Capstone-Project-2:Data Visualization and Inference Modeling-The Case of Nifty

CAPSTONE REPORT

INDUSTRIAL PROJECT BASED LEARNING

Department of Computer Science and Engineering
Accredited by NBA

Geethanjali College of Engineering and Technology
(UGC Autonomous)
(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

Team – 9

Team Details:

Aishwarya Sandlapuram	21R11A6754
Bhavani Vuppuloju	21R11A6765
Deekshitha Rayabandi	21R11A6752
Sreekaree Pachalla	21R11A6743
Siddu Siddarth Banoth	22R11A6702

ABSTRACT

This project embarks on a meticulous exploration of stock market data, delving into the nuances of Nifty monthly and yearly returns spanning a comprehensive 20-year period. With a laser focus on the unprecedented events of 2020, notably the global COVID-19 pandemic, the project endeavors to unravel the intricate patterns and trends within the stock market. Leveraging an array of powerful Python libraries including Seaborn, Pandas, yfinance, and Matplotlib, the project kicks off with the crucial task of importing and structuring the dataset. Following this initial step, the project employs sophisticated visualization techniques such as heat maps to provide a comprehensive overview of monthly returns, offering insights into the volatility and fluctuations inherent in the market. Additionally, histograms are crafted to dissect the distribution of returns across different standard deviation buckets, providing a nuanced understanding of the market's risk profile over time. Through an exhaustive exploratory data analysis, the project aims to extract actionable insights and meaningful observations, shedding light on the underlying dynamics of the stock market.

Expanding its scope, the project integrates the SARIMA forecasting model into a user-friendly web application using Flask. SARIMA's ability to capture seasonal patterns makes it ideal for stock market predictions. The application empowers users to interactively forecast future market values, providing a seamless interface for data exploration and decision-making. By bridging analytical techniques with user-centric design, the project aims to democratize predictive insights, aiding investors in navigating the complexities of the stock market confidently.

TABLE OF CONTENTS

1. INTRODUCTION

The NIFTY 50 serves as a key benchmark for the Indian stock market, representing the combined performance of 50 of the largest companies listed on the National Stock Exchange (NSE). Managed by NSE Indices, a subsidiary of NSE Strategic Investment Corporation Limited, the index was launched in April 1996 and has since become a cornerstone of India's financial landscape. It offers exposure to various sectors of the Indian economy, allowing investment managers to access the market through a single portfolio.

Over the years, the NIFTY 50 has evolved into the largest financial product in India, fostering a robust ecosystem that includes exchange-traded funds, futures, and options, both domestically and internationally. Despite its prominence, the index experienced a decline in market share from 65% to 29% between 2008 and 2012, partly due to the emergence of sector-specific indices such as NIFTY Bank, NIFTY IT, NIFTY Pharma, and NIFTY Next 50.

Comprising 13 sectors, the NIFTY 50 provides a diversified representation of the Indian economy. As of March 2024, the index allocates significant weightage to financial services (33.53%), information technology (13.04%), oil and gas (12.87%), consumer goods (8.15%), and automotive (7.57%). This broad sectoral coverage underscores the index's role as a comprehensive barometer of the Indian stock market, facilitating investment decisions and portfolio diversification for market participants.

NIFTY is a popular stock market index in India that investors, traders, analysts, and fund managers widely use to track the overall performance of the Indian stock market. The index comprises the top 50 companies listed on the National Stock Exchange (NSE) based on their market capitalization. It is reviewed and rebalanced periodically to reflect the current market conditions.

One of the main uses of NIFTY is as a benchmark for measuring the performance of the Indian stock market. By tracking the movement of the index, investors can get a sense of the overall health of the market and its direction. They can also use the index to compare the performance of individual stocks or their portfolios against the broader market.

NIFTY is also used as a tool for investment. Investors can use the index to gain exposure to the Indian stock market by investing in index funds or exchange-traded funds (ETFs) that track the NIFTY index's performance. This provides a convenient way for investors to diversify their portfolios across various companies and sectors.

