DATA MINING TECHNIQUES

Project on

STOCK MARKET DATA ANALYSIS AND PREDICTION using DATA MINING TECHNIQUES

Author: Shivani Battu

FINAL REPORT

CONTENTS

Introduction

Predicting stock market trends accurately has always been a highly sought-after skill, offering significant financial benefits to those who can consistently beat the market. Nevertheless, predicting outcomes in financial markets is challenging due to the intricate interplay of economic, political, and psychological variables. Despite the difficulties, researchers have been actively exploring new data-driven methods, using historical data and advanced computational techniques to uncover patterns and relationships that may help predict future price changes. Lately, the combination of data mining techniques and machine learning algorithms has become a promising approach for stock market forecasting. These methods utilize the capacity to extract valuable insights from large, complex datasets, uncover hidden patterns, and model intricate relationships.

Investors and fund managers are constantly seeking ways to optimize their investment strategies and maximize returns. By analyzing historical stock data, patterns and trends can be identified, aiding in the selection of stocks with strong potential for growth or value simultaneously enabling the investors to implement appropriate risk mitigation strategies.

Therefore, vast amounts of data generated by stock exchanges provide a wealth of information that can be leveraged to gain insights into market trends and make informed decisions. In this project, we will delve into the realm of stock market data analysis and prediction, using historical stock prices of companies listed in the S&P 500 index.

Project Description

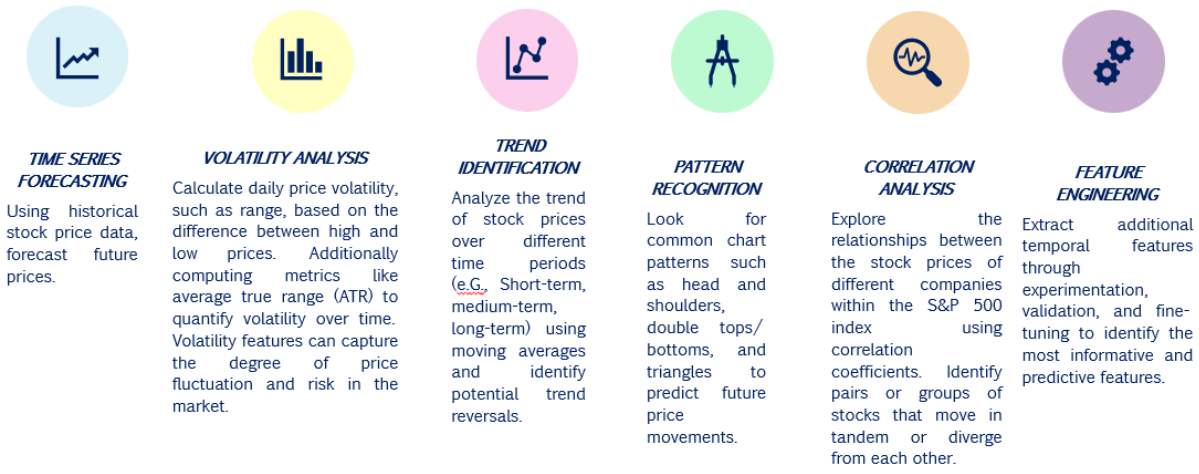
Through this project we would like to train a model for stock market price and use K-means clustering to draw additional conclusions from the data. The dataset we obtained consists of historical stock prices of the companies listed in the S&P 500 index, along with various attributes such as opening price, closing price, high price, low price, trading volume, and more.

We will leverage features such as opening price, closing price, high price, low price, trading volume, and more to train predictive a model that can forecast the direction and magnitude of price movements and provide solutions to the following questions,

- How accurately can we forecast the closing prices of S&P 500 stocks using machine learning algorithms?
- Is there a correlation between the stock prices of different companies within the S&P 500 index?
- What are the most volatile stocks in the S&P 500 index, and how does their volatility compare over time?

Objectives:

- ***Time Series Forecasting:*** Use historical stock price data to forecast future prices.
- ***Volatility Analysis:*** Calculate daily price volatility, such as range, based on the difference between high and low prices.. Volatility features can capture the degree of price fluctuation and risk in the market.
- ***Trend Identification:*** Analyze the trend of stock prices over different time periods (e.g., short-term, medium-term, long-term) using moving averages and identify potential trend reversals.
- ***Pattern Recognition:*** Look for common chart patterns such as head and shoulders, double tops/bottoms, and triangles to predict future price movements.
- ***Correlation Analysis:*** Explore the relationships between the stock prices of different companies within the S&P 500 index using correlation coefficients. Identify pairs or groups of stocks that move in tandem or diverge from each other.
- ***Feature Engineering:*** Extract additional temporal features through experimentation, validation, and fine-tuning to identify the most informative and predictive features.



Process-flow:

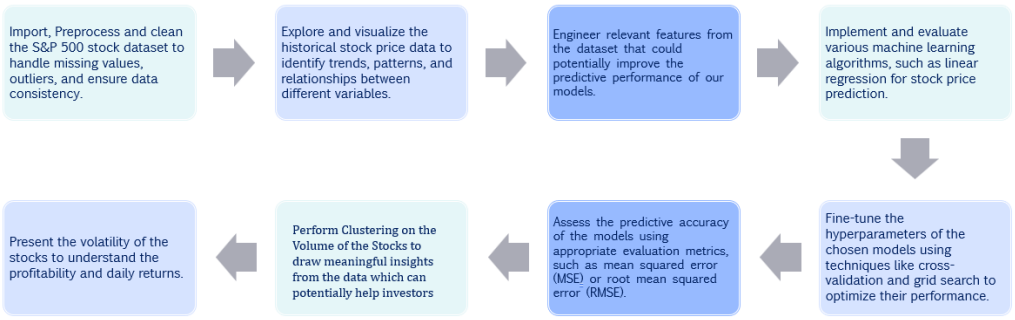
i. Preprocess and clean the S&P 500 stock dataset to handle missing values, outliers, and ensure data consistency.
ii. Explore and visualize the historical stock price data to identify trends, patterns, and relationships between different variables.
iii. Engineer relevant features from the dataset that could potentially improve the predictive performance of our models.
iv. Implement and evaluate various machine learning algorithms, such as linear regression, for stock price prediction.
v. Fine-tune the hyperparameters of the chosen models using techniques like cross-validation and grid search to optimize their performance.

vi. Assess the predictive accuracy of the models using appropriate evaluation metrics, such as mean squared error (MSE) or root mean squared error (RMSE).

vii. Perform Clustering on the Volume of the Stocks to draw meaningful insights from the data which can potentially help investors.

viii. Deploy the best-performing model to predict future stock prices and evaluate its performance on unseen data.

PROCESS FLOW



We as a team plan to collaborate closely with each other by sharing our progress, and provide regular updates on our respective tasks. We have distributed the project tasks and prepared a provisional outline and plan to modulate as needed.

Current layout of the workload distribution is as follows:

Team Member	***Task***	***Description***
Nagavalli Mareedu	Preprocessing and Data Analysis	Preprocess the data to ensure its quality and consistency. This includes handling missing values, outliers, and formatting issues.
Divya Channamallu	Feature Engineering	Conduct feature engineering to create relevant features from the raw dataset. This involves extracting temporal features, calculating price differences, moving averages, technical indicators, volume features, and market sentiment indicators.
Shivani Battu	Model Development and Evaluation	Implement machine learning models for stock price prediction using the engineered features. Experiment and evaluate metrics to identify the best-performing approach.
Gowthami Pakanati	Model Testing, Tuning and Classification:	Test the results on unseen data and perform clustering.

Abhinandh Valaboju	Graphical representation and Documentation	Represent the results graphically. Document the entire project workflow, including data collection, preprocessing steps, feature engineering techniques, model development, evaluation results, and insights gained.
Team (as a whole)	Presentation	Prepare a comprehensive report and presentation summarizing the project findings and recommendations.

Background

- **Related papers:**
  - ✓ Chongda Liu, Jihua Wang, Di Xiao, Qi Liang, "Forecasting S&P 500 Stock Index Using Statistical Learning Models", Department of Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA.
  - ✓ Suryoday Basak, Saibal Kar, Snehanshu Saha, Luckyson Khaidem, Sudeepa Roy Dey,"Predicting the direction of stock market prices using tree-based classifiers", The North American Journal of Economics and Finance, Volume 47, 2019,Pages 552-567, ISSN 1062-9408, https://doi.org/10.1016/j.najef.2018.06.013.

- **Software Tools:**

IDE (Integrated Development Environment): RStudio, JupyterLab.

Python Libraries:

- Numpy: to perform mathematical operations on data
- Pandas: to manipulate dataset
- Plotly: to plot data.
- Scipy: to train and build Machine learning Models.NumPy, scikit-learn, TensorFlow, and matplotlib/seaborn.

