE: 2.10 MSE: 11.63 RMSE: 3.41	Linear Regression MAE: 2.10 MSE: 11.64 RMSE: 3.41
Support Vector Regressor MAE: 2.13 MSE: 11.82 RMSE: 3.44	Support Vector Regressor MAE: 2.21 MSE: 12.34 RMSE: 3.51	Support Vector Regressor MAE: 2.25 MSE: 12.68 RMSE: 3.56	Support Vector Regressor MAE: 2.26 MSE: 12.77 RMSE: 3.57	Support Vector Regressor MAE: 2.27 MSE: 12.79 RMSE: 3.58
Random Forest Regressor MAE: 102.08 MSE: 17969.81 RMSE: 134.05	Random Forest Regressor MAE: 102.07 MSE: 17975.18 RMSE: 134.07	Random Forest Regressor MAE: 102.10 MSE: 17985.23 RMSE: 134.11	Random Forest Regressor MAE: 102.61 MSE: 18104.22 RMSE: 134.55	Random Forest Regressor MAE: 102.72 MSE: 18130.11 RMSE: 134.65
Gradient Boosting Regressor MAE: 102.80 MSE: 18143.42 RMSE: 134.70	Gradient Boosting Regressor MAE: 102.84 MSE: 18155.04 RMSE: 134.74	Gradient Boosting Regressor MAE: 102.86 MSE: 18159.30 RMSE: 134.76	Gradient Boosting Regressor MAE: 102.86 MSE: 18159.83 RMSE: 134.76	Gradient Boosting Regressor MAE: 102.98 MSE: 18185.67 RMSE: 134.85
KNeighbors Regressor MAE: 102.16 MSE: 17991.52 RMSE: 134.13	KNeighbors Regressor MAE: 102.12 MSE: 17980.90 RMSE: 134.09	KNeighbors Regressor MAE: 101.97 MSE: 17951.13 RMSE: 133.98	KNeighbors Regressor MAE: 102.10 MSE: 17976.67 RMSE: 134.08	KNeighbors Regressor MAE: 102.32 MSE: 18023.50 RMSE: 134.25
Neural Networks MAE: 5.33 MSE: 59.98 RMSE: 7.74	Neural Networks MAE: 2.52 MSE: 14.29 RMSE: 3.78	Neural Networks MAE: 19.39 MSE: 672.40 RMSE: 25.93	Neural Networks MAE: 6.51 MSE: 67.21 RMSE: 8.20	Neural Networks MAE: 2.45 MSE: 14.76 RMSE: 3.84

Results and Analysis

- According to above model evaluation results, linear regression has got **MAE (2.10)**, **MSE (11.63)**, and **RMSE (3.14)** using 4 timesteps as features and parameter (**fit_intercept='True'**).
- It is the best model for the supposed dataset among others.



Future Scope

- Exploring additional features and considering incorporating external factors such as economic indicators, news sentiment, or social media data may have predictive power for stock price movements.
- Feature engineering plays a crucial role in enhancing the model's predictive capabilities.

Conclusion

- Data mining and machine learning plays a vital role in stock price prediction due to its ability to analyze large volumes of data, identify patterns, and make accurate predictions.
- The model can augment decision-making processes by providing predictions and identifying potential investment opportunities. It can assist in optimizing trading strategies, managing risk, and improving portfolio allocation.
- Traders and investors can use the predictions as guidance for making informed decisions on when to enter or exit positions.

References

- https://ceur-ws.org/Vol-3283/Paper86.pdf
- https://www.diva-portal.org/smash/get/diva2:1672304/FULLTEXT01.pdf
- https://arxiv.org/ftp/arxiv/papers/2111/2111.01137.pdf
- https://www.diva-portal.org/smash/get/diva2:1672304/FULLTEXT01.pdf
- https://ieeexplore.ieee.org/document/8920761
- https://iopscience.iop.org/article/10.1088/1742-6596/2161/1/012065/pdf
- file:///C:/Users/htet/Downloads/Stock_Market_Prediction_Using_Machine_Learning.pdf
- https://ieeexplore.ieee.org/document/8703332

Thank You!