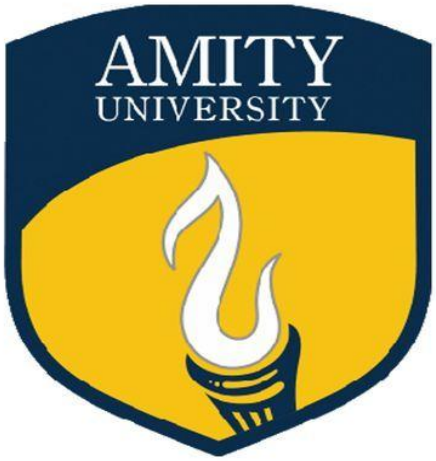


INTEGRATED ASSIGNMENT REPORT

Title : Integrated Assignment on Synthetic Stock Data Analysis
Name : Nitesh
Roll Number: A044161824008
Course : Masters' in Data Science
Date : 9 April, 2025

TOPIC

Analyzing synthetic stock data using machine learning, time series, sampling techniques and inferential statistics.

INTRODUCTION

This integrated assignment focuses on analyzing synthetic stock data through a systematic application of core data science and statistical techniques. The process begins with sampling techniques to simulate realistic data collection methods, followed by inferential statistics to draw conclusions about the population from the sampled data. Building on this, ANOVA was used to determine whether the observed differences among group means were statistically significant. The analysis then moves to count and categorical data, using models like logistic and Poisson regression to interpret binary and discrete variables within the dataset.

From there, the study transitions into financial data analysis, focusing on asset returns, distribution patterns, and volatility modeling to understand market behavior. Time series analysis is then applied using tools like ARIMA and exponential smoothing to model and forecast trends in stock prices over time. The project concludes with machine learning techniques, including decision trees and correlation matrix, to classify data and predict future movements. Each section includes relevant code, graphs, and interpretation, reflecting a practical, end-to-end data science workflow.

OBJECTIVE

The aim of this project is to explore and analyze financial data using a blend of data science techniques. This includes uncovering trends through time series forecasting, drawing insights with exploratory analysis, validating assumptions using statistical tests, and examining patterns in categorical data. We also apply different sampling methods to ensure fair and meaningful analysis. Overall, the goal is to build a well-rounded understanding of the data and support informed financial decisions using Python.

The following are the techniques that has been used in this Integrated assignment to get the desired results from the data :

- Sampling techniques
- Inferential statistics
- Count and Categorical Data Analysis
- Financial Data Analysis
- Time Series
- Machine Learning

In this project, we explored financial data using a mix of data science techniques to draw meaningful insights and make predictions. We started by applying sampling methods like random and stratified sampling to work with manageable portions of the data while keeping it representative. From there, we dove into exploratory analysis, using visual tools and summary stats to understand the behavior of categorical and count-based variables such as sector types, credit ratings, and company classifications. This helped us spot trends, imbalances, and interesting patterns early on.

We then used inferential statistics to test assumptions and draw conclusions from the data — for example, checking if average returns differed significantly between sectors. For the time-based data, we performed a time series analysis, breaking it down into trends and seasonality, and used ARIMA models to forecast future stock prices. Finally, we brought in machine learning techniques to build simple predictive models, helping us classify or predict financial outcomes based on historical data. Overall, the project gave us hands-on experience with a full data science pipeline, from raw data to actionable insights.

Using Sampling techniques and Preparing Stock Data Chronologically

In this section, we extract a random sample from a larger stock dataset to work with a more manageable subset of data. The steps involve sampling, formatting, and organizing the data for analysis.

---

1. Reading the Dataset

We begin by reading the full synthetic stock dataset from a CSV file. This dataset contains multiple rows of stock-related data such as prices, dates, and other financial information.

---

2. Random Sampling

To work with a smaller but still representative subset, we take a random sample of 150 rows from the dataset. Sampling is done without replacement, meaning no duplicate rows are selected. A fixed **random_state** is used to make the sampling process reproducible.

---

3. Converting and Sorting Dates

After sampling, the Date column is converted to datetime format (if it wasn't already). Then, the sampled data is sorted in chronological order based on the date. Sorting is important to ensure that any time series or trend analysis we perform later uses properly ordered data.

---

4. Adding Serial Numbers

To make the dataset easier to reference, a serial number column is added at the beginning of the table. The serial numbers start from 1 and go up to the total number of sampled rows (150 in this case).

---

5. Displaying and Saving the Result

The final sorted sample is printed in a clean table format for inspection. This helps verify that the sample is correctly ordered and formatted. Optionally, the processed data is saved to a new CSV file called chronological_sample.csv, which can be used in later steps like time series modeling.

Interpretation of the Result

After executing the sampling and formatting code, we obtain a **chronologically sorted subset** of stock market data consisting of **150 randomly selected rows**. Each row represents a particular date's stock data for a company, and the rows are now ordered by date from earliest to latest.

By adding a serial number column, the data becomes easier to navigate and reference, especially when analyzing trends over time or presenting the data in reports. Since the sample was taken randomly (but without replacement), it represents a diverse mix of entries across different companies, sectors, and trends, while still preserving the natural time flow once sorted.

This prepared dataset—saved as chronological_sample.csv—is now ready for further time series analysis, such as ARIMA modeling or trend visualization, and ensures that any insights derived are based on properly ordered historical data.

Inferential statistics Performed on the Stock Dataset

1: One-Sample t-Test on Closing Prices

In this step, a one-sample t-test is performed to check whether the average of the 'Close' prices in the sample is statistically different from a benchmark value of 100. The test assumes that the data comes from a normally distributed population.

The t-test compares the sample mean to the hypothesized population mean and calculates the t-statistic, which represents how many standard errors the sample mean is away from the hypothesized mean. A p-value is also calculated, which indicates the probability of observing such a result if the null hypothesis were true (i.e., if the actual mean was 100).

If the p-value is small (typically less than 0.05), we have sufficient evidence to reject the null hypothesis and conclude that the average closing price is significantly different from 100.

2: Proportion Test on Gain Days

This part of the analysis focuses on identifying whether the proportion of days where the closing price is higher than the opening price (referred to as “gain days”) is equal to 50%.

The test calculates the sample proportion of gain days and compares it to a hypothesized neutral proportion of 0.5. A z-score is computed to show how far the observed proportion is from the expected proportion in terms of standard errors. A two-tailed p-value helps assess whether the deviation is statistically significant.

If this p-value is less than 0.05, it indicates that the market behavior (in terms of gain days) is not balanced and leans toward being either bullish or bearish.

3: F-Test on Variance Between Early and Late Periods

The sample data is divided into two equal parts: the first 15 records as the early period and the remaining records as the late period. The goal here is to determine whether the variability (variance) in the closing prices has changed over time.

An F-test is used for this comparison. It calculates the ratio of the variances of the two groups and compares it to an F-distribution to obtain a p-value. The test checks for equality of variances, assuming both samples come from normally distributed populations.

If the p-value is low, we conclude that the variability in closing prices has shifted, suggesting that the stock may have become volatile over time.

4: One-Way ANOVA by Sector

The final test investigates whether there are any significant differences in average closing prices across different sectors. For this, a one-way ANOVA (Analysis of Variance) is used.
The test fits a linear model where the closing price is the dependent variable, and the sector is the independent (categorical) variable. The ANOVA method partitions the total variance into two components:

- Between-group variance (how much closing prices differ across sectors)
- Within-group variance (how much they differ within each sector)

A large F-statistic and a small p-value suggest that sector affiliation does have a meaningful impact on closing prices.

Interpretation of Results

- One-sample t-test: A significant p-value indicates the average closing price is not 100, pointing to a shift in the overall price level.
- Proportion test: A significant result means the frequency of gain days differs from 50%, suggesting market bias.
- F-test: If significant, it implies a change in stock price volatility between the early and late periods.
- ANOVA: A significant result means sector categories affect closing prices, potentially due to differing sector performance or market sentiment.

Implementation of Count and Categorical Data Analysis

1. Data Loading and Preprocessing

The dataset is first loaded using pandas from a CSV file containing chronologically ordered stock data. The 'Trend' column, which describes market direction (e.g., Bullish, Bearish, Neutral), is a categorical variable and must be converted into numerical format. This is done using LabelEncoder, which assigns an integer to each unique trend label.
Next, a set of relevant features is selected to help predict market trends. These features include basic stock indicators like Open, High, Low, Close, and Volume, as well as more informative metrics such as Market_Cap, PE_Ratio, Dividend_Yield, Volatility, and Sentiment_Score. To avoid errors during model training, any missing values in the selected features are replaced with the mean of the corresponding column.

---

2. Splitting the Dataset

The dataset is split into a training set and a testing set using train_test_split, where 70% of the data is used for training the model and 30% is kept aside to evaluate how well the model performs on unseen data. The use of a fixed random state ensures that the same split can be reproduced each time the code runs.

---

3. Feature Standardization

Since the chosen features vary widely in scale (e.g., Volume can be very large while Dividend_Yield may be small), they are standardized using StandardScaler. This transformation shifts the distribution of each feature to have a mean of 0 and a standard deviation of 1. Standardizing improves model performance and training stability.

---

4. Model Training with Logistic Regression

A multinomial logistic regression model is built, which is suitable for problems involving more than two classes—in this case, different market trends. The model is trained using the standardized training data. The solver used is 'lbfgs', which is efficient for handling multiclass logistic regression. The max_iter=1000 parameter ensures that the algorithm has enough iterations to converge.

---

5. Making Predictions and Evaluating the Model

The trained model is then used to make predictions on the test data. Several metrics are used to evaluate its performance:

- Accuracy gives the overall percentage of correct predictions.
- Confusion Matrix shows how well the model performs for each class, helping visualize which types of trends the model is confusing with others.
- Classification Report provides detailed metrics like precision, recall, and F1-score for each trend class.

These metrics collectively help assess how well the model can generalize to new, unseen data.

---

6. Feature Importance (Model Coefficients)

To interpret how the model makes its predictions, the coefficients for each feature are printed. These coefficients show the influence of each feature on the probability of a particular market trend. Features with higher absolute values have a stronger impact. A positive coefficient for a feature means that as the feature increases, the likelihood of a certain trend also increases, and vice versa.

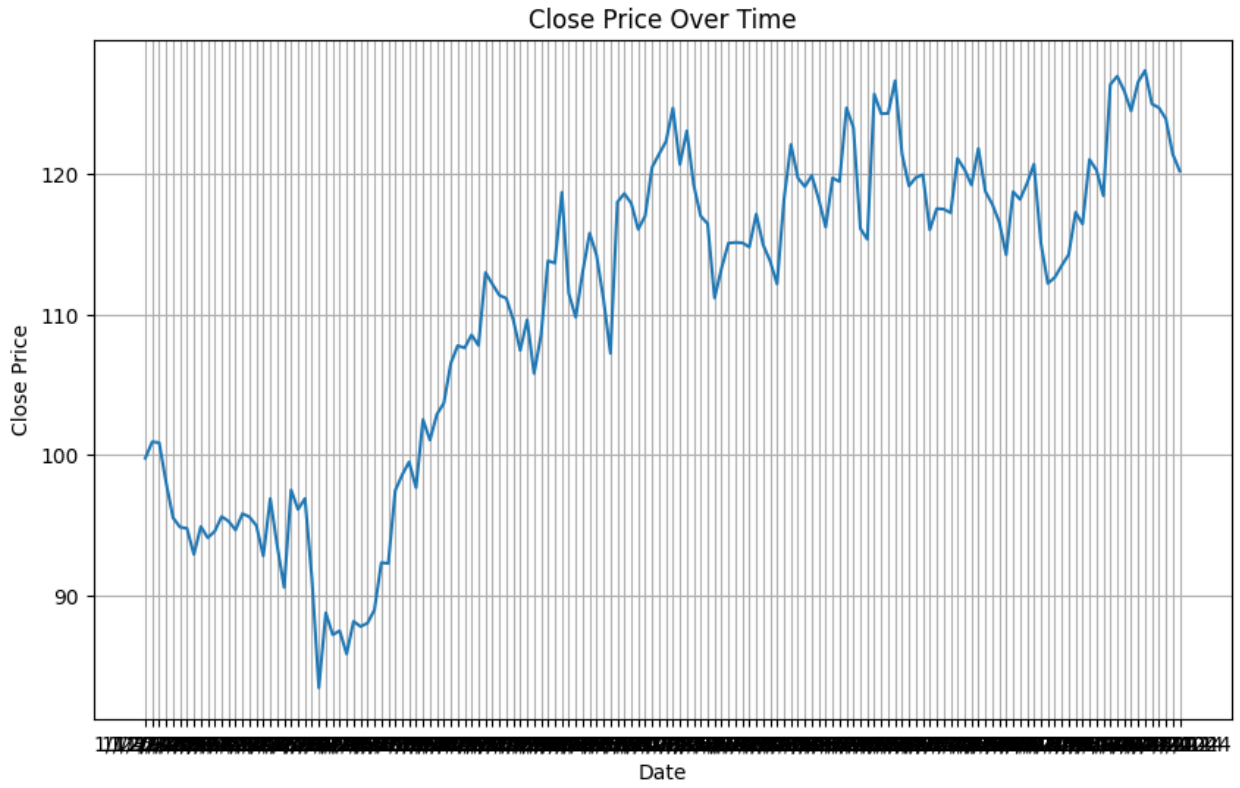


Figure: Close Price Over Time

This line chart shows the daily close price of a financial asset over a specific time period.

- X-axis (Date): Represents the timeline.
- Y-axis (Close Price): Indicates the asset's closing price.

Key Observations:

- There is an overall upward trend, indicating growth in the asset's value.
- The chart shows volatility, with noticeable fluctuations in price, including short-term dips and spikes.
- A significant drop occurs early in the timeline, followed by a steady rise.
- The price stabilizes at higher levels towards the end, showing potential market confidence or recovery.

Conclusion:

The figure highlights the asset's positive performance over time, while also reflecting short-term market volatility. This visualization is useful for analyzing trends, price behavior, and potential investment risks.

Figure: Close Price vs. Volume

This scatter plot visualizes the relationship between trading volume (x-axis) and close price (y-axis) of a financial asset.

Key Observations:

- The points are widely scattered, indicating no strong correlation between trading volume and closing price.
- Both high and low prices occur across various volume levels, suggesting that price movements are not directly tied to volume.
- The spread is dense in the mid-to-high volume range, possibly indicating higher trading activity in that range.

Conclusion:

There is no clear linear or nonlinear relationship between close price and volume based on this plot. This suggests that other factors, beyond trading volume, may be influencing the price behavior of the asset.

How Financial Data Analysis was used in this assignment

1. Data Loading and Preprocessing

The dataset is first imported using pandas from a CSV file that contains chronological stock data. The Date column is converted to datetime format to ensure proper time-series handling. To maintain consistency in time progression, the data is sorted by both company and date.

---

2. Log Return Calculation

For each company, the log returns are calculated using the formula:
Log Return = ln(Close_t / Close_{t−1})
This step involves grouping the dataset by company and then computing the daily log returns. Log returns are a commonly used measure in finance as they stabilize variance and allow easier modeling of returns over time.

After calculation, the Log_Returns column is appended to each group, and all company-wise data is combined into a single dataframe for further analysis.

---

3. Descriptive Statistics

The combined log returns data is used to calculate four key statistics:

- **Mean**: Represents the average daily return across all companies.
- **Standard Deviation**: Indicates the volatility or risk.
- **Skewness**: Measures asymmetry of the distribution. A positive skew indicates a longer tail on the right; a negative skew, on the left.
- **Kurtosis**: Captures the "tailedness" of the distribution. Higher kurtosis indicates more outliers.

These statistics help understand the nature of return distribution and potential risk in the data.

---

4. Augmented Dickey-Fuller (ADF) Test

To check whether the log returns are stationary (i.e., their statistical properties like mean and variance do not change over time), the ADF test is performed.

- **Null Hypothesis (H_o)**: The data is non-stationary.
- **Alternative Hypothesis (H_1)**: The data is stationary.

If the **p-value < 0.05**, we reject the null hypothesis and conclude that the series is stationary. Stationarity is crucial for time series modeling, such as ARIMA.

---

5. Visual Analysis - Distribution Plot

A histogram with a KDE (Kernel Density Estimate) line is plotted to visualize the distribution of log returns. A normal distribution curve is also overlaid to visually compare how well the actual data fits a Gaussian distribution.

This visual helps in understanding if returns follow a normal pattern or have heavier tails, which is often the case in real-world financial data.

---

6. Visual Analysis - Q-Q Plot

The Quantile-Quantile (Q-Q) plot is used to further assess normality. If the log returns follow a normal distribution, the points will lie along the diagonal line. Deviations from this line indicate departures from normality, such as skewness or fat tails.

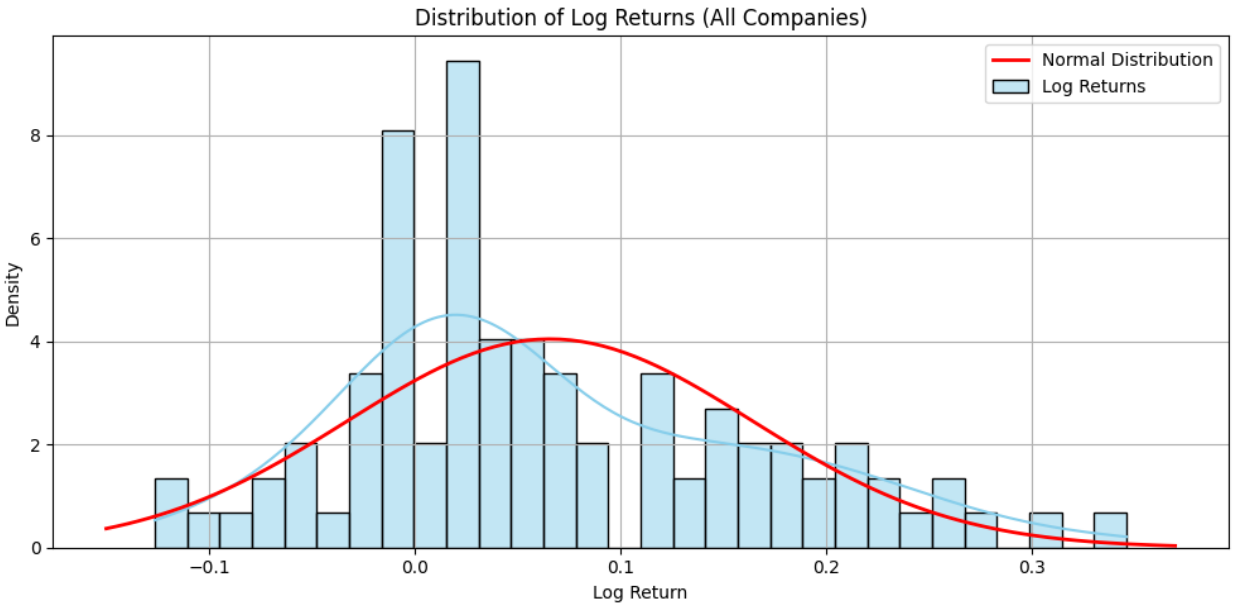


Figure: Distribution of Log Returns (All Companies)

This plot shows the distribution of log returns for multiple companies, comparing the actual return distribution with a normal distribution.

Components:

- Histogram (light blue bars): Shows the frequency of log return values.
- KDE Curve (light blue line): A smoothed estimate of the empirical distribution.
- Red Curve: Represents the theoretical normal distribution with the same mean and variance as the data.

Key Observations:

- The log returns are centered around zero, with most values clustered near the mean.
- The distribution is positively skewed, with a longer right tail, indicating more frequent high positive returns.
- The empirical distribution deviates from normality, especially in the tails, suggesting fat tails and non-normal behavior.

Conclusion:

Log returns roughly follow a bell-shaped curve but do not perfectly match a normal distribution. This implies the presence of skewness and extreme values, which are important considerations in financial risk analysis and modeling.

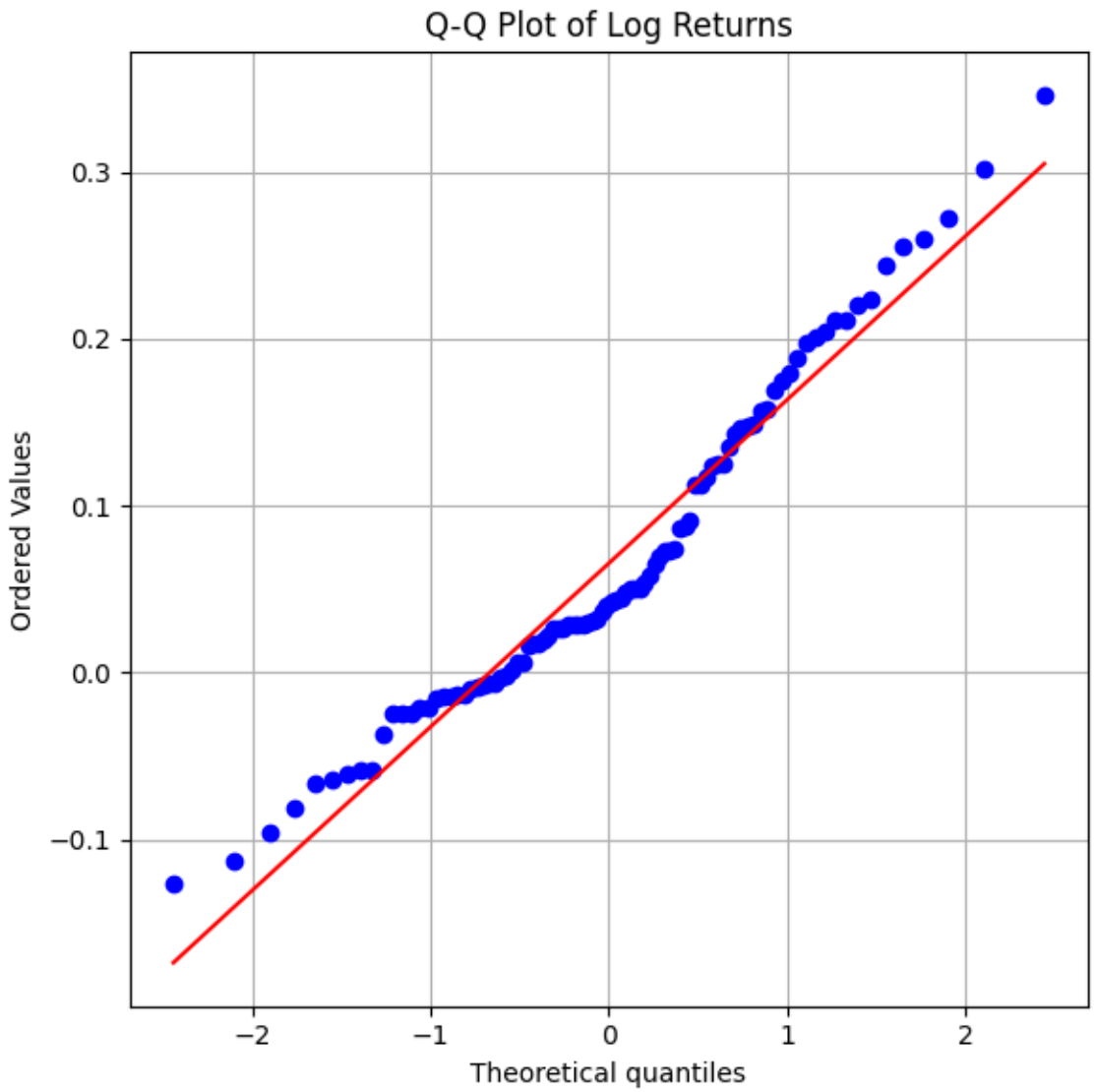


Figure: Q-Q Plot of Log Returns

This **Quantile-Quantile (Q-Q) plot** compares the **distribution of log returns** to a **theoretical normal distribution**.

Interpretation:

- The **x-axis** represents the quantiles from a normal distribution.
- The **y-axis** represents the quantiles of the actual log return data.
- The **red line** is the reference line for a perfect normal distribution. If the data is normally distributed, the points should lie along this line.

Key Observations:

- Most of the points lie **close to the red line**, especially in the center, indicating that **log returns are approximately normally distributed** in the middle range.
- **Deviations** are observed at both ends (tails), with points **slightly diverging** from the line.
    - **Upper tail (right side)**: Observations lie above the line, suggesting **heavier positive tails**.
    - **Lower tail (left side)**: Slight deviation indicates mild **negative tail asymmetry**.

Conclusion:

The Q-Q plot suggests that while the **log returns generally follow a normal distribution**, there are **slight departures in the tails**, pointing to **non-normal behavior** such as **fat tails or skewness**. This aligns with the histogram analysis and has implications for modeling extreme market movements.

Time Series Forecasting Using ARIMA

This section describes how we used the ARIMA (Auto-Regressive Integrated Moving Average) model to analyze and forecast stock closing prices over time. The dataset includes information like company names, sectors, stock prices, trading volume, and trend labels. The overall process involves cleaning the

data, preparing it for modeling, fitting the ARIMA model, visualizing results, and making short-term forecasts.

---

1. Data Preprocessing

1.1 Reading and Sorting Data

We start by reading the stock data from a CSV file. The Date column is converted into a datetime format, and the data is sorted to make sure everything is in proper chronological sequence. This is important because time series models rely on the order of data to identify patterns over time.

1.2 Encoding Categorical Data

Since machine learning models can't work directly with text data, we convert categorical columns like Company, Sector, and Trend into numbers using label encoding. This assigns a unique number to each unique category. These encoded values can then be used for analysis or future modeling.

---

2. Preparing the Time Series

For this analysis, we focus only on the closing price of stocks. The dataset is adjusted so that the Date column becomes the index, creating a clean time series of daily closing prices that we can feed into the ARIMA model.

---

3. Fitting the ARIMA Model

We use the ARIMA model to analyze the historical closing prices. ARIMA combines:

- AR (Auto-Regressive): Uses past values to predict the current one.
- I (Integrated): Applies differencing to make the series stationary.
- MA (Moving Average): Accounts for past forecast errors.

We select ARIMA parameters (2, 1, 2) for this model. After training, we view a summary of the model's performance, including coefficients and how well the model fits the data.

---

4. Visualizing the Fit

Next, we plot the actual closing prices alongside the values predicted by the model. This helps us see how closely the ARIMA model is tracking real-world data. A good fit means the model has successfully captured underlying trends.

---

5. Forecasting the Future

Using the trained model, we generate forecasts for the next 5 days. These predicted values are displayed on a plot along with confidence intervals that show the possible range of future prices. This step demonstrates the model's ability to project short-term market behavior.

---

6. Saving Outputs

Finally, we saved the processed data—including the encoded values—into a new CSV file. We also print out the encoding mappings so it's clear how the original text values (like company names or sector labels) were converted into numbers.

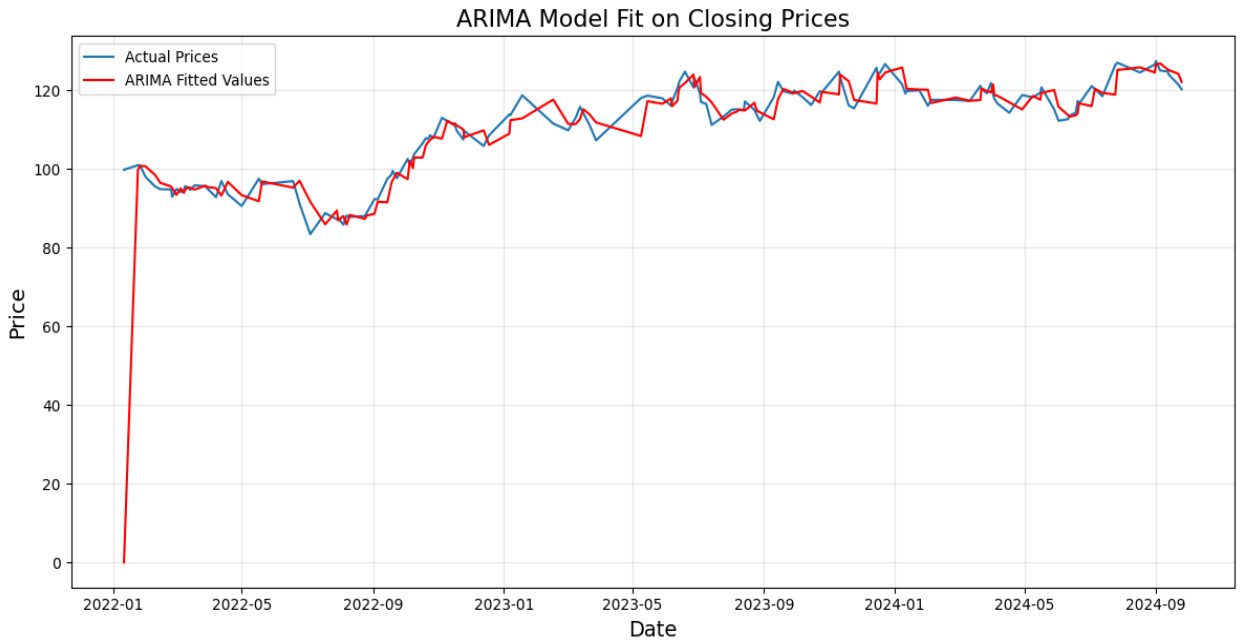


Figure: ARIMA Model Fit on Closing Prices

This line chart illustrates the **performance of an ARIMA model** in fitting the historical **closing prices** of a stock over time.

Components:

- **Blue Line:** Represents the actual observed closing prices.
- **Red Line:** Shows the closing prices predicted by the ARIMA model (fitted values).

Key Observations:

- The ARIMA model closely **tracks the actual price trend**, especially after the initial few data points.
- A significant initial **deviation** is seen due to the model's need to initialize using past data (lag structure).
- Throughout most of the timeline, the **predicted values align well** with the actual prices, suggesting **good model performance** in capturing the trend and short-term fluctuations.

Conclusion:

The ARIMA model effectively fits the closing prices, demonstrating its capability in **time series forecasting**. However, the early deviation highlights the importance of initialization and potential limitations in the beginning of the forecast horizon.

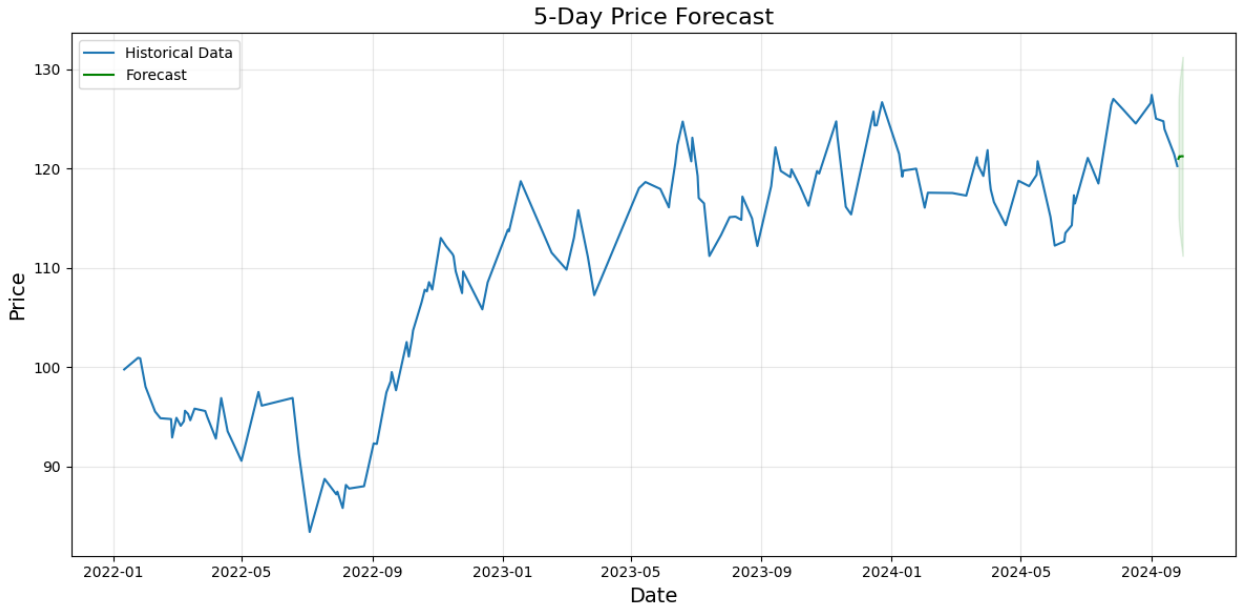


Figure: 5-Day Price Forecast

This line chart displays the historical closing prices of a stock along with a **5-day forecast** using a time series model (likely ARIMA).

Components:

- **Blue Line:** Historical closing prices over time.
- **Green Line:** Predicted prices for the next 5 days.
- **Shaded Area:** Confidence interval around the forecast, representing uncertainty in predictions.

Key Insights:

- The forecast extends smoothly from the recent trend, indicating the model's ability to **capture short-term momentum**.
- The narrow confidence band suggests a **high level of confidence** in the forecast.
- This short horizon prediction is useful for **near-term investment or risk management decisions**.

Using decision tree From Machine learning to get the results

1. Data Loading and Preparation

The dataset is read from a CSV file containing chronological stock data. Since machine learning models require numeric inputs, all categorical columns (Sector, Company, and Trend) are encoded using LabelEncoder. Each unique category is mapped to an integer, allowing the model to interpret them correctly.

---

2. Feature and Target Definition

The features (X) are defined as all columns except S.No, Date, and the target variable Trend. The target (y) is the encoded version of the Trend column, which represents the movement of stock prices (like "Up", "Down", or "Stable").

---

3. Train-Test Split

The data is split into training and testing sets using an 80-20 ratio. This is a common practice to train the model on one portion of the data and evaluate it on unseen data to test its performance.

---

4. Model Training

A DecisionTreeClassifier is created and trained on the training dataset. This algorithm creates a tree-like structure by learning decision rules from the features that best split the data at each step to classify the target variable.

---

5. Model Evaluation

The trained model is used to predict the trends on the test dataset. The results are evaluated using:

- **Confusion Matrix**: A heatmap shows how well the model predicted each class. The diagonal values indicate correct predictions.
- **Classification Report**: Includes accuracy, precision, recall, and F1-score for each class. These metrics provide a deeper look into the model's ability to correctly classify each type of trend.

---

6. Decision Tree Visualization

The final step is visualizing the decision tree. Each node in the tree represents a decision made on a feature, leading to a final classification. This makes it easy to interpret how the model is making predictions and which features are the most influential.

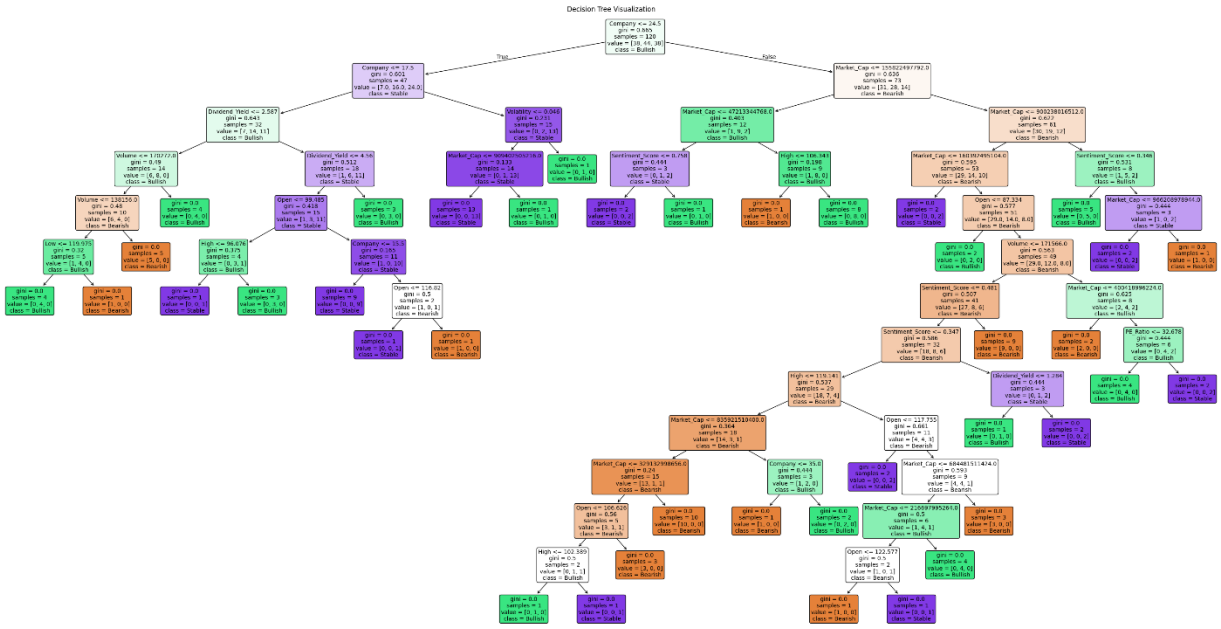


Figure: Decision Tree Classifier for Stock Market Sentiment

This decision tree model classifies stock market sentiment into **Bullish**, **Bearish**, or **Stable**, based on multiple financial and technical indicators.

Key Features Used:

- Market_Cap, Company, Dividend_Yield, Sentiment_Score, Volume, Open, Low, High, Volatility, PE_Ratio

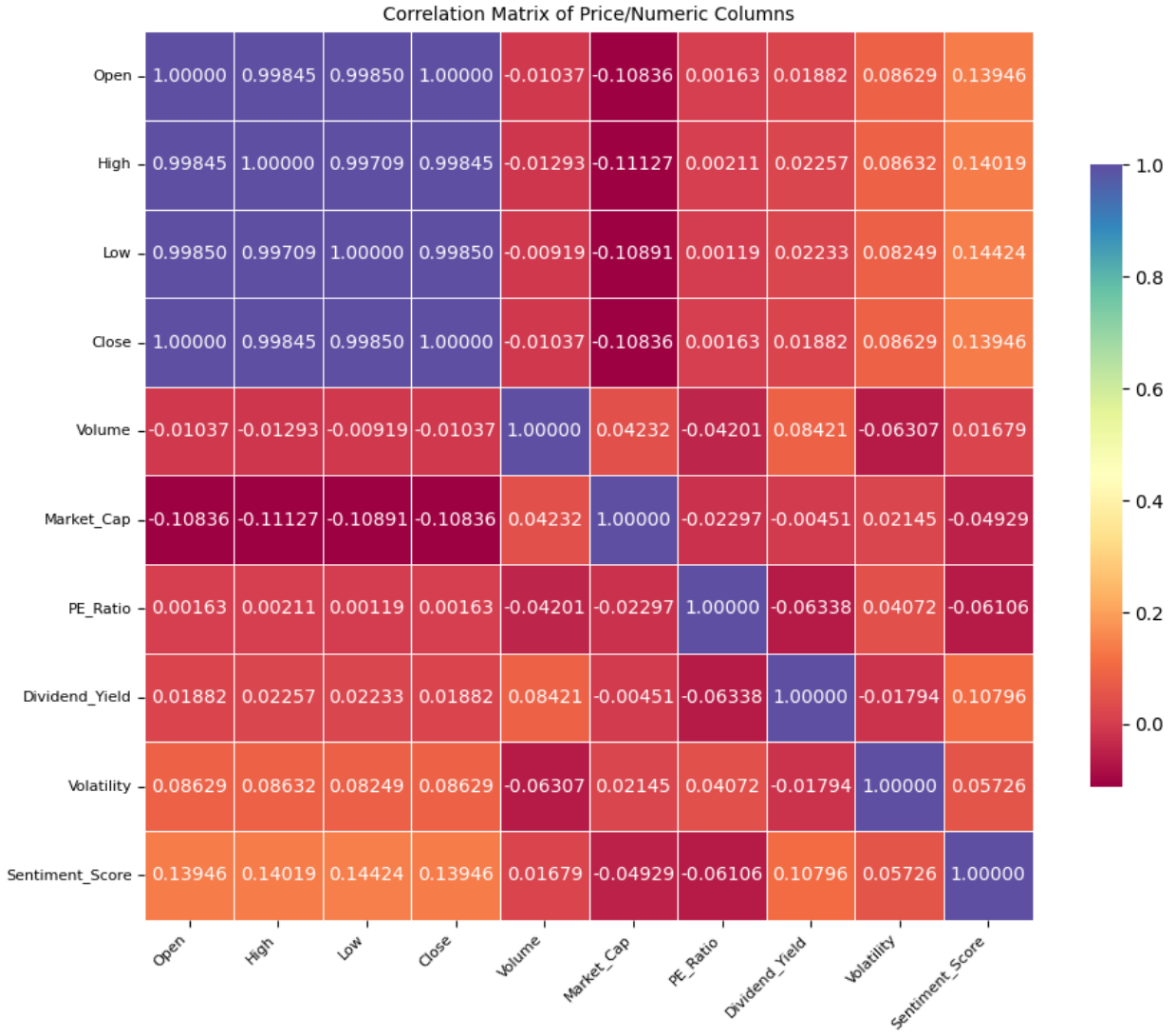
Structure Overview:

- **Root Node:** Splits based on Company, indicating company identity is highly influential.
- **Intermediate Splits:** Factors like Dividend_Yield, Sentiment_Score, and Market_Cap frequently guide decision-making.
- **Leaf Nodes:** Final classification into **Bullish (green)**, **Bearish (orange)**, or **Stable (purple)** classes.

Additional Info:

- Each node shows:
    - gini (impurity)
    - samples (data points at the node)
    - value (distribution across classes)
    - class (predicted sentiment)

CORRELATION MATRIX ***on the data***



Analysis of the Correlation Matrix

This image presents a correlation matrix showing relationships between various financial metrics, including price data and other numeric columns. Here's a detailed explanation:

Structure of the Matrix

The matrix displays correlation coefficients (ranging from -1 to 1) between pairs of variables:

1. **Price-related variables** (Open, High, Low, Close) are shown first
2. **Other financial metrics** follow (Volume, Market Cap, PE Ratio, Dividend Yield, Volatility)
3. **Sentiment Score** is listed separately at the bottom

Key Observations

Strong Correlations (near 1)

- The price columns (Open, High, Low, Close) show nearly perfect correlation with each other (0.998-1.000)
    - This is expected as these prices typically move together throughout a trading day
    - Close price has perfect 1.0 correlation with Open price

Weak/No Correlations (near 0)

- Volume shows almost no correlation with price movements (-0.01 to 0.01)
- Market Cap has very weak negative correlation with prices (-0.11 range)
- PE Ratio shows virtually no correlation with prices (0.001-0.002)

Notable Relationships

- **Sentiment Score** has:
    - Moderate positive correlation with prices (0.139-0.144)
    - Weak positive correlation with Dividend Yield (0.108)
    - Very weak correlations with other metrics
- **Volatility** shows:
    - Small positive correlation with prices (0.082-0.086)
    - Negative correlation with Volume (-0.063)
- **Dividend Yield** has:
    - Positive correlation with Volume (0.084)
    - Negative correlation with PE Ratio (-0.063)

Negative Correlations

- Market Cap shows weak negative correlation with prices
- Volume and PE Ratio have a negative relationship (-0.042)

Interpretation

The matrix reveals that:

1. Price movements are highly internally consistent but largely independent of other metrics
2. Fundamental factors (PE Ratio, Market Cap) show little direct relationship with daily price movements
3. Sentiment has a modest but noticeable relationship with price
4. Most other relationships are either very weak or nonexistent

References

Core Data & Scientific Computing

1. **pandas** – data manipulation and analysis
2. **numpy** – numerical operations
3. **scipy** – statistical functions and distributions (including t-tests, z-tests, etc.)
4. **tabulate** – for displaying tables in a formatted way
5. **IPython.display** – to style and display DataFrames nicely in Jupyter

---

Visualization

6. **seaborn** – for data visualization (likely used for plots; let me know if you want to list all plots too)
7. **matplotlib** *(used implicitly through seaborn; not explicitly imported but often relevant if plotting is done)*

Statistical Modeling

8. **statsmodels.api** – for statistical models (like linear regression)
9. **statsmodels.formula.api** – allows modeling using R-style formulas
10. **statsmodels.stats.anova** – for ANOVA tests
11. **statsmodels.tsa.stattools** – time series tools (e.g., ADF test for stationarity)
12. **statsmodels.tsa.arima.model** – for ARIMA time series forecasting

---

Machine Learning (scikit-learn)

13. **sklearn.model_selection** – for splitting data (e.g., train_test_split)
14. **sklearn.linear_model** – for logistic regression
15. **sklearn.preprocessing** – for data scaling and encoding (StandardScaler, LabelEncoder)
16. **sklearn.metrics** – for evaluating models (accuracy_score, confusion_matrix, classification_report)
17. **sklearn.tree** – for decision trees and visualization (DecisionTreeClassifier, plot_tree)

Book:

- Fundamental of Mathematical Statistics
- Hands on Machine learning with Scikit-learn, keras and tensorFlow

YouTube :

- https://www.youtube.com/playlist?list=PLu0W_9lII9agwh1XjRt242xIpHhPT2llg
- https://www.youtube.com/watch?v=het9HFqo1TQ&list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU&index=5&ab_channel=StanfordOnline

Data Source :

KAGGLE