- **Required Hardware:**

A standard desktop or laptop computer with sufficient RAM

- **Related programming Skills:**
  - Python Programming Language.
  - Familiarity with above mentioned Libraries.
  - Basic knowledge of Machine Learning Models.
  - Basic Knowledge of statistics.

Problem Definition & Proposed Techniques

- ***Feature Engineering:*** Extracting relevant features from the raw dataset, including historical prices, technical indicators, volume features, and market sentiment indicators, to capture informative signals for stock price prediction.
- ***Model Selection and Evaluation:*** Experimenting with machine learning algorithms (e.g., linear regression) and evaluating their performance using appropriate metrics (e.g., mean squared error, root mean squared error, mean absolute error) to identify if the model is suitable model for the task.
- ***Pattern Recognition:*** Looking for common chart patterns such as head and shoulders, double tops/bottoms, and triangles to predict future price movements.
- ***Correlation Analysis:*** Exploring the relationships between the stock prices of different companies within the S&P 500 index using correlation coefficients. Identifying pairs or groups of stocks that move in tandem or diverge from each other.

I. Challenges of Tackling the Problems:

Addressing Noisy and Non-linear Data: Stock market data is noisy and exhibits complex non-linear relationships, making it challenging to extract meaningful patterns and predict future price movements accurately. By applying techniques such as feature scaling and normalization, we can mitigate the effects of noise in the data. Additionally using non-linear models such as decision trees, random forests can be used to capture complex relationship between features.

Market Volatility and Uncertainty: Stock prices are influenced by various factors such as economic indicators, geopolitical events, and investor sentiment, leading to volatility and uncertainty in the market that can affect prediction accuracy thereby highly susceptible to Volatility. By developing robust models that can adapt to changing market conditions by implementing ensemble methods or continuous learning approaches we can potentially handle volatility.

Missing Value Handling: Ensuring data quality, handling missing values, and addressing data inconsistencies are crucial steps in preprocessing stock market data to build reliable predictive models.

Overfitting and Generalization: Avoiding overfitting and ensuring model generalization to unseen data by careful model selection, hyperparameter tuning, and evaluation strategies.

II. Details of Major Techniques:

Linear Model:

Linear regression is a fundamental statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. The linear equation takes the form,

$$Y=\beta_0+ \beta_1X_1+ \beta_2X_2+ \beta_3X_3+ \beta_4X_4+....+ \beta_nX_n+ \epsilon$$

Where Y is the dependent variable and X_1, X_2, X_3,...., X_n, are the independent variables.

β_1, β_2, β_3, β_4,....β_n are the coefficients, and ϵ represents the error term. The goal of linear regression is to estimate the coefficients that best fit the observed data points, allowing for prediction and inference.

Clustering:

Clustering is a machine learning technique used to group similar data points into clusters or segments based on their intrinsic characteristics or features. The objective of clustering is to partition the data such that data points within the same cluster are more similar to each other than to those in other clusters. Common clustering algorithms include k-means clustering, hierarchical clustering, and density-based clustering. Clustering is often used for exploratory data analysis, pattern recognition, and customer segmentation in various fields such as marketing, biology, and finance.

Volatility:

Volatility refers to the degree of variation or dispersion in the returns of a financial instrument over a specific period of time. It is a measure of the uncertainty or risk associated with an investment and is commonly used in financial markets to assess price fluctuations. Volatility can be calculated using various methods, including historical volatility (based on past price movements), implied volatility (derived from option prices), and realized volatility (based on actual price changes). High volatility implies greater uncertainty and potential for large price swings, while low volatility suggests stability and predictability in asset prices. Understanding and managing volatility is crucial for risk management, portfolio optimization, and investment decision-making.

Visual Application

I. DATASET DESCRIPTION:

```
#Import Dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Import the stock dataset
stock_data = pd.read_csv("My Usage/MS Coursework/Sem2/Data Mining Techniques/Assignments/Group Project/all_stocks_5yr.csv")
```

```
#Printing the head
print("First few rows of the dataset:")
print(stock_data.head())
```

```
First few rows of the dataset:
         date   open   high    low  close    volume Name
0  2013-02-08  15.07  15.12  14.63  14.75   8407500  AAL
1  2013-02-11  14.89  15.01  14.26  14.46   8882000  AAL
2  2013-02-12  14.45  14.51  14.10  14.27   8126000  AAL
3  2013-02-13  14.30  14.94  14.25  14.66  10259500  AAL
4  2013-02-14  14.94  14.96  13.16  13.99  31879900  AAL
```

The dataset used consisted of historical stock market data, specifically focusing on stocks included in the S&P 500 index.

Data Source: The dataset is sourced from Kaggle where it was originally drawn from an open-source financial data provider, likely Yahoo Finance, known for providing comprehensive historical stock market data.

Content: The dataset contains information about various stocks, including their historical prices, trading volumes, and possibly other financial metrics such as open, high, low, and close prices from the period 2013 to 2018

Granularity: The dataset is organized at a daily frequency, with each data point representing a single trading day. This granularity allows for the analysis of daily fluctuations and trends in stock prices.

Variables: Key variables in the dataset include:

Date: The date of each trading day.

Name: Identifiers for individual stocks included in the S&P 500 index.

Open, High, Low, Close Prices: The opening, highest, lowest, and closing prices of each stock on a given trading day.

Volume: The trading volume or the total number of shares traded for each stock on a particular day.

II. PRE-PROCESSING

The dataset underwent preprocessing steps such as normalization to remove noise and ensure consistency across different stocks. Missing values were also handled using appropriate techniques.

Data Transformation: Converting the dataset into a suitable format for analysis. This includes pivoting the data to have stocks as columns and dates as rows, as seen in some clustering examples. The 'date' column was split into Year, Month and Day which helped in visualization and grouping data.

Data Normalization: Scaling the data to a common range or distribution to ensure consistency and facilitate model training. StandardScaler from scikit-learn is used to normalize volume data in one of the clustering examples.

Missing Value Handling: Handling missing values by filling them with zeros or using other appropriate techniques. Rows with Missing values were removed from the dataset and additional checks such as filling them with zeros before normalization.

Feature Selection: Choosing relevant features or variables for analysis and modeling. For example, selecting 'close' prices for clustering or 'open' prices for regression modeling.

Model Evaluation: Assessing the performance of predictive models using various metrics such as mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). These metrics provide insights into the accuracy and effectiveness of the models in predicting stock prices.

```
*[4]: #Pre-processing Data - Step 1
print("Count of null values:")
#Count of the null values
print(stock_data.isna().sum())

# Remove NA values
stock_data.dropna(inplace=True)

# Get the type of each variable (column) in the DataFrame
variable_types = stock_data.dtypes

# Print the type of each variable
print("The data type of each variable is as follows")
print(variable_types)
print(stock_data.head())
```

```
Count of null values:
date       0
open      11
high       8
low        8
close      0
volume     0
Name       0
dtype: int64
The data type of each variable is as follows
date       object
open      float64
high      float64
low       float64
close     float64
volume      int64
Name       object
dtype: object
         date   open   high    low  close    volume Name
0  2013-02-08  15.07  15.12  14.63  14.75   8407500  AAL
1  2013-02-11  14.89  15.01  14.26  14.46   8882000  AAL
2  2013-02-12  14.45  14.51  14.10  14.27   8126000  AAL
3  2013-02-13  14.30  14.94  14.25  14.66  10259500  AAL
4  2013-02-14  14.94  14.96  13.16  13.99  31879900  AAL
```

```
[5]: # Identify total unique stocks in 'Name'
total_unique_stocks = stock_data['Name'].nunique()
print("Total unique stocks in the data are:")
print(total_unique_stocks)
```

```
Total unique stocks in the data are:
505
```

```
[21]: #Preprocessing-Step 2  : Convert the 'Date' column to datetime object
stock_data['date'] = pd.to_datetime(stock_data['date'])

# Split the 'Date' column into 'Year', 'Month', and 'Day' columns
stock_data['Year'] = stock_data['date'].dt.year
stock_data['Month'] = stock_data['date'].dt.month
stock_data['Day'] = stock_data['date'].dt.day
print(stock_data.head())
```

```
        date   open   high    low  close    volume Name  Year  Month  Day
0 2013-02-08  15.07  15.12  14.63  14.75   8407500  AAL  2013      2    8
1 2013-02-11  14.89  15.01  14.26  14.46   8882000  AAL  2013      2   11
2 2013-02-12  14.45  14.51  14.10  14.27   8126000  AAL  2013      2   12
3 2013-02-13  14.30  14.94  14.25  14.66  10259500  AAL  2013      2   13
4 2013-02-14  14.94  14.96  13.16  13.99  31879900  AAL  2013      2   14
```

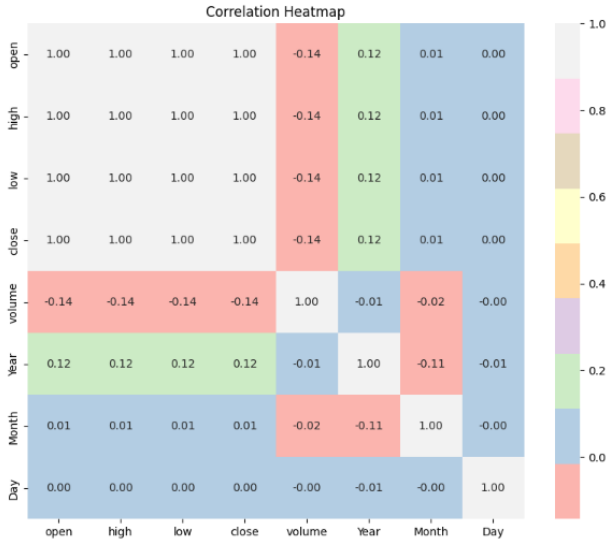
III. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is crucial for understanding the characteristics and patterns within a dataset before diving into modeling or analysis. While specific EDA steps may vary depending on the dataset and objectives, here are some common exploratory analyses that might have been performed:

By conducting these exploratory analyses, we gained a deeper understanding of the dataset's characteristics, and identified patterns or anomalies, and inform subsequent modeling or analysis decisions.

Correlation Analysis: Computing correlation coefficients variables helps identify relationships and dependencies among variables. From our analysis, we can clearly understand the high correlation between the price variables is an indication of the linearity between the values. There seems to be low correlation between year and prices while negative correlation between the volume and other variables.

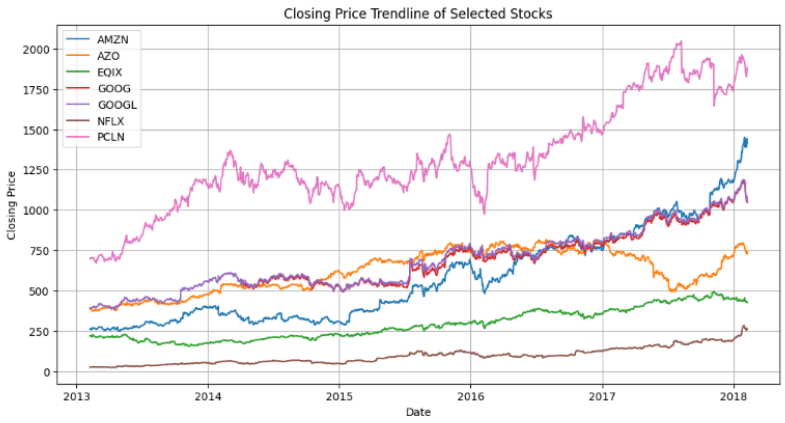
```
[202]: #Correlation heatmap


# EDA: Correlation Heatmap
correlation_matrix = stock_data.drop(columns=['date','Name']).corr()  # Exclude 'Date' column from correlation calculation
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='Pastel1', fmt=".2f", square=True)

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

plt.title('Correlation Heatmap')
plt.show()

#Values for cmap=Blues,RdBu,coolwarm, Pastel1, Set3, tab10
```



Closing Price-trendline for 5 stocks:

From this analysis we compared the closing prices of about 7 stocks which gave insights into their closing patterns and trend over the years. This also provided information about the stability, rise and drop of stocks.

```
[203]: #Closing price trendline for 5 stocks
import pandas as pd
import matplotlib.pyplot as plt

# Select stocks for comparison
selected_stocks = ['AMZN', 'GOOGL', 'GOOG','PCLN', 'AZO', 'NFLX','EQIX']

# Filter data for selected stocks
selected_stock_data = stock_data[stock_data['Name'].isin(selected_stocks)]

# Convert 'Date' column to datetime
selected_stock_data['date'] = pd.to_datetime(selected_stock_data['date'])

# Set 'Date' as index
selected_stock_data.set_index('date', inplace=True)

# Plot line charts for selected stocks
plt.figure(figsize=(12, 6))
for stock_name, group in selected_stock_data.groupby('Name'):
    plt.plot(group.index, group['close'], label=stock_name)

plt.title('Closing Price Trendline of Selected Stocks')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.legend()
plt.grid(True)
plt.show()
```

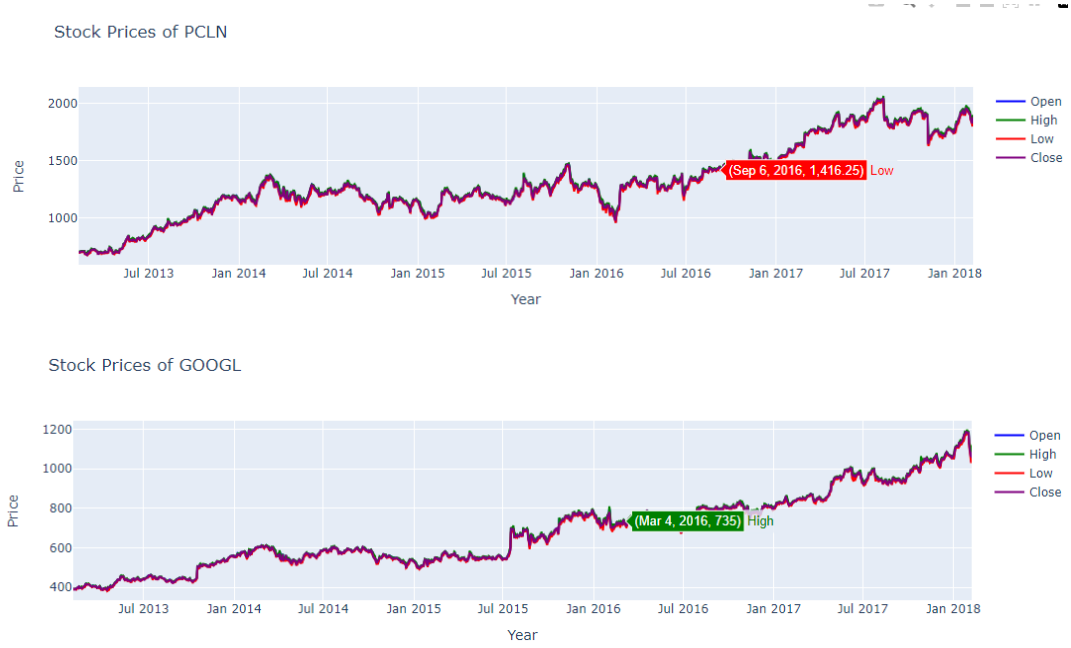


Drawing the trendline graph for the stock closing price over the years 2013 to 2018 shows the companies with the highest closing prices.

The highest closing price among the dataset is that of PCLN (Priceline Group) followed by AMZN (Amazon.Inc) , GOOGL & GOOG (Alphabet Inc), and AZO (Autozone)

Time Series Visualization: Plotting time series of stock prices to visualize trends, seasonality, and fluctuations over time.

```
[230]: #Dynamic Interactive Graph for both PCLN and GOOGL -Open,Close, High and Low line graph
import plotly.graph_objs as go

def plot_stock_prices(stock_data, stock_name):
    # Filter data for the specified stock
    stock_data = stock_data[stock_data['Name'] == stock_name]
    stock_data['date'] = pd.to_datetime(stock_data['date'])

    # Set 'date' as index
    stock_data.set_index('date', inplace=True)

    # Create traces for each stock price component
    trace_open = go.Scatter(x=stock_data.index, y=stock_data['open'], mode='lines', name='Open', line=dict(color='blue'))
    trace_high = go.Scatter(x=stock_data.index, y=stock_data['high'], mode='lines', name='High', line=dict(color='green'))
    trace_low = go.Scatter(x=stock_data.index, y=stock_data['low'], mode='lines', name='Low', line=dict(color='red'))
    trace_close = go.Scatter(x=stock_data.index, y=stock_data['close'], mode='lines', name='Close', line=dict(color='purple'))

    # Create layout
    layout = go.Layout(
        title=f'Stock Prices of {stock_name}',
        xaxis=dict(title='Year'),
        yaxis=dict(title='Price'),
        hovermode='closest'
    )

    # Plot the time series of stock prices using Plotly
    fig = go.Figure(data=[trace_open, trace_high, trace_low, trace_close], layout=layout)
    fig.show()
    # Format x-axis dates
    axes[row, col].xaxis.set_major_formatter(mdates.DateFormatter('%m-%Y'))

# Call the function with the stock data and the desired stock names
plot_stock_prices(stock_data, 'PCLN')
plot_stock_prices(stock_data, 'GOOGL')
```