Another important use of NIFTY is as a barometer of the Indian economy's health. As the index comprises the top 50 companies in India, it reflects the country's economic growth and development over time. Therefore, changes in the NIFTY index can be seen as indicators of the overall direction of the Indian economy.

Simply put, NIFTY is crucial in the Indian stock market ecosystem. It is an essential tool for market participants, providing a comprehensive snapshot of the Indian stock market's performance and facilitating investment decisions. By tracking the movement of the index, investors can gain insights into market trends, risks, and opportunities and make informed investment decisions.

2.PROBLEM STATEMENT

This project dives deep into analyzing stock market data, with a specific focus on the Nifty index. The dataset spans an extensive 20-year period and includes detailed monthly returns of Nifty, offering a wealth of information for analysis. By examining this data, we aim to identify trends and patterns that can guide investment decisions and enhance portfolio management strategies. Through thorough analysis and interpretation, we seek to provide valuable insights that empower investors to make informed choices in navigating the complexities of the stock market.

The year 2020 holds particular significance due to the unprecedented challenges posed by the COVID-19 pandemic. Amidst this backdrop, we are keen to explore how the stock market, especially the Nifty index, responded to the turmoil and volatility induced by the pandemic. By dissecting the data from this pivotal year, we hope to uncover valuable insights into market dynamics and investor behavior during times of uncertainty. Our ultimate goal is to equip investors with the knowledge and understanding needed to navigate turbulent market conditions effectively and make sound investment decisions that align with their financial goals

3.Objectives

- Begin by importing the dataset into Python to enable further analysis and exploration. This initial step is crucial for laying the groundwork and accessing the Nifty data for comprehensive examination.

- Construct a heat map to visually represent Nifty's monthly returns across the specified timeframe. This visualization provides valuable insights into the temporal fluctuations and trends within Nifty's performance, aiding in understanding its behavior over different time periods.

- Categorize Nifty's monthly returns into distinct buckets based on standard deviations and generate a histogram to showcase the distribution of returns. This segmentation strategy offers a nuanced understanding of return variability and highlights the prevalence of outlier events within Nifty's performance.

- Similar to the monthly analysis, create a histogram illustrating the distribution of yearly returns by categorizing them into standard deviation-based buckets. This macroscopic view of Nifty's performance on an annual basis helps identify overarching trends and anomalies.

- Extend the project beyond mere analysis by developing a WEBUI or application using the Flask micro web framework. This interactive platform empowers stakeholders to engage directly with the analyzed data, fostering a deeper understanding of Nifty's performance and facilitating informed investment decisions.

4. METHODOLOGY

➢ 4.1 Data Source

▪ Brief description of the data source :

Data Overview
The dataset provides a comprehensive overview of stock price data spanning from the years 2000 to 2023. Each row corresponds to a specific year, with columns representing individual months from January to December. Additionally, there is a column labeled "Annual" which signifies the total stock price for the entire year.

Feature Description

. Year: This column denotes the years covered in the dataset, ranging from 2000 to 2023.
. Jan-Dec: These columns represent the stock prices for each respective month of the year. For example, the "Jan" column contains stock prices for January, "Feb" for February, and so on.
. Annual: The "Annual" column provides the total stock price for the entire year, aggregating the monthly data. This column offers a holistic view of the yearly performance of the stock.

➢ 4.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) serves as a crucial initial phase in the machine learning workflow, aiming to comprehensively examine and comprehend the data's properties before engaging in modeling endeavors. Through EDA, analysts seek to elucidate fundamental data characteristics, including distribution, inter-variable correlations, and discernible patterns or irregularities. This preliminary exploration is pivotal as it furnishes a profound insight into the dataset, facilitating informed decisions regarding feature manipulation, data preprocessing, and model selection.

By undertaking EDA, researchers can pinpoint and rectify any missing or erroneous data, outliers, or inconsistencies, thus ensuring a more robust foundation for subsequent machine learning tasks.

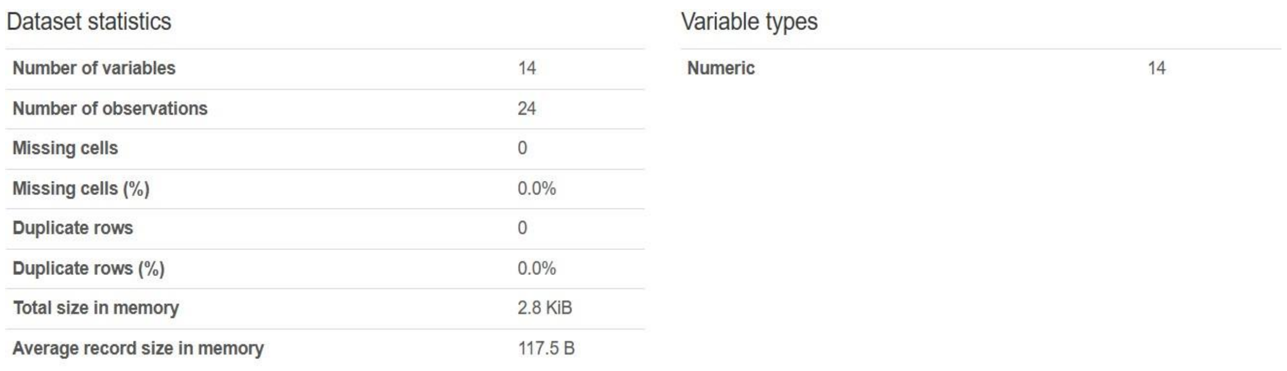
▪ Information about the Features & their data types:

Column No.1: Year, Data Type: int64
Column No.2: Jan, Data Type: float64
Column No.3: Feb, Data Type: float64
Column No.4: Mar, Data Type: float64
Column No.5: Apr, Data Type: float64
Column No.6: May, Data Type: float64
Column No.7: Jun, Data Type: float64

Column No.8: Jul, Data Type: float64
Column No.9: Aug, Data Type: float64
Column No.10: Sep, Data Type: float64
Column No.11: Oct, Data Type: float64
Column No.12: Nov, Data Type: float64
Column No.13: Dec, Data Type: float64
Column No.14: Annual, Data Type: float64

The dataset consists of 14 columns. Column No.1 represents the year and is of integer data type. Columns No.2 to No.13 represent the months from January to December, each of float64 data type. Column No.14 represents the annual data and is also of float64 data type.

➢ 4.2.2 Checking for Data Consistency



Dataset statistics

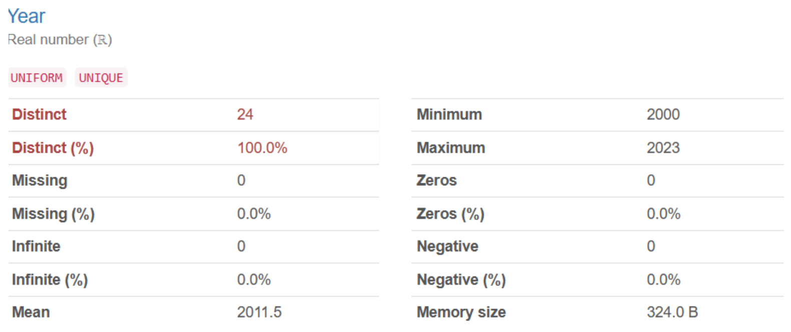
Number of variables	14
Number of observations	24
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	2.8 KiB
Average record size in memory	117.5 B

Variable types

Numeric	14

The dataset contains 14 variables, all of which are numeric. It comprises 24 observations with no missing ce lls or duplicate rows. In memory, the dataset occupies 2.8 KiB, with an average record size of 117.5 bytes.

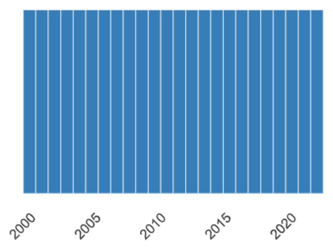
Year



Year
Real number (ℝ)

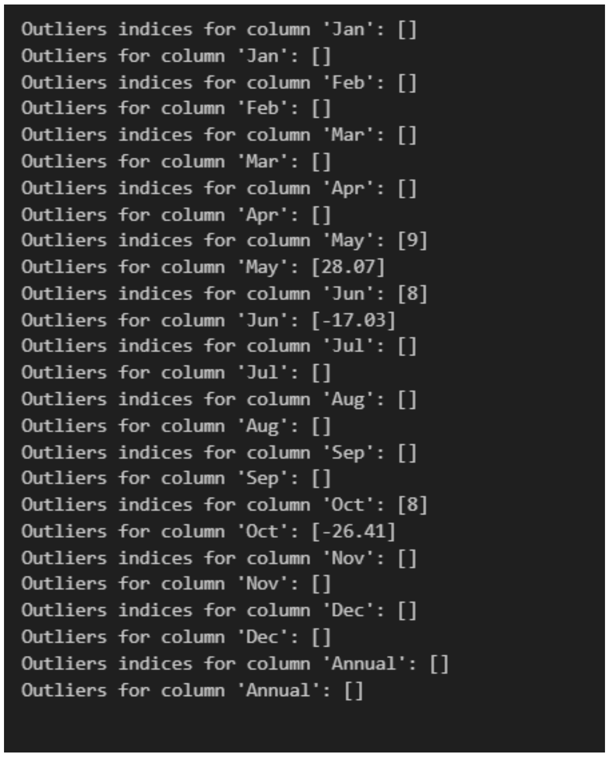
UNIFORM UNIQUE

Distinct	24	Minimum	2000
Distinct (%)	100.0%	Maximum	2023
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	2011.5	Memory size	324.0 B



The "Year" variable consists of real numbers (ℝ) ranging uniformly from 2000 to 2023, with 24 distinct valu es covering the entire range. There are no missing, infinite, zero, or negative values present. The mean valu e is 2011.5, and the memory size occupied by this variable is 324.0 bytes.

Outliers:



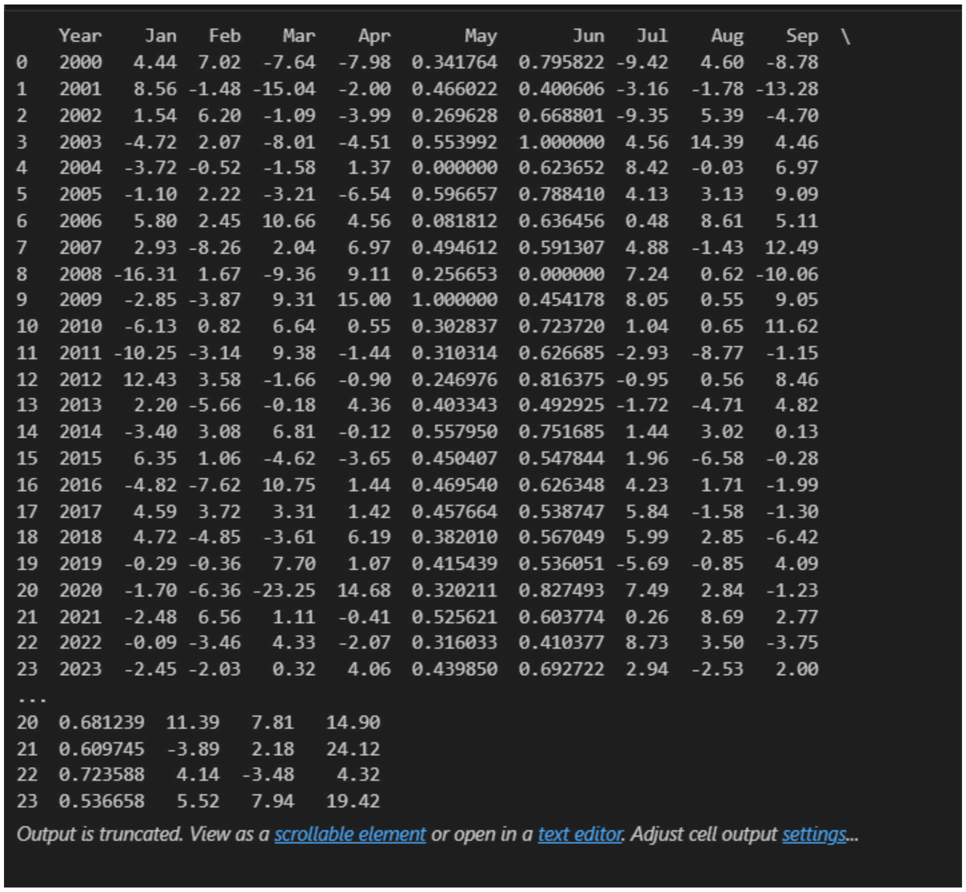
```
Outliers indices for column 'Jan': []
Outliers for column 'Jan': []
Outliers indices for column 'Feb': []
Outliers for column 'Feb': []
Outliers indices for column 'Mar': []
Outliers for column 'Mar': []
Outliers indices for column 'Apr': []
Outliers for column 'Apr': []
Outliers indices for column 'May': [9]
Outliers for column 'May': [28.07]
Outliers indices for column 'Jun': [8]
Outliers for column 'Jun': [-17.03]
Outliers indices for column 'Jul': []
Outliers for column 'Jul': []
Outliers indices for column 'Aug': []
Outliers for column 'Aug': []
Outliers indices for column 'Sep': []
Outliers for column 'Sep': []
Outliers indices for column 'Oct': [8]
Outliers for column 'Oct': [-26.41]
Outliers indices for column 'Nov': []
Outliers for column 'Nov': []
Outliers indices for column 'Dec': []
Outliers for column 'Dec': []
Outliers indices for column 'Annual': []
Outliers for column 'Annual': []
```

Observations:

- Above figure indicates the indices and corresponding values of outliers detected for each column in the dataset.
- For the 'Jan', 'Feb', 'Mar', 'Apr', 'Jul', 'Aug', 'Sep', 'Nov', 'Dec', and 'Annual' columns:
  No outliers were detected.
- For the 'May' column:
  There is one outlier detected at index 9 with the value 28.07.
- For the 'Jun' column:
  There is one outlier detected at index 8 with the value -17.03.
- For the 'Oct' column:
  There is one outlier detected at index 8 with the value -26.41.

These observations suggest that the majority of the columns have no outliers according to the specified threshold. However, a few columns ('May', 'Jun', and 'Oct') contain outliers at specific indices. These outliers might represent data points that significantly deviate from the mean and could potentially be errors or represent unusual phenomena.

Normalization:



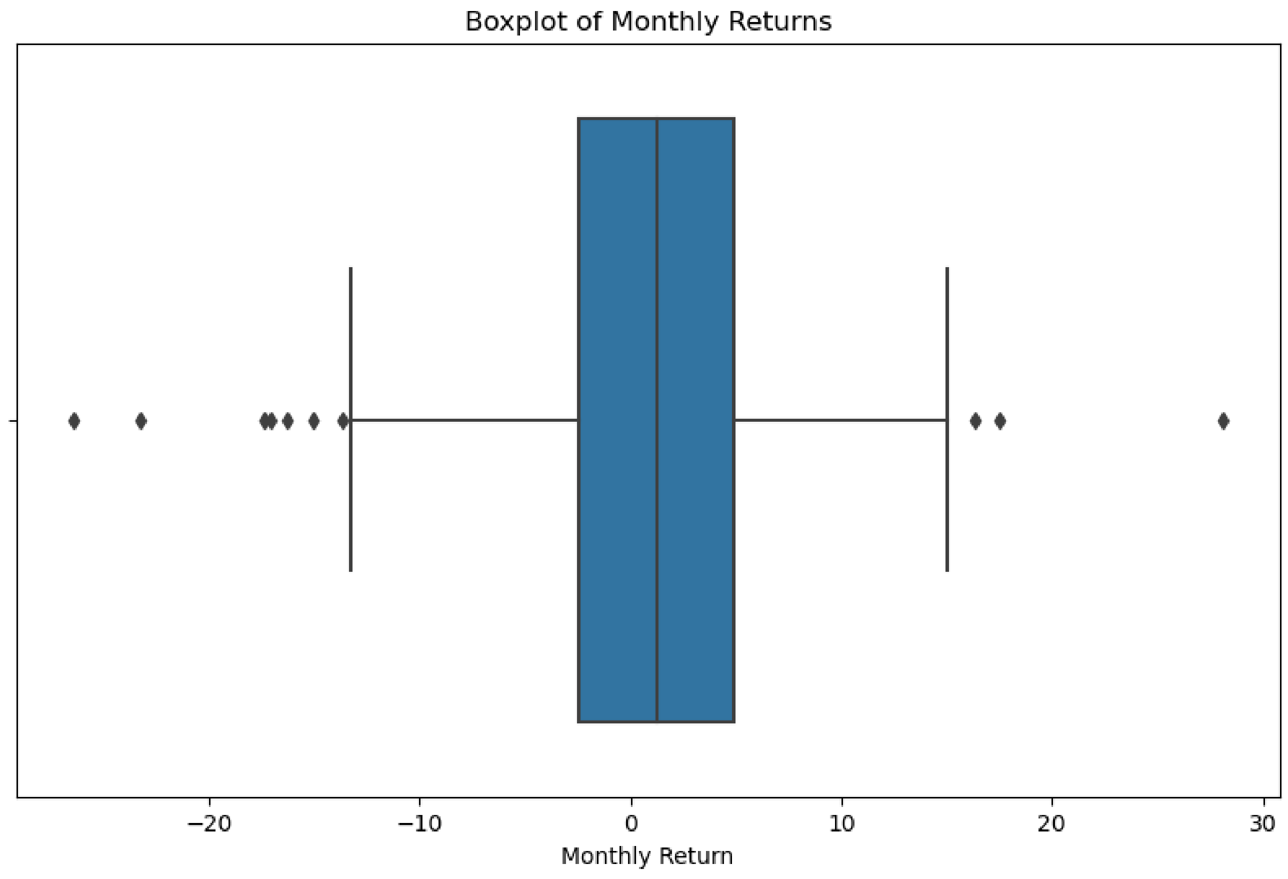
```
    Year    Jan   Feb    Mar    Apr       May       Jun   Jul    Aug    Sep  \
0   2000   4.44  7.02  -7.64  -7.98  0.341764  0.795822 -9.42   4.60  -8.78
1   2001   8.56 -1.48 -15.04  -2.00  0.466022  0.400606 -3.16  -1.78 -13.28
2   2002   1.54  6.20  -1.09  -3.99  0.269628  0.668801 -9.35   5.39  -4.70
3   2003  -4.72  2.07  -8.01  -4.51  0.553992  1.000000  4.56  14.39   4.46
4   2004  -3.72 -0.52  -1.58   1.37  0.000000  0.623652  8.42  -0.03   6.97
5   2005  -1.10  2.22  -3.21  -6.54  0.596657  0.788410  4.13   3.13   9.09
6   2006   5.80  2.45  10.66   4.56  0.081812  0.636456  0.48   8.61   5.11
7   2007   2.93 -8.26   2.04   6.97  0.494612  0.591307  4.88  -1.43  12.49
8   2008 -16.31  1.67  -9.36   9.11  0.256653  0.000000  7.24   0.62 -10.06
9   2009  -2.85 -3.87   9.31  15.00  1.000000  0.454178  8.05   0.55   9.05
10  2010  -6.13  0.82   6.64   0.55  0.302837  0.723720  1.04   0.65  11.62
11  2011 -10.25 -3.14   9.38  -1.44  0.310314  0.626685 -2.93  -8.77  -1.15
12  2012  12.43  3.58  -1.66  -0.90  0.246976  0.816375 -0.95   0.56   8.46
13  2013   2.20 -5.66  -0.18   4.36  0.403343  0.492925 -1.72  -4.71   4.82
14  2014  -3.40  3.08   6.81  -0.12  0.557950  0.751685  1.44   3.02   0.13
15  2015   6.35  1.06  -4.62  -3.65  0.450407  0.547844  1.96  -6.58  -0.28
16  2016  -4.82 -7.62  10.75   1.44  0.469540  0.626348  4.23   1.71  -1.99
17  2017   4.59  3.72   3.31   1.42  0.457664  0.538747  5.84  -1.58  -1.30
18  2018   4.72 -4.85  -3.61   6.19  0.382010  0.567049  5.99   2.85  -6.42
19  2019  -0.29 -0.36   7.70   1.07  0.415439  0.536051 -5.69  -0.85   4.09
20  2020  -1.70 -6.36 -23.25  14.68  0.320211  0.827493  7.49   2.84  -1.23
21  2021  -2.48  6.56   1.11  -0.41  0.525621  0.603774  0.26   8.69   2.77
22  2022  -0.09 -3.46   4.33  -2.07  0.316033  0.410377  8.73   3.50  -3.75
23  2023  -2.45 -2.03   0.32   4.06  0.439850  0.692722  2.94  -2.53   2.00
...
20  0.681239  11.39   7.81   14.90
21  0.609745  -3.89   2.18   24.12
22  0.723588   4.14  -3.48    4.32
23  0.536658   5.52   7.94   19.42
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Observations:

- The Min-Max scaling method has been applied to the columns 'May', 'Jun', and 'Oct' of the DataFrame.
- After scaling, the values in these columns have been transformed to a range between 0 and 1, ensuring that each feature contributes equally to the analysis and preventing features with larger scales from dominating the model's learning process.
- For example, the values in the 'May' column originally ranged from approximately -0.03 to 28.07. After scaling, these values are transformed to a range between 0.00 and 1.00.
- Similarly, the 'Jun' column values originally ranged from approximately -17.03 to 12.65, which have been transformed to a range between 0.00 and 1.00 after scaling.
- This normalization technique ensures that the features are on a similar scale, which can improve the performance of machine learning algorithms that are sensitive to the scale of the input features, such as gradient descent-based algorithms.
- The transformed DataFrame now contains scaled values for the specified columns, which can be used for further analysis or modeling.

5. Data Visualization

Boxplot visualization

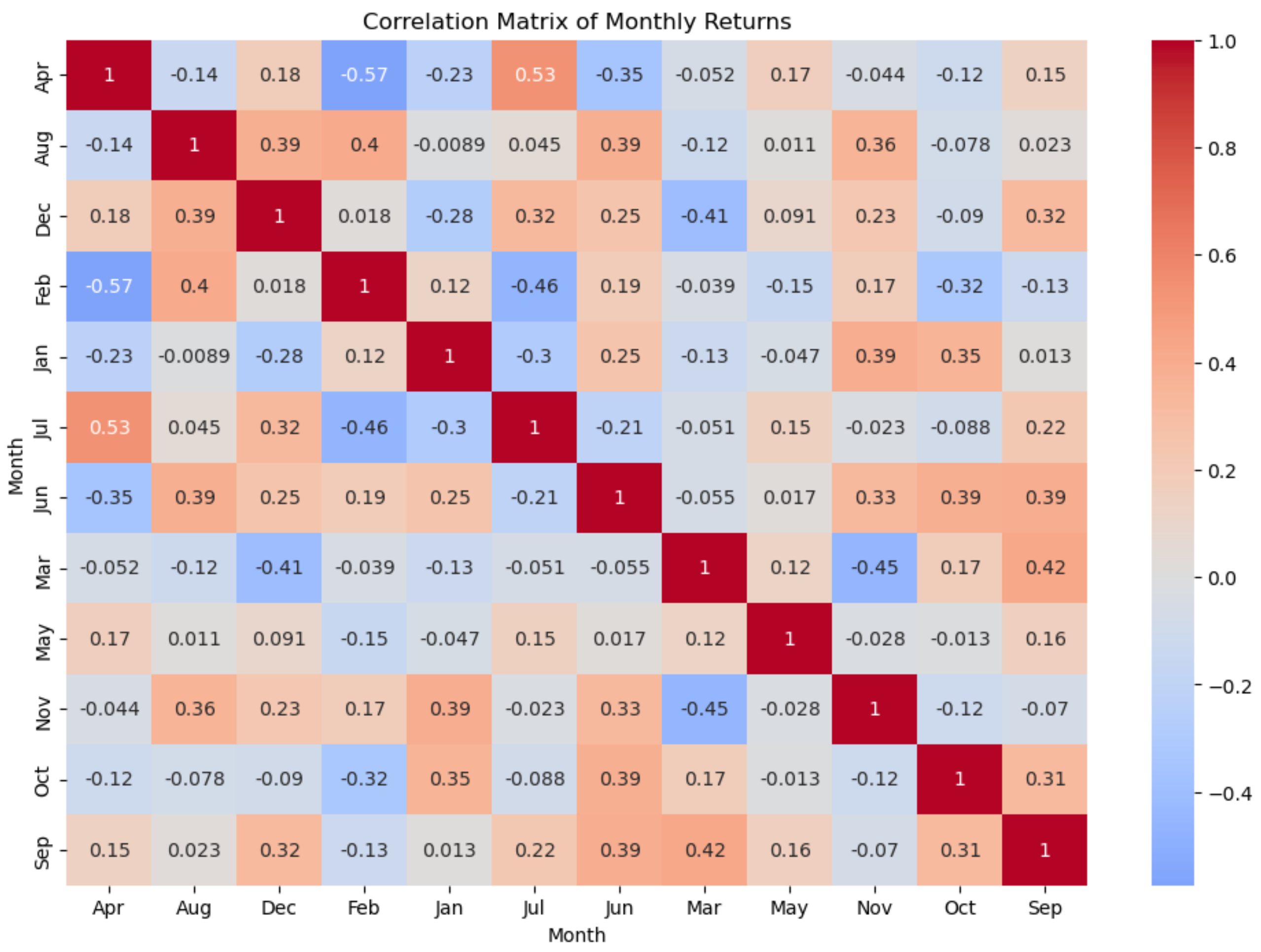


1. A **boxplot** (also known as a **box and whisker plot**) is a type of chart used in **explanatory data analysis**. It visually represents the distribution of numerical data and skewness by displaying key summary statistics. Here's how to interpret a boxplot:
    - **Minimum Score**: The lowest score (excluding outliers) is shown at the end of the left whisker.
    - **Lower Quartile (Q1)**: Twenty-five percent of scores fall below this value.
    - **Median (Q2)**: The mid-point of the data, dividing the box into two parts. Half the scores are greater than or equal to this value, and half are less.
    - **Upper Quartile (Q3)**: Seventy-five percent of scores fall below this value.
    - **Maximum Score**: The highest score (excluding outliers) is shown at the end of the right whisker.
    - **Whiskers**: Represent scores outside the middle 50% (i.e., the lower 25% and upper 25% of scores).
    - **Interquartile Range (IQR)**: The range between Q1 and Q3, showing the middle 50% of scores.
2. **Interpretation**:
    - The data for the entire year (annual summary) is significantly higher than the monthly data. This indicates a **large increase or accumulation** over time.

- The absence of boxes for individual months suggests that their data points may be more concentrated or less variable compared to the annual data.

3. The x-axis represents **months of the year** from **January** to **December**, with additional labels for **“Year”** and **“Annual”**.
4. The y-axis displays **numerical values** ranging from **0 to 2000** (marked at intervals of 500).
5. Notable observations:
    - A prominent marker appears at the **“Year”** label on the x-axis, reaching up to **2000** on the y-axis.
    - There are black markers at each month, but their significance remains unclear without additional context.

Histogram visualization



Histogram of Monthly Returns:

