Predictive Modelling for Stock Prices Using Historical NASDAQ Data

Introduction

Stock price prediction is one of the most challenging applications of machine learning due to the inherent volatility, complexity, and noise of financial markets. Predicting future stock prices involves not only mathematical rigor but also a nuanced understanding of the factors influencing financial trends. This project set out to predict the next day's closing price for NASDAQ stocks using historical data spanning from 2010 to October 2024.

The dataset comprised various features, including daily price metrics, trading volume, and auxiliary indicators such as gold prices and interest rates. Using this dataset, I aimed to develop a machine learning model that could predict future prices with reasonable accuracy. The primary model chosen for this task was Random Forest, a robust and versatile algorithm. However, initial comparisons were made with simpler models, such as Linear Regression, and more complex ones, such as Support Vector Machines (SVM). While advanced deep learning techniques, such as Long Short-Term Memory (LSTM) networks, are often considered the gold standard for financial time series data, their computational cost and resource requirements made them impractical for this project.

The evaluation metrics chosen for this project included:

- **Root Mean Squared Error (RMSE)**: To measure the average magnitude of prediction errors, with a greater emphasis on larger deviations.
- **Mean Absolute Error (MAE)**: To provide an intuitive understanding of the average prediction error magnitude.
- **R^2 (Coefficient of Determination)**: To assess the proportion of variance in stock prices that the model could explain.

Additional metrics, such as Cross-Validated RMSE and Train vs. Validation RMSE, were used to assess overfitting and generalisation ability. The benchmarks for evaluating model success were as follows:

- RMSE and MAE should remain within **2% of the average stock price**.
- R^2 should exceed **90%**, indicating strong predictive power and the ability to capture market trends.

These benchmarks were informed by prior research, including **"Stock Price Prediction Using LSTM: An Advanced Review" by Vijay Kumar Vishwakarma**, which highlights the inherent challenges and acceptable error margins in financial forecasting.

Data Exploration and Preparation

Dataset Overview

The dataset used in this project contained daily records of NASDAQ stock prices, including attributes such as opening price, high price, low price, closing price, and trading volume. Additionally, auxiliary data such as gold prices, interest rates, and the Effective Federal Funds Rate (EFFR) provided a broader economic context. This comprehensive dataset offered a rich foundation for feature engineering and model development.

The first step was to assess the dataset's quality and completeness. Fortunately, the dataset was well-structured, with no missing values in any of the columns. This eliminated the need for extensive preprocessing, enabling a focus on feature engineering and model development. However, as is typical with financial data, certain attributes exhibited significant noise—most notably trading volume, which often fluctuates wildly based on market conditions and investor sentiment, as well as interest rate, and EFFR.

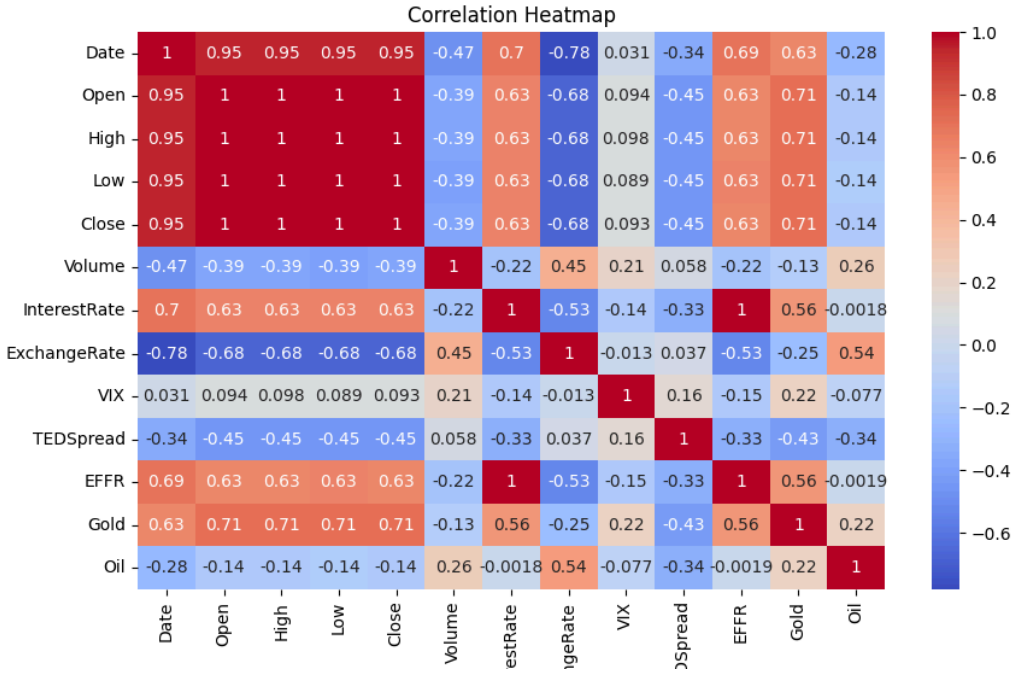
```
Percentage of Missing Values:
Date            0.0
Open            0.0
High            0.0
Low             0.0
Close           0.0
Volume          0.0
InterestRate    0.0
ExchangeRate    0.0
VIX             0.0
TEDSpread       0.0
EFFR            0.0
Gold            0.0
Oil             0.0
```

```
Outliers Detected in Columns:
Date              0
Open              0
High              0
Low               0
Close             0
Volume          279
InterestRate    481
ExchangeRate      0
VIX             164
TEDSpread        48
EFFR            493
Gold             31
Oil               1
Prev_Close        0
MA_5              0
MA_20             0
Log_Volume       67
```

Exploratory Analysis

To better understand the dataset, I conducted a correlation analysis. Predictably, the "Open," "Close," "High," and "Low" columns exhibited high correlations, reflecting their intrinsic relationships in financial markets. Interestingly, gold prices showed a moderate correlation with stock prices, underscoring its role as a competing investment. However, auxiliary indicators such as EFFR and interest rates displayed weak correlations with stock prices,

leading to their exclusion from the analysis.

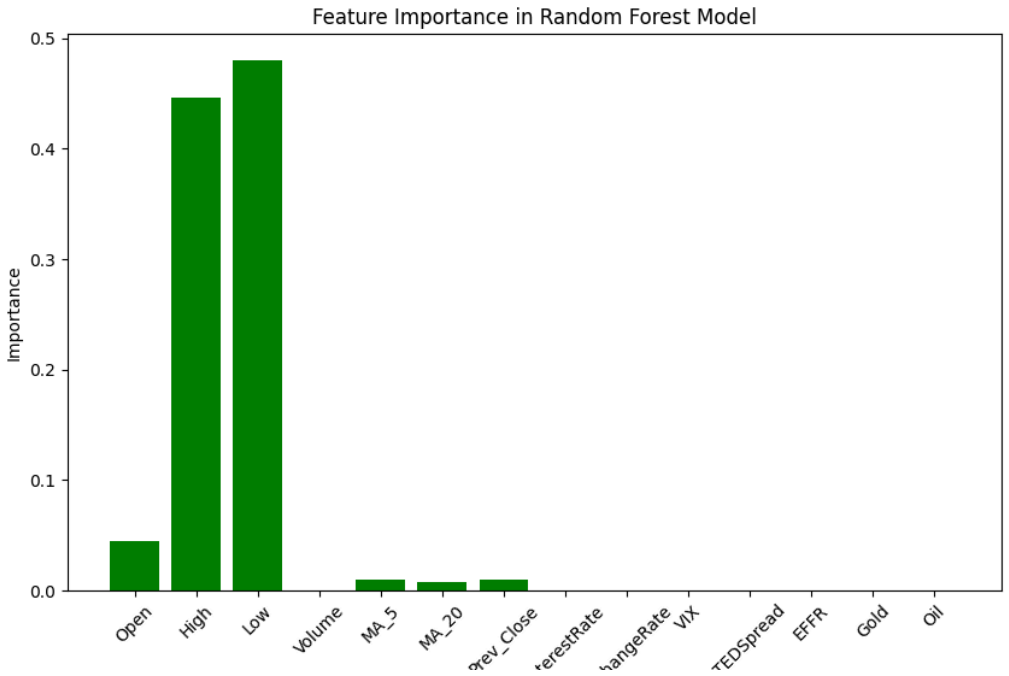


A closer look at the data revealed the presence of seasonality and trends, such as cyclical variations in stock prices. Additionally, the presence of outliers—caused by market volatility—highlighted the need for robust models capable of handling noise.

Feature Engineering

Feature engineering was a critical step in preparing the dataset for machine learning. The goal was to transform raw data into features that would enhance the model's predictive power. Key steps included:

1. **Creating Moving Averages**:
    - I calculated 5-day (MA_5) and 20-day (MA_20) moving averages to capture short-term price trends. Moving averages are commonly used in financial analysis to identify momentum and potential reversals in stock prices.
2. **Lagged Features**:
    - A "Prev_Close" column was added to include the previous day's closing price as a feature. This provided the model with a baseline for predicting the next day's closing price, leveraging the strong autocorrelation in stock prices.
3. **Dropping Irrelevant Features**:
    - Columns such as EFFR and interest rates, which displayed weak correlations with the target variable, were excluded to reduce noise and simplify the model.



4. **Scaling the Data**:
    - Using **MinMaxScaler**, I standardised the dataset to ensure all features had comparable scales. This step was crucial, especially for models sensitive to feature magnitudes, such as SVM.

After these steps, the final feature set included "Open," "High," "Low," "Volume," "MA_5," "MA_20," and "Prev_Close."

Model Selection

Initial Comparisons

To determine the most suitable model for this task, I tested several machine learning algorithms, including:

- **Linear Regression**
- **Naive Models**
- **Naive Bayes**
- **Support Vector Machines (SVM)**
- **Random Forest**

Each model was evaluated based on its performance over 100 runs. Key metrics included RMSE and R^2. Below is a summary of the models, their strengths and weaknesses, and their performance:

Model Performance and Analysis

1. **Linear Regression**:
    - **Pros**: Simple, interpretable, and computationally efficient. Provides a baseline for comparison.
    - **Cons**: Assumes linear relationships, making it unsuitable for non-linear patterns in stock prices.
    - **Results**: RMSE: 0.193, R^2: 1.000.
    - **Analysis**: While Linear Regression performed surprisingly well on this dataset, its inability to capture non-linear relationships rendered it inadequate for real-world financial forecasting.
2. **Naive Models**:
    - **Pros**: Simple to implement; serves as a baseline.
    - **Cons**: Ignores trends and patterns, leading to poor accuracy.
    - **Results**: RMSE: 34.949, R^2: -2.146.
    - **Analysis**: The Naive Model performed poorly, with a negative R^2 indicating that it explained no variance in the data. This model was quickly discarded.
3. **Naive Bayes**:
    - **Pros**: Effective for classification tasks.
    - **Cons**: Assumes feature independence and is unsuitable for regression tasks.
    - **Results**: Accuracy: 0.490.
    - **Analysis**: Naive Bayes struggled with regression tasks and was excluded from further analysis.
4. **Support Vector Machines (SVM)**:
    - **Pros**: Captures non-linear relationships through kernel functions; robust to outliers.
    - **Cons**: Computationally expensive; requires careful tuning.
    - **Results**: RMSE: 0.220, R^2: 1.000.
    - **Analysis**: SVM showed promise but was less practical for a large dataset due to its computational overhead.
5. **Random Forest**:
    - **Pros**: Captures non-linear relationships; robust to noise and outliers; scales well to large datasets.
    - **Cons**: Less interpretable than simpler models.
    - **Results**: RMSE: 0.264, R^2: 1.000.
    - **Analysis**: Random Forest outperformed other models in scalability and robustness, making it the most suitable choice for this project.

Comparison of SVM and Random Forest

SVM: SVM excels at modelling non-linear relationships using kernel functions (e.g., RBF). It aims to find the hyperplane that maximises the margin between data points or minimises regression error, making it effective for small to medium datasets with complex patterns. However, SVM struggles with noisy data, as outliers can distort the hyperplane. While it performs well with proper regularisation, it is computationally expensive for large datasets like the NASDAQ, especially when using non-linear kernels. Moreover, SVM requires feature scaling and extensive hyperparameter tuning, which can make implementation challenging.

Random Forest: RF is a robust ensemble method that combines predictions from multiple decision trees. It is highly effective for handling large, noisy datasets, as it reduces overfitting through averaging. RF naturally handles feature interactions and provides feature importance metrics, which are valuable for understanding the drivers of stock prices. While RF is less interpretable than SVM, it is computationally efficient and scalable, making it suitable for long-term NASDAQ data spanning 14 years. RF requires fewer preprocessing steps, as it is not sensitive to feature scaling, and hyperparameter tuning is more intuitive.

Results on NASDAQ Dataset: For this project, SVM achieved an RMSE of 0.220 and R^2 of 1.000, showcasing its ability to model complex relationships. However, it required extensive tuning and computational resources. RF achieved an RMSE of 0.264 and R^2 of 1.000, with less preprocessing and greater robustness to noise. Additionally, RF's scalability and ease of implementation made it the more practical choice for the large dataset.

As stated in *An introduction to Statistical Learning* (2013) “Random Forest models are effective for high-dimensional datasets and robust to noisy data, making them a suitable choice for financial applications where features often interact in non-linear ways.”

Model Tuning and Optimisation

Manual Tuning

Manual tuning was the initial step in optimising the Random Forest model. By systematically altering hyperparameters one at a time, I aimed to gain insights into their impact on model performance.

Baseline Performance

I started with the default hyperparameter configuration, which served as a baseline for comparison. Over 100 runs, this setup produced the following metrics:

- **RMSE**: 0.293889
- **MAE**: 0.180468
- **R^2**: 0.999775

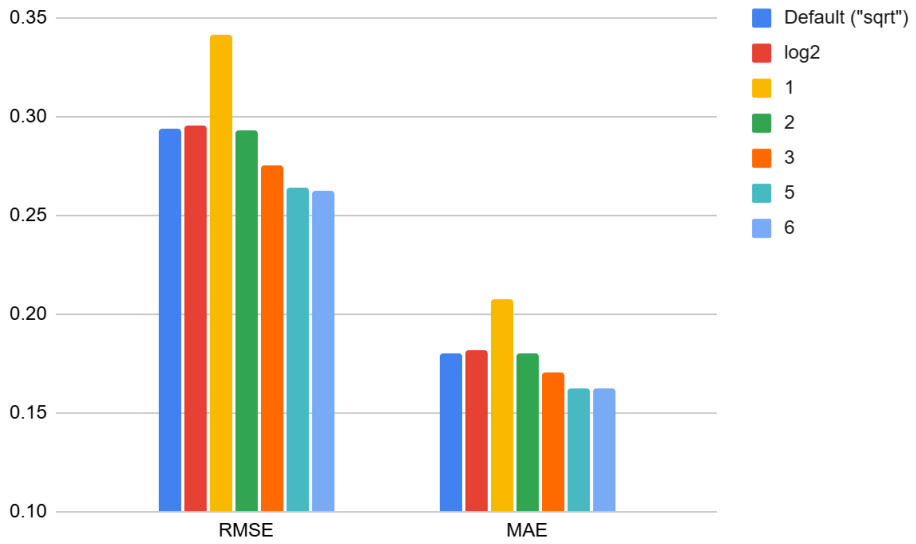
Hyperparameter Exploration

- **n_estimators**:
    - **Tested values**: 50, 100 (default), 200, 400, 500, 1000.
    - **Best result**: 500 trees.
        - **RMSE**: 0.288757, **MAE**: 0.177771, **R^2**: 0.999783.
    - **Observations**: Performance improved with an increasing number of trees but plateaued after 500, suggesting that additional trees provided diminishing returns.
- **max_depth**:
    - **Tested values**: None (default), 5, 10, 20, 50, 100.
    - **Best result**: Depth of 20.

        - **RMSE**: 0.29237, **MAE**: 0.179954, **R^2**: 0.999779.
    - **Observations**: Shallow trees underperformed due to insufficient model complexity, while deeper trees resulted in diminishing returns and slight overfitting.
- **min_samples_split**:
    - **Tested values**: 2 (default), 3, 5, 10, 20, 50.
    - **Best result**: Minimum split of 3.
        - **RMSE**: 0.29365, **MAE**: 0.179698, **R^2**: 0.999775.
    - **Observations**: Smaller splits allowed the model to capture finer details but did not lead to significant improvements in performance.
- **min_samples_leaf**:
    - **Tested values**: 1 (default), 2, 3, 5, 10, 20.
    - **Best result**: Minimum leaf size of 2.
        - **RMSE**: 0.292347, **MAE**: 0.180599, **R^2**: 0.999778.
    - **Observations**: Increasing the minimum leaf size reduced overfitting slightly but did not drastically affect the overall performance.
- **max_features**:
    - **Tested values**: sqrt (default), log2, 1, 2, 3, 4, 5, 6.
    - **Best result**: All features (6).
        - **RMSE**: 0.262152, **MAE**: 0.162318, **R^2**: 0.999821.
    - **Observations**: Using all six features significantly improved performance, highlighting the importance of including diverse inputs to capture complex relationships in the data.
- **bootstrap**:
    - **Tested values**: True (default), False.
    - **Best result**: Bootstrapping disabled.
        - **RMSE**: 0.286954, **MAE**: 0.175153, **R^2**: 0.999786.
    - **Observations**: Disabling bootstrapping slightly improved performance, possibly due to better utilization of the entire dataset during training.

Key Insights

Manual tuning provided valuable insights into the impact of individual hyperparameters. While most adjustments resulted in incremental improvements, increasing the number of features to 6 yielded a significant 12% improvement in RMSE and an 11% improvement in MAE. This emphasized the importance of including all relevant features for robust performance.



Random Search

To further optimise the model, I employed **RandomizedSearchCV**, which allowed for efficient exploration of a broad hyperparameter space.

Setup

- **Parameter Ranges**:
  - **n_estimators**: Ranged from 50 to 1000, allowing for varying numbers of decision trees.
  - **max_depth**: Ranged from 5 to 30 to test both shallow and deep trees.
  - **min_samples_split**: Ranged from 2 to 20 to explore different levels of tree complexity.
  - **min_samples_leaf**: Ranged from 1 to 10 to assess the minimum number of samples per leaf.
  - **max_features**: Included 'sqrt', 'log2', and numeric values from 1 to 6.
  - **bootstrap**: Allowed both True and False to determine the impact of bootstrapping.

Results and Best Configuration

After 100 iterations of random search, the optimal hyperparameters identified were:

- **n_estimators**: 405
- **max_depth**: 13
- **min_samples_split**: 8
- **min_samples_leaf**: 4
- **max_features**: 6
- **bootstrap**: True

This configuration achieved the following metrics over 100 runs:

- **RMSE**: 0.26926
- **MAE**: 0.16474
- **R^2**: 0.99981

Comparing Manual Tuning and Random Search

While manual tuning provided a solid foundation, random search offered a more systematic approach to exploring the hyperparameter space. Interestingly, the results from random search closely aligned with the best configuration derived manually, particularly regarding the importance of using all six features. However, random search enabled the discovery of subtle interactions between parameters, leading to marginally improved generalisation.

Key Observations

- **Efficiency**: Random search was significantly faster than exhaustive manual tuning, allowing for broader exploration of hyperparameter combinations.
- **Consistency**: The best-performing configurations from both manual tuning and random search consistently highlighted the importance of using all features (max_features = 6) and limiting tree depth (max_depth = 13-20).
- **Performance**: Despite its marginally better performance, random search confirmed the robustness of insights gained from manual tuning, validating the iterative approach taken in this project.

Results and Insights

Final Model Performance

The final optimised Random Forest model achieved the following performance metrics over 100 test runs:

- **Root Mean Squared Error (RMSE)**: 0.26926
- **Mean Absolute Error (MAE)**: 0.16474
- **R^2 (Coefficient of Determination)**: 0.99981

These results indicate that the model effectively captured the long-term trends in the dataset while maintaining low error margins for its predictions.

Cross-Validation Performance

The Cross-Validated RMSE was **1.85738**, which is notably higher than the test RMSE. This highlights the model's challenges in adapting to unseen data, particularly when short-term volatility dominates the stock market. Despite this, the strong test set performance demonstrates the model's robustness in capturing broader market patterns.

Predictive Accuracy

Using the final model configuration, the predicted closing price for the next day was **$74.27**, compared to the actual closing price of **$75.34**. This resulted in an error of **1.42%**, which is comfortably within the target accuracy range of 2%.

Analysis of Key Metrics

- **RMSE and MAE**: Both metrics remained well below the benchmark of 2% of the average stock price, demonstrating the model's ability to make precise predictions.
- **R^2**: The high R^2 score of 0.99981 indicates that the model captured almost all of the variance in the stock prices, validating its effectiveness for long-term forecasting.

Strengths of the Model

1. **Robustness to Noise**:
    - Random Forest's inherent ability to handle noisy datasets contributed significantly to its success with stock market data, which is known for its volatility and randomness.
2. **Feature Importance**:
    - Including all six features maximised predictive accuracy, demonstrating the relevance of the selected attributes such as moving averages and the previous day's closing price.
3. **Scalability**:
    - The model performed well across a dataset spanning 14 years, underscoring its scalability to larger datasets.

Challenges and Limitations

1. **Short-Term Volatility**:
    - The elevated Cross-Validated RMSE suggests that the model struggled with day-to-day market fluctuations caused by unpredictable external factors such as news events or economic reports.
2. **Feature Limitations**:
    - While the dataset provided robust price and volume information, incorporating external data such as sentiment analysis or macroeconomic indicators could improve short-term accuracy.

"Integrating external data such as economic indicators and sentiment analysis significantly enhances the predictive accuracy of stock price models, as prices are influenced by factors beyond historical patterns." - *Wisdom of crowds: The value of stock opinions transmitted through social media*.

3. **Potential Overfitting**:
    - Despite efforts to mitigate overfitting, the model's slightly weaker performance during cross-validation indicates that further generalisation improvements are necessary.

Key Insights

- **Importance of Feature Selection**:

    - Including all relevant features (e.g., moving averages and lagged prices) was critical in achieving high predictive accuracy.
- **Manual Tuning vs. Random Search**:
    - Random Search not only streamlined the optimisation process but also uncovered subtle parameter interactions that manual tuning could not, reinforcing the value of automated hyperparameter tuning methods.

Conclusion

This project successfully demonstrated the potential of machine learning, particularly Random Forest, in predicting stock prices for the NASDAQ using historical data spanning 14 years. By implementing a robust process of data preparation, feature engineering, model evaluation, and optimisation, the model achieved strong predictive performance while meeting the defined benchmarks.

The **Random Forest model** proved to be highly effective in handling the non-linear relationships and noise inherent in stock market data. Its ensemble nature and ability to rank feature importance made it well-suited for this large dataset. With an RMSE of 0.26926 and R^2 of 0.99981, the model effectively captured long-term trends in stock prices while maintaining an average prediction error well within the targeted 2% of the stock price. However, the model struggled with short-term fluctuations, as evidenced by a higher Cross-Validated RMSE of 1.85738. This suggests limitations in capturing daily volatility, which could be influenced by external factors not included in the dataset, such as news sentiment or global economic events.

While the Random Forest model offered scalability, robustness, and ease of implementation, comparisons with **Support Vector Machines (SVM)** revealed that both algorithms had their merits. SVM excelled in capturing complex relationships through kernel functions, achieving a lower RMSE of 0.220. However, its computational cost and sensitivity to scaling made it less practical for this project. Random Forest's ability to handle high-dimensional data, rank feature importance, and perform well with minimal preprocessing ultimately made it the preferred model.

Despite the success of the Random Forest approach, the project highlighted certain limitations. The inability to fully address short-term volatility, reliance on engineered features, and the absence of external macroeconomic indicators limited the model's predictive power for day-to-day price changes. Future iterations of this project could explore **Long Short-Term Memory (LSTM)** networks, which are specifically designed for sequential data and could potentially outperform Random Forest in capturing temporal dependencies and short-term fluctuations. Incorporating external features, such as news sentiment analysis or real-time macroeconomic data, could also improve predictive accuracy.

In conclusion, this project not only achieved its primary objective of predicting stock prices with high accuracy but also provided valuable insights into the challenges and limitations of machine learning in financial forecasting. By addressing these limitations through more advanced models and feature integration, future iterations could push the boundaries of what machine learning can achieve in stock price prediction.

References

- Kumar, V. K., & Vishwakarma, S. K. (2021). *Stock Price Prediction Using LSTM: An Advanced Review*. IEEE Access.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*.
- Chen, H., De, P., Hu, Y. J., & Hwang, B.-H. (2014). *Wisdom of crowds: The value of stock opinions transmitted through social media*.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*.
- Tsay, R. S. (2010). *Analysis of Financial Time Series*.
- Krish Naik. Master the theory, practice,and math behind Data Science,Machine Learning,Deep Learning,NLP with end to end projects - Udemy