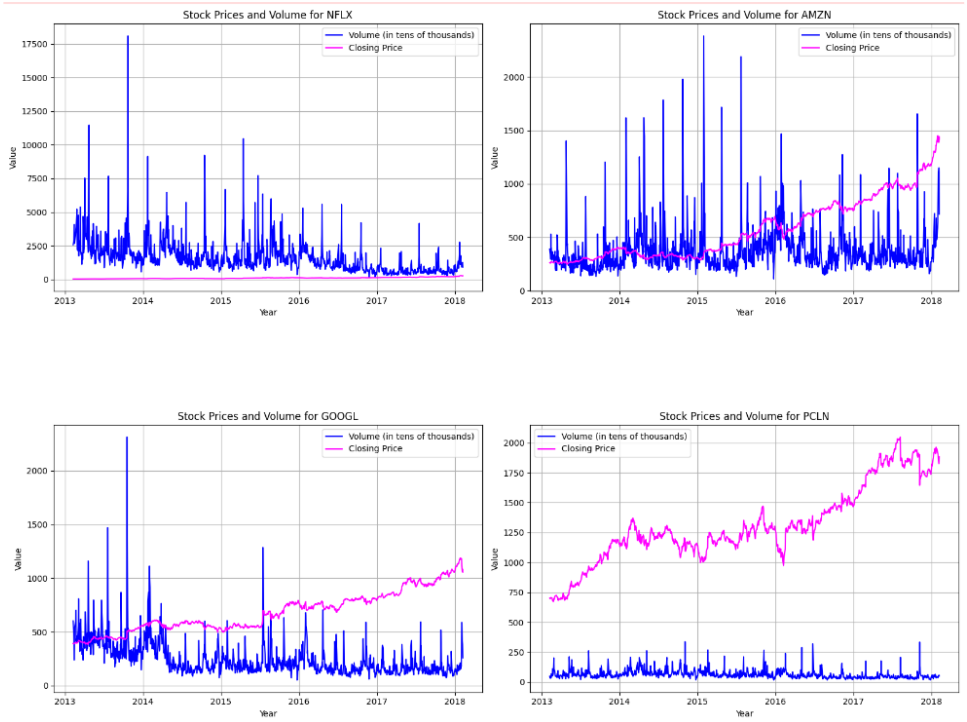
Time series visualization of stock prices (open, high, low, close) for each stock over the 5-year period for PCLN and GOOGL indicates that the prices are highly collinear with each other.

FEATURE ENGINEERING

Feature engineering is the process of selecting, transforming, or creating new features from raw data to improve the performance of machine learning models. It involves identifying and extracting relevant information from the original dataset to create input features that are more informative and suitable for the learning algorithm.

Distribution and Volume Analysis: Examining the distribution of stock prices and volumes helped in understanding the periods of high or low trading activity providing insights into liquidity and market participation.

```
[219]: #Stock closing price fluctuations against Volume(in tens of thousands)
# Create subplots for each stock as a 2x2 matrix
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

# Filter data for the selected stocks
selected_stocks_data = stock_data[stock_data['Name'].isin(['NFLX', 'AMZN', 'GOOGL', 'PCLN'])]

# Convert 'date' column to datetime
selected_stocks_data['date'] = pd.to_datetime(selected_stocks_data['date'])

# Set 'date' as index
selected_stocks_data.set_index('date', inplace=True)

# Plot volume and closing values for each stock
for i, stock_name in enumerate(['NFLX', 'AMZN', 'GOOGL', 'PCLN']):
    row = i // 2  # Calculate row index
    col = i % 2   # Calculate column index

    stock_data_subset = selected_stocks_data[selected_stocks_data['Name'] == stock_name]
    axes[row, col].plot(stock_data_subset.index, stock_data_subset['volume'] / 1e4, color='blue', label='Volume (in tens of thousands)')
    axes[row, col].plot(stock_data_subset.index, stock_data_subset['close'], color='magenta', label='Closing Price')
    axes[row, col].set_title(f'Stock Prices and Volume for {stock_name}')
    axes[row, col].set_xlabel('Year')
    axes[row, col].set_ylabel('Value')
    axes[row, col].legend()
    axes[row, col].grid(True)

# Adjust layout and scale
plt.tight_layout()
plt.subplots_adjust(hspace=0.5)  # Adjust vertical spacing between subplots
plt.show()
```



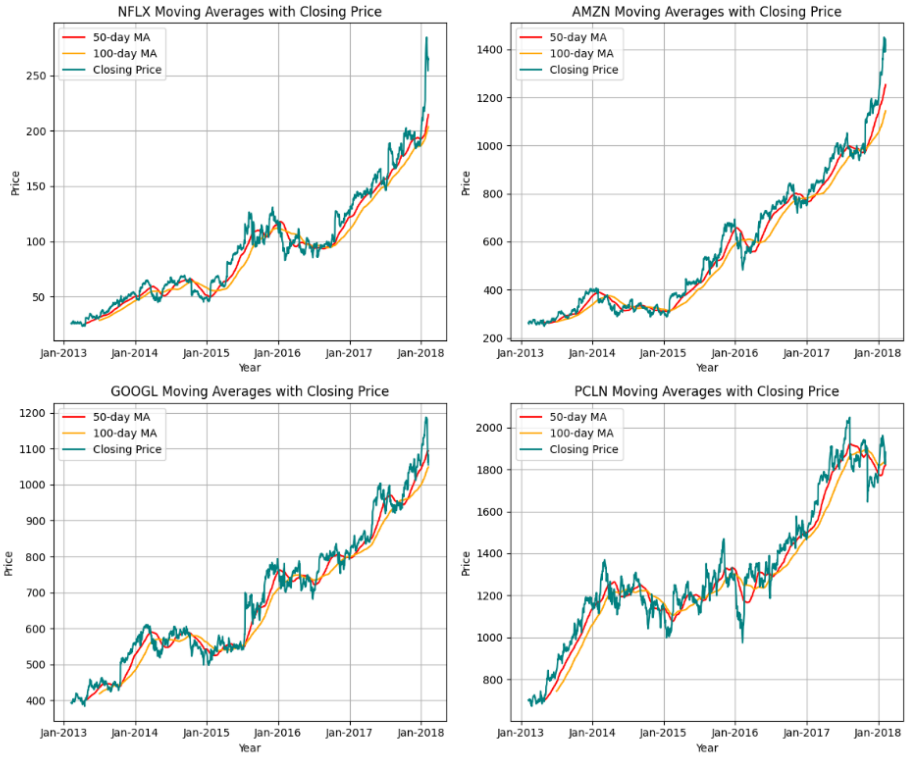
The volume-closing price graph illustrates the relationship between trading volume and closing prices for a specific stock or set of stocks over a given period. Typically, the x-axis represents time (e.g., dates), while the y-axis displays both trading volume and closing prices, with separate scales for each. High trading volume suggests increased market activity and investor interest, while changes in closing prices indicate shifts in market sentiment and supply-demand dynamics. By analyzing this graph, we can identify patterns such as volume spikes accompanying price movements, divergence between volume and price trends, or clustering of volume around key price levels, all of which can provide insights into potential market trends, investor behavior, and trading opportunities.

Seasonal Decomposition: Decomposing time series data into trend, seasonal, and residual components using methods like moving averages. This helped separate the underlying patterns from noise.

The moving averages graph presented a visualization of the moving average trends alongside the actual closing prices of multiple stocks over a defined period. Typically, this graph displays two or more moving averages, such as the 50-day and 100-day moving averages, along with the actual closing prices of the stocks. Moving averages smooth out price data over a specified time frame, providing insights into the overall trend direction and potential reversal points. By comparing the moving averages to the actual closing prices, we can assess the momentum and direction of the market, identify potential entry or exit points based on moving average crossovers, and gauge the strength of prevailing trends.

```
[224]: import matplotlib.pyplot as plt
       import matplotlib.dates as mdates  # Import for date formatting


       def plot_multi_stock_moving_averages_with_close_custom_colors(stock_data, stock_names, window_sizes):
           fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))

           # Define colors for the moving averages
           ma_colors = ['red', 'orange']  # Custom colors for 50-day and 200-day moving averages

           for i, stock_name in enumerate(stock_names):
               row = i // 2  # Calculate row index
               col = i % 2   # Calculate column index

               # Filter data for the specified stock
               stock_data_subset = stock_data[stock_data['Name'] == stock_name]
               stock_data_subset['date'] = pd.to_datetime(stock_data_subset['date'])

               # Set 'date' as index
               stock_data_subset.set_index('date', inplace=True)

               for j, window_size in enumerate(window_sizes):
                   # Calculate moving average
                   moving_average = stock_data_subset['close'].rolling(window=window_size).mean()

                   # Plot moving average with custom color
                   axes[row, col].plot(stock_data_subset.index, moving_average, label=f'{window_size}-day MA', color=ma_colors[j])

               # Plot individual stock closing pattern
               axes[row, col].plot(stock_data_subset.index, stock_data_subset['close'], label='Closing Price', color='teal')

               # Add labels and legend
               axes[row, col].set_title(f'{stock_name} Moving Averages with Closing Price')
               axes[row, col].set_xlabel('Year')
               axes[row, col].set_ylabel('Price')
               axes[row, col].legend()
               axes[row, col].grid(True)

               # Format x-axis dates
               axes[row, col].xaxis.set_major_formatter(mdates.DateFormatter('%b-%Y'))

           # Adjust layout
           plt.tight_layout()

           # Show plot
           plt.show()

       # Call the function with the stock data, stock names, and window sizes
       plot_multi_stock_moving_averages_with_close_custom_colors(stock_data, ['NFLX', 'AMZN', 'GOOGL', 'PCLN'], [50, 100])
```

- In the code, we first calculate the 50-day and 100-day moving averages based on the 'close' for selected stocks using the rolling() function.
- Plot the moving averages against each closing price.
- The libraries needed for this include *numpy* and *matplotlib*



- **Moving averages** help identify the direction of the overall trend in stock prices while smoothing out short-term fluctuations.
- An upward-sloping moving average such as the one in PCLN (2016-2017) suggests a **bullish trend**, while a downward-sloping moving average (NFLX 2016-2017) indicates a **bearish trend**.
- When the price is above the moving average (AMZN 2015 -2016), it may act as **support**, and when the price is below the moving average (NFLX 2016-2017), it acts as **resistance**.
- The crossing of short-term and long-term moving averages creates change in trend direction.
- A **golden cross** occurs when a short-term moving average crosses above a long-term moving average, indicating a bullish signal. such as GOOGL 2015-2016. Conversely, a **death cross** occurs when a short-term moving average crosses below a long-term moving average, indicating a bearish signal such as such as PCLN 2016.

Correlation Analysis: Computing correlation coefficients between different stocks' prices and volume. Correlation matrices with heatmap visualizations help identify relationships and dependencies among variables.

Cross-Sectional Analysis: Analyzing cross-sectional relationships between different stocks or sectors using scatter plots reveal patterns such as sector correlations or clustering tendencies.

The correlation heatmap between the 10 stocks we used provides a visual representation of the pairwise correlations between their closing prices. Each cell in the heatmap represents the correlation coefficient between two stocks, ranging from -1 to 1. A correlation coefficient close to 1 indicates a strong positive correlation, meaning that the two stocks tend to move in the same direction. Conversely, a coefficient close to -1 indicates a strong negative correlation, suggesting that the two stocks move in opposite directions. A coefficient close to 0 indicates little to no linear relationship between the stocks.

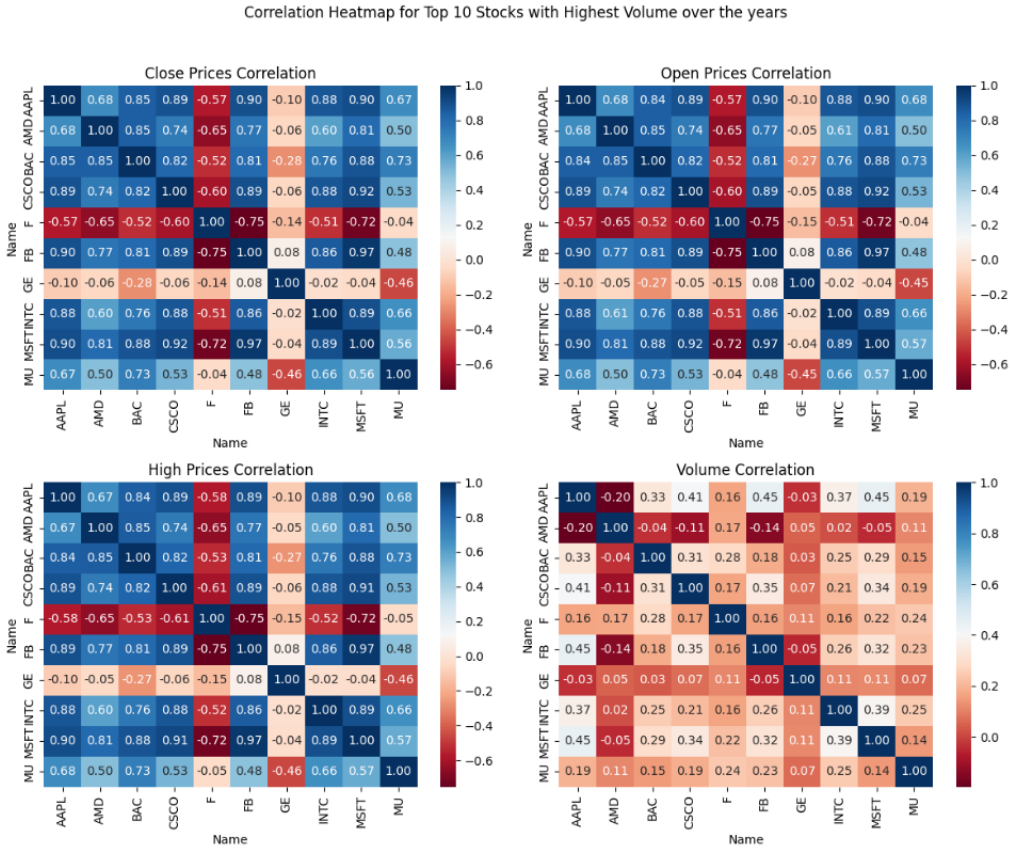
The first step in plotting this graph is to identify the 10 stocks which have the highest volume over the years. Then plotting a correlation heatmap for their close, high, open and volume indicates that there is high correlation between stocks on the prices but very little correlation between stocks based on their volume.

```
[209]: #Unique Stocks having highest Closing Price correlation
import pandas as pd

# Assuming you have loaded your dataset into a DataFrame named 'stock_data'

# Filter the DataFrame to include only the closing prices of the stocks
closing_prices = stock_data.pivot(index='date', columns='Name', values='close')

# Calculate the correlation matrix
correlation_matrix = closing_prices.corr()

# Exclude self-correlations (correlation of a stock with itself)
correlation_matrix_filtered = correlation_matrix.where(~np.tril(np.ones(correlation_matrix.shape, dtype=bool)))

# Find the top 10 highest correlations
top_10_correlations = correlation_matrix_filtered.unstack().sort_values(ascending=False).head(10)

# Display the top 10 highest correlations
print("Top 10 Highest Unique Correlations (excluding self-correlations and duplicates):")
print(top_10_correlations)
```

```
Top 10 Highest Unique Correlations (excluding self-correlations and duplicates):
Name  Name
GOOGL GOOG    0.999085
DISCK DISCA   0.998131
XEL   CMS     0.993641
UAA   UA      0.992877
FOXA  FOX     0.992754
NOC   LMT     0.991380
UNH   CTAS    0.989285
TXN   ITW     0.989052
MMC   AON     0.989003
WEC   CMS     0.988950
dtype: float64
```

```
[210]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have loaded your dataset into a DataFrame named 'stock_data'

# Group the data by stock name and calculate the total volume
stock_volume = stock_data.groupby('Name')['volume'].sum()

# Select the top 20 stocks with the highest volume
top_10_stocks = stock_volume.nlargest(10).index

# Filter the DataFrame to include only the data of the top 20 stocks
top_10_stock_data = stock_data[stock_data['Name'].isin(top_10_stocks)]

# Create subplots for close, open, high, and volume
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))

# Plot close prices
sns.heatmap(top_10_stock_data.pivot(index='date', columns='Name', values='close').corr(), ax=axes[0, 0], cmap='RdBu', annot=True, fmt=".2f")
axes[0, 0].set_title("Close Prices Correlation")

# Plot open prices
sns.heatmap(top_10_stock_data.pivot(index='date', columns='Name', values='open').corr(), ax=axes[0, 1], cmap='RdBu', annot=True, fmt=".2f")
axes[0, 1].set_title("Open Prices Correlation")

# Plot high prices
sns.heatmap(top_10_stock_data.pivot(index='date', columns='Name', values='high').corr(), ax=axes[1, 0], cmap='RdBu', annot=True, fmt=".2f")
axes[1, 0].set_title("High Prices Correlation")

# Plot volume
sns.heatmap(top_10_stock_data.pivot(index='date', columns='Name', values='volume').corr(), ax=axes[1, 1], cmap='RdBu', annot=True, fmt=".2f")
axes[1, 1].set_title("Volume Correlation")

# Set the title of the entire figure
fig.suptitle("Correlation Heatmap for Top 10 Stocks with Highest Volume over the years\n\n")

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

Analyzing the heatmap allowed us to identify patterns of correlation among the stocks. Strong positive correlations indicate that the stocks are influenced by similar market factors or belong to the same industry sector. On the other hand, negative correlations suggest that the stocks are affected by different market dynamics or have contrasting performance drivers. Understanding these correlations have been valuable for portfolio diversification and risk management strategies, as it helps investors identify assets that may behave differently under various market conditions.



Experimental Evaluation

MODEL BUILDING:

Framework:

Once the data has been pre-processed and analysed, it was split into two sets: train and test.

a. The training set is used to train the machine learning model.

b. The testing set is used to evaluate the final performance of the model on data that it has never seen before.

It is important to split the data randomly so that the training and test sets are representative of the overall population. A common 80% training and 20% test split was used.

[211]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# 1. Subset the Dataset
subset_data = stock_data[['Name', 'open', 'close']]

# 2. Handle Missing Values (if any)
subset_data.dropna(inplace=True)

# 3. Split the Data
X = subset_data[['Name', 'open']]
y = subset_data['close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Encode Categorical Variables (if necessary)
X_train_encoded = pd.get_dummies(X_train)
X_test_encoded = pd.get_dummies(X_test)

# 5. Train the Model
model = LinearRegression()
model.fit(X_train_encoded, y_train)

# 6. Make Predictions
y_pred = model.predict(X_test_encoded)

# 7. Evaluate the Model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
C:\Users\shiva\AppData\Local\Temp\ipykernel_17940\3916556847.py:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gui
```

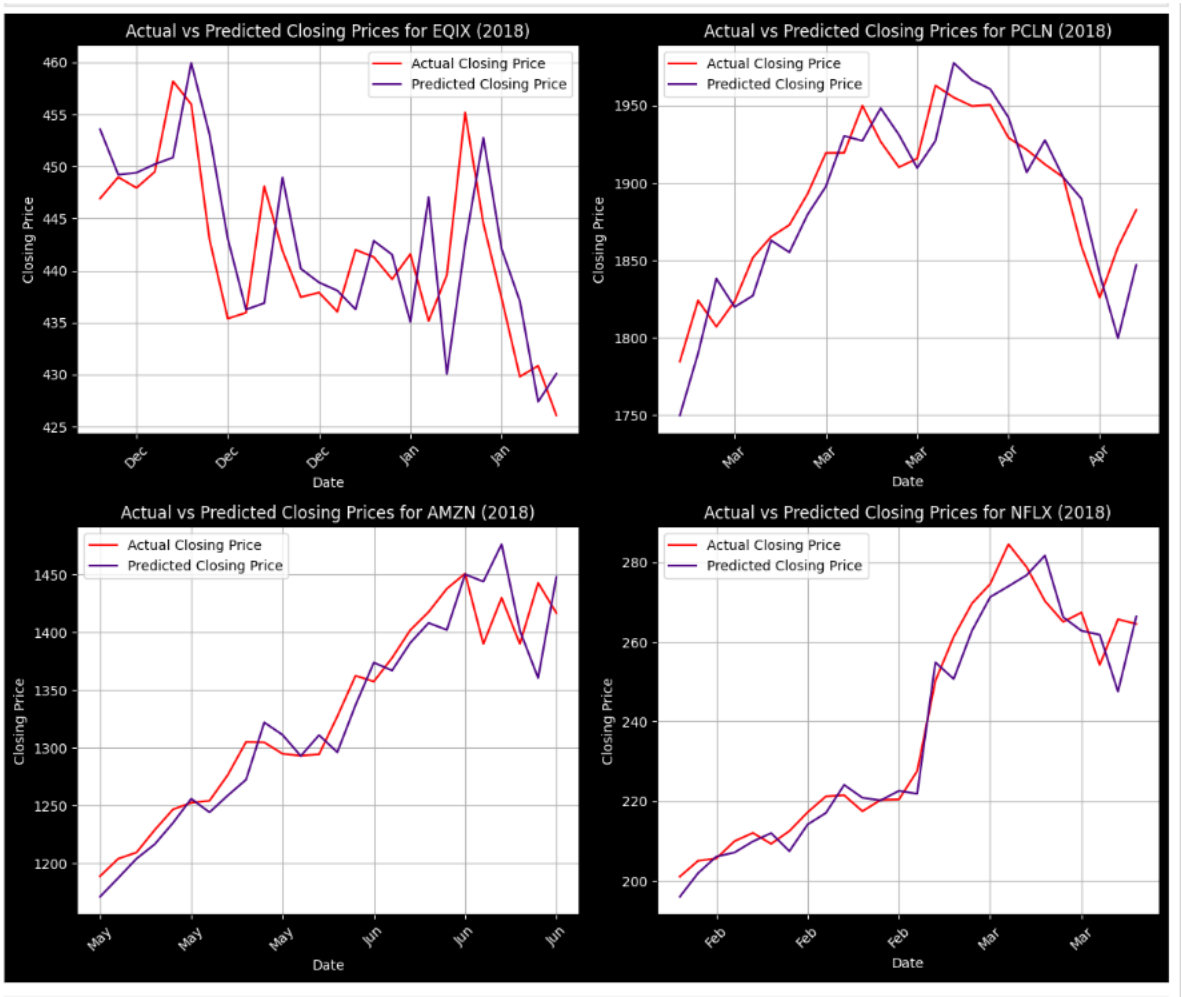
```
Mean Squared Error: 2.3472488909029274
```

[73]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Calculate MAE
mae = mean_absolute_error(y_test, y_pred)

print("RMSE:", rmse)
print("MAE:", mae)
```

```
RMSE: 1.5320733960561184
MAE: 0.7357614864701028
```

[151]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
import matplotlib.dates as mdates

# Assuming you have loaded your data into a DataFrame named 'stock_data'

# Filter data for the training period (2013-2017)
train_data = stock_data[stock_data['date'].dt.year <= 2017]

# Create a list of stock symbols
stocks = ['EQIX', 'PCLN', 'AMZN', 'NFLX']

# Create subplots for each stock
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(12, 10), facecolor='black')

# Initialize lists to store evaluation metrics
test_mse_list = []
test_mae_list = []
test_rmse_list = []

# Iterate over each stock symbol
for i, stock in enumerate(stocks):
    # Filter test data for the current stock symbol
    test_data = stock_data[(stock_data['date'].dt.year == 2018) & (stock_data['Name'] == stock)]

    # Extract features (X: open price) and target (Y: close price)
    X_train = train_data[train_data['Name'] == stock][['open']]
    y_train = train_data[train_data['Name'] == stock]['close']
    X_test = test_data[['open']]
    y_test = test_data['close']

    # Train linear regression model
    regression_model = LinearRegression()
    regression_model.fit(X_train, y_train)

    # Predict closing prices
    y_pred = regression_model.predict(X_test)

    # Calculate evaluation metrics
    test_mse = mean_squared_error(y_test, y_pred)
    test_mae = mean_absolute_error(y_test, y_pred)
    test_rmse = np.sqrt(test_mse)

    # Append evaluation metrics to lists
    test_mse_list.append(test_mse)
    test_mae_list.append(test_mae)
    test_rmse_list.append(test_rmse)

    # Plot actual vs predicted closing prices
    row = i // 2
    col = i % 2
    axs[row, col].plot(test_data.index, y_test, label='Actual Closing Price', color='Red')
    axs[row, col].plot(test_data.index, y_pred, label='Predicted Closing Price', color='Indigo')
    axs[row, col].set_title(f'Actual vs Predicted Closing Prices for {stock} (2018)', color='white')
    axs[row, col].set_xlabel('Date', color='white')
    axs[row, col].set_ylabel('Closing Price', color='white')
    axs[row, col].legend()
    axs[row, col].grid(True)
    axs[row, col].xaxis.set_major_formatter(mdates.DateFormatter('%b'))  # Format x-axis dates
    axs[row, col].tick_params(axis='x', rotation=45)
    axs[row, col].tick_params(colors='white')

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```

- ✓ Split the data into training and testing sets in 80:20 ratio
- ✓ Encoded the Categorical data that is ‘Name’
- ✓ Trained the Model
- ✓ Predicted the closing prices
- ✓ Evaluated the model’s performance based on MSE, RMSE, MAE



The linear regression model constructed in this analysis aimed to predict the closing prices of selected stocks based on their opening prices. Using historical stock data, the model was trained on a subset of the data covering the period from 2013 to 2017. It utilized the relationship between the opening and closing prices to learn a linear function that could estimate the closing price given the opening price. After training, the model was evaluated using data from 2018, and performance metrics such as mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE) were calculated to assess its predictive accuracy.

CLUSTERING:

K-means clustering was applied to group high-volume stocks based on their closing prices.

The process involved selecting an appropriate number of clusters (k=5) and identifying similarities in the volume among different stocks.

- As a first step for Clustering, we calculated the average value of the Volume in the entire dataset.
- Grouped individual stocks and calculated their average volume.
- Extracted the stocks whose volume is greater than average_volume.
- Normalized the 'volume' data using StandardScalar() normalizer.
- Performed k-means clustering based on the Volume with k=5 using the Kmeans() function.

[178]:
```
# Calculate the average volume for the entire dataset
average_volume = stock_data['volume'].mean()
print("Average Volume for the Entire Dataset:", average_volume)
```

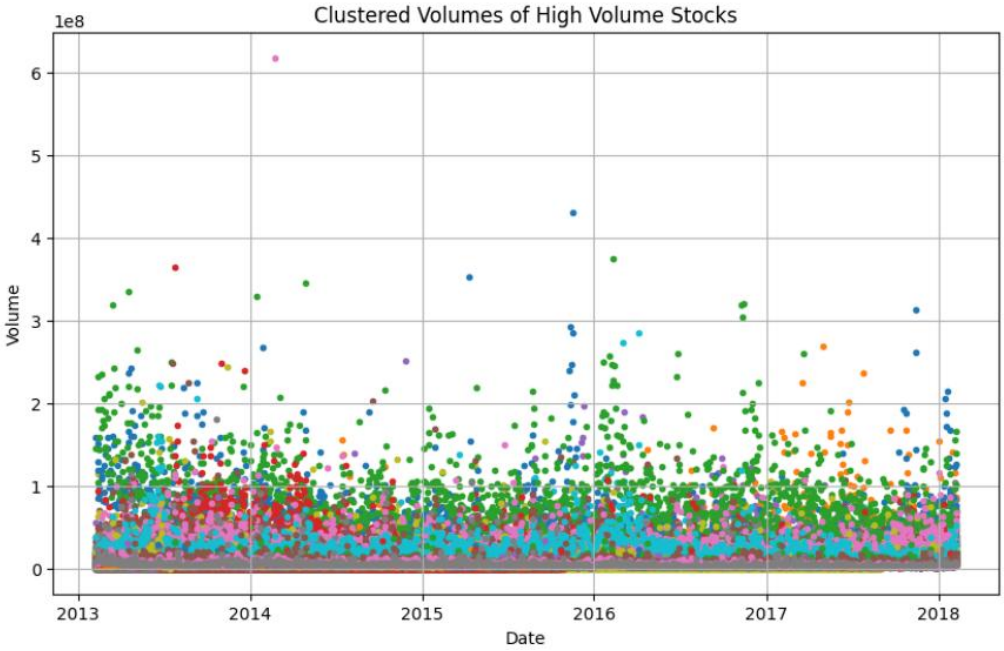
```
Average Volume for the Entire Dataset: 4321891.930248825
```

[174]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Group the data by stock name and calculate average volume
avg_volume_per_stock = stock_data.groupby('Name')['volume'].mean().reset_index()
avg_volume_per_stock.columns = ['Name', 'avg_volume']

# Filter stocks with average volume greater than the threshold
threshold = 4321891.93024882
high_volume_stocks = avg_volume_per_stock[avg_volume_per_stock['avg_volume'] > threshold]

# Extract volume data for high volume stocks
high_volume_stock_data = stock_data[stock_data['Name'].isin(high_volume_stocks['Name'])]

# Pivot the data to have stocks as columns and dates as rows
volume_pivot = high_volume_stock_data.pivot(index='date', columns='Name', values='volume')

# Fill missing values with 0
volume_pivot.fillna(0, inplace=True)

# Normalize the volume data
scaler = StandardScaler()
scaled_volume = scaler.fit_transform(volume_pivot)

# Perform K-means clustering
kmeans = KMeans(n_clusters=5, random_state=42)
cluster_labels = kmeans.fit_predict(scaled_volume.T)  # Transpose to match dimensions

# Plot the clustered volumes as dots
plt.figure(figsize=(10, 6))
for i in range(5):
    cluster_stocks = volume_pivot.columns[cluster_labels == i]
    for stock in cluster_stocks:
        plt.scatter(volume_pivot.index, volume_pivot[stock], label=f'Cluster {i}' if i == 0 else None, marker='.')


plt.xlabel('Date')
plt.ylabel('Volume')
plt.title('Clustered Volumes of High Volume Stocks')
plt.grid(True)
plt.show()
```

Each cluster represents a groups of high-volume stocks based on their trading volume patterns. By utilizing the volume data from the stock market, the clustering algorithm identified similarities in the trading activity among different stocks. The process involved selecting an appropriate number

of clusters (k=5) and partitioning the stocks into distinct groups based on their trading volume behavior over time. Each cluster represents a group of stocks exhibiting similar volume patterns, providing insights into trading activity, liquidity, and investor interest. Analyzing the characteristics of each cluster can offer valuable information about market dynamics, investor sentiment, and potential trading opportunities.

Each cluster represents a group of stocks with similar volume movements, allowing investors to diversify their portfolios by selecting stocks from different clusters.



- This Data Mining Technique helped in forming clusters among the stocks which can be used by stakeholders to make decisions.
- The below figure indicates the different stocks that fall into the 5 clusters over the 5 –year period.

[168]:
```
# Initialize dictionary to store stocks in each cluster
cluster_stocks_dict = {i: [] for i in range(5)}

# Iterate over clusters and stocks
for i in range(5):
    cluster_stocks = volume_pivot.columns[cluster_labels == i]
    # Limit to first 5 stocks in each cluster
    cluster_stocks = cluster_stocks[:5]
    cluster_stocks_dict[i] = cluster_stocks

# Print stocks in each cluster
for cluster, stocks in cluster_stocks_dict.items():
    print(f"Cluster {cluster}: {', '.join(stocks)}")
```

```
Cluster 0: GGP
Cluster 1: ABBV, CBS, CRM, CTSH, HD
Cluster 2: AMD, APC, CFG, CHK, CTL
Cluster 3: AAL, ABT, AES, AMAT, ARNC
Cluster 4: AAPL, AIG, BAC, BBY, BMY
```

VOLATILITY ANALYSIS:

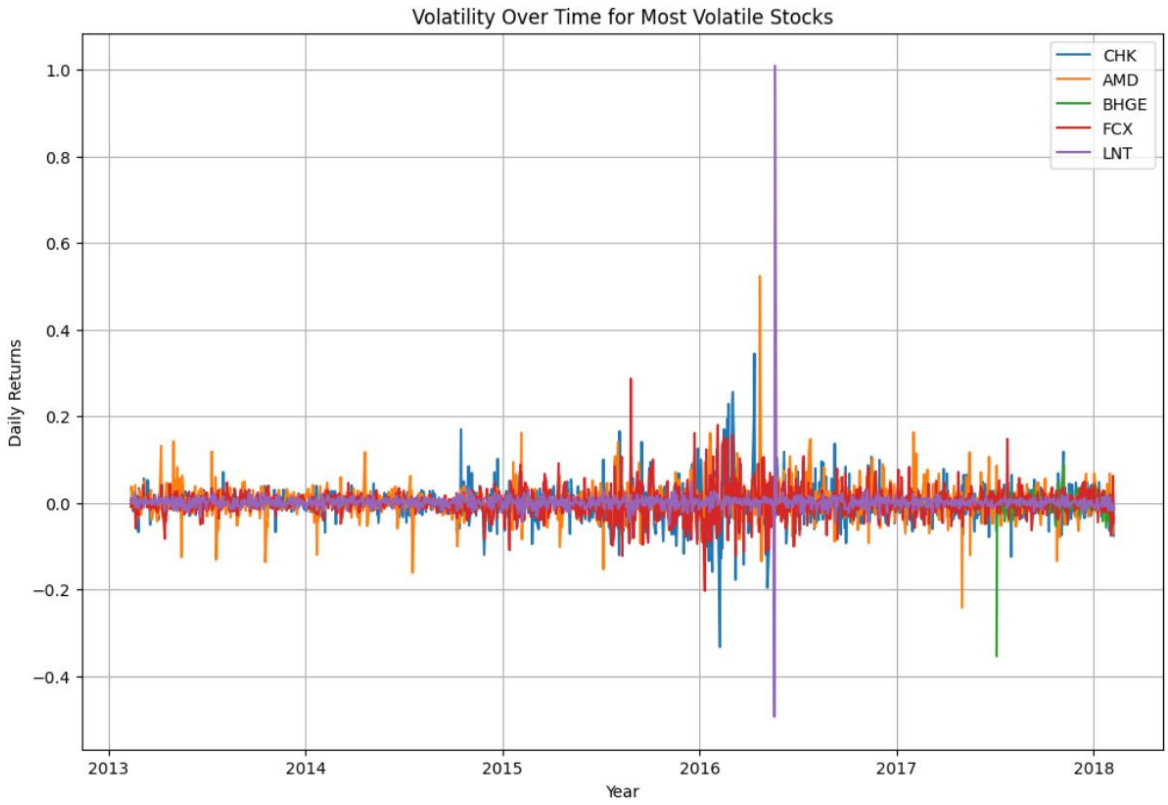
Volatility refers to the degree of variation in stock prices over time, and it is a key metric for investors to gauge the level of risk associated with a particular stock or portfolio.

We assessed volatility in the stock market by calculating the daily returns of the selected stocks. By computing daily returns, which represent the percentage change in stock prices from one day to the next, we were able to quantify the magnitude of price fluctuations. These daily return values were then used to examine the historical volatility of each stock and assess how it compares to others in the dataset. Understanding volatility is crucial for investors to make informed decisions about portfolio allocation, risk management, and trading strategies.

[195]:
```python
#VOLATILITY
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Assuming you have loaded your stock data into a DataFrame named 'stock_data'
# Make sure the DataFrame has columns 'Date' and 'Close' for the date and closing price, respectively

# Calculate daily returns for each stock
stock_data['Daily_Return'] = stock_data.groupby('Name')['close'].pct_change()

# Compute standard deviation of daily returns for each stock
volatility = stock_data.groupby('Name')['Daily_Return'].std()

# Find the top 5 most volatile stocks
most_volatile_stocks = volatility.nlargest(5)

# Extract data for the most volatile stocks
volatile_stock_data = stock_data[stock_data['Name'].isin(most_volatile_stocks.index)]

# Plot volatility over time for the most volatile stocks
plt.figure(figsize=(12, 8))
for stock in most_volatile_stocks.index:
    stock_data_filtered = volatile_stock_data[volatile_stock_data['Name'] == stock]
    plt.plot(stock_data_filtered['date'], stock_data_filtered['Daily_Return'], label=stock)

plt.title('Volatility Over Time for Most Volatile Stocks')
plt.xlabel('Year')
plt.ylabel('Daily Returns')
plt.legend()
plt.grid(True)
plt.show()
```

The daily return rate which we calculated using standard deviation holds the same usage as the original formula for daily returns as stated below.

- $R_t = \frac{(\text{Close Price}_t - \text{Close Price}_{t\_1})}{\text{Close Price}_{t\_1}} \times 100$



Daily returns represent the percentage change in the value of an asset over a single trading day. It indicates how much the value of the asset has changed from the previous day's closing price to the current day's closing price, expressed as a percentage.

Interpreting Volatility:

- A positive daily return indicates that the asset's value increased from the previous day.
- A negative daily return indicates that the asset's value decreased from the previous day.
- The magnitude of the daily return indicates the extent of the change in value thereby determining the Volatility of the Stock. Higher magnitudes (positive or negative) imply larger changes in value.
- Stocks with high Volatility – CHK, AMD, BHGE, FCX, UNT

Future Work

The future scope for our project is vast and includes several potential avenues for further exploration and enhancement. Some of the potential future work includes:

Advanced Modeling Techniques: Exploring advanced machine learning models such as deep learning architectures (e.g., convolutional neural networks, recurrent neural networks with attention mechanisms) to capture complex patterns and dependencies in stock market data more effectively.

Sentiment Analysis: Integrating sentiment analysis of news articles, social media posts, and other textual data related to stocks to gauge market sentiment and its impact on stock prices. Natural language processing (NLP) techniques can be employed for sentiment analysis.

Ensemble Learning: Implementing ensemble learning techniques to combine predictions from multiple models or clustering algorithms to further enhance predictive accuracy and robustness.

Real-Time Prediction: Developing real-time prediction models that can adapt to dynamic market conditions and provide timely insights for traders and investors.

Portfolio Optimization: Integrating portfolio optimization techniques, such as Markowitz portfolio theory or modern portfolio theory, to construct diversified portfolios that maximize returns while minimizing risk.

Interpretability and Explainability: Enhancing the interpretability and explainability of the models to provide actionable insights and build trust among users, stakeholders, and regulators.

Deployment and Integration: Deploying the developed models into production environments and integrating them with trading platforms, investment management systems, or financial applications to enable practical use by investors and financial professionals.

By addressing these areas of future work, the project can contribute to advancing the field of quantitative finance, empowering investors with valuable tools for decision-making, and facilitating more informed and profitable investment strategies in the stock market.

Conclusion

Our analysis delved into various aspects of stock market data, including exploratory data analysis, feature engineering, model building, and clustering. We began by preprocessing the data, which involved handling missing values, normalizing data, and selecting relevant features. Exploratory data analysis provided insights into the trends and relationships within the dataset, including visualizations of stock prices, volumes, moving averages, and correlation heatmaps. We then proceeded to build a linear regression model to predict stock prices, followed by clustering analysis

to group stocks based on their volume patterns. Additionally, we explored volatility by calculating daily returns and assessing the magnitude of price fluctuations. Through these analyses, we gained valuable insights into the behavior of the stock market and potential strategies for investment and portfolio management. Moving forward, further research could explore advanced modeling techniques, sentiment analysis, and alternative data sources to enhance predictive accuracy and decision-making in stock market investments. Some of our key findings are-

- There is a correlation between multiple stock prices of different companies within the S&P 500 index.
- Stocks with high Volatility are CHK, AMD, BHGE, FCX, UNT. Their volatility seems to be fluctuating between (0.2%, -0.2%)
- The accuracy of our LR model prediction can be measured using the following metrics

  MSE: 2.347

  RMSE: 1.532

  MAE: 0.736

  Low error rates indicate that Linear Regression was a good fit for the data and was successful in predicting the closing prices on the unseen test data.

References

1. Makadia, D. R., Kotadia, J., Monasara, A., and Garg, D., "A machine learning method for stock market price prediction," in International Conference on ICT for Sustainable Development. Springer, 2023, pp.217–227.
2. Pawar, K., Jalem, R. S., and Tiwari, V., "Stock market price prediction using lstm rnn," in Emerging Trends in Expert Applications and Security: Proceedings of ICETEAS 2018. Springer, 2019, pp. 493–503.
3. Maulana, R., Nugraha, W., Nurmalasari, N., Latifah, L., and Rahayuningsih, P. A., "Comparison of data mining algorithm in predicting tlkm stock price," in AIP Conference Proceedings, vol. 2714, no. 1. AIP Publishing, 2023.
4. Khedr, A. E., Yaseen, N. et al., "Predicting stock market behavior using data mining technique and news sentiment analysis," International Journal of Intelligent Systems and Applications, vol. 9, no. 7, p. 22, 2017.
5. Nimje, S., Mayya, R., Baig, M. N. A., and Jawale, S., "Prediction on stocks using data mining," in Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST), 2020.

6. Reddy, V. R. and Gayathri, A., "An intelligence prediction of stock market analysis using extreme dynamic data mining techniques predictions." Turkish Online Journal of Qualitative Inquiry, vol. 12, no. 8, 2021.
7. Nti, I. K., Adekoya, A. F., and Weyori, B. A., "A systematic review of fundamental and technical analysis of stock market predictions," Artificial Intelligence Review, vol. 53, no. 4, pp. 3007–3057, 2020.
8. Hu, Z., Liu, W., Bian, J., Liu, X., and Liu, T.-Y., "Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction," in Proceedings of the eleventh ACM international conference on web search and data mining, 2018, pp. 261–269.
9. Kumar, A. and Chaudhry, M., "Review and analysis of stock market data prediction using data mining techniques," in 2021 5th International Conference on Information Systems and Computer Networks (ISCON). IEEE, 2021, pp. 1–10.
10. Yan, X. and Zheng, L., "Fundamental analysis and the cross-section of stock returns: A data-mining approach," The Review of Financial Studies, vol. 30, no. 4, pp. 1382–1423, 2017.
11. Athira, Raj, A., Pushpan, A., and Jisha, R., "Machine learning-based Indian stock market's price movement prediction and trend analysis," in International Conference on Innovations in Computer Science and Engineering. Springer, 2022, pp. 139–152.
12. Madaan, D., Gupta, T., Rani, L., Sahoo, A. K., and Sarangi, P. K., "Comparative study of cnn and lstm on short-term future stock price prediction," in International Conference on Data Analytics & Management. Springer, 2023, pp. 141–157.
13. Guan, B., Zhao, C., Yuan, X., Long, J., and Li, X., "Price prediction in china stock market: an integrated method based on time series clustering and image feature extraction," The Journal of Supercomputing, pp. 1–39, 2023.
14. Yang, X., Cao, Y., and Cheng, X., "Stock's closing price prediction based on gru neural network," in International Conference on Artificial Intelligence, Robotics, and Communication. Springer, 2022, pp. 137–150.
15. Sharma, D. and Kaur, D., "Stock price prediction using long short term memory algorithm," in 2023 4th IEEE Global Conference for Advancement in Technology (GCAT). IEEE, 2023, pp. 1–5.
16. Hong, S., "A study on stock price prediction system based on text mining method using lstm and stock market news." Journal of Digital Convergence, vol. 18, no. 7, 2020.
17. Zhu, C., Kang, L., and Feng, W., "Predicting stock closing price with stock network public opinion based on adaboost-aafsa-elman model and ceemdan algorithm," Journal of Shanghai Jiaotong University (Science), vol. 28, no. 6, pp. 809–821, 2023.
18. Ranaldi, L., Gerardi, M., and Fallucchi, F., "Crypto net: using autoregressive multi-layer artificial neural networks to predict financial time series," Information, vol. 13, no. 11, p. 524, 2022.
19. Alzazah, F. S. and Cheng, X., "Recent advances in stock market prediction using text mining: A survey," E-Business-Higher Education and Intelligence Applications, pp. 1–34, 2020.
20. Alshammari, B. M., Aldhmour, F., AlQenaei, Z. M., and Almohri, H., "Stock market prediction by applying big data mining," Arab Gulf Journal of Scientific Research, vol. 40, no. 2, pp. 139–152, 2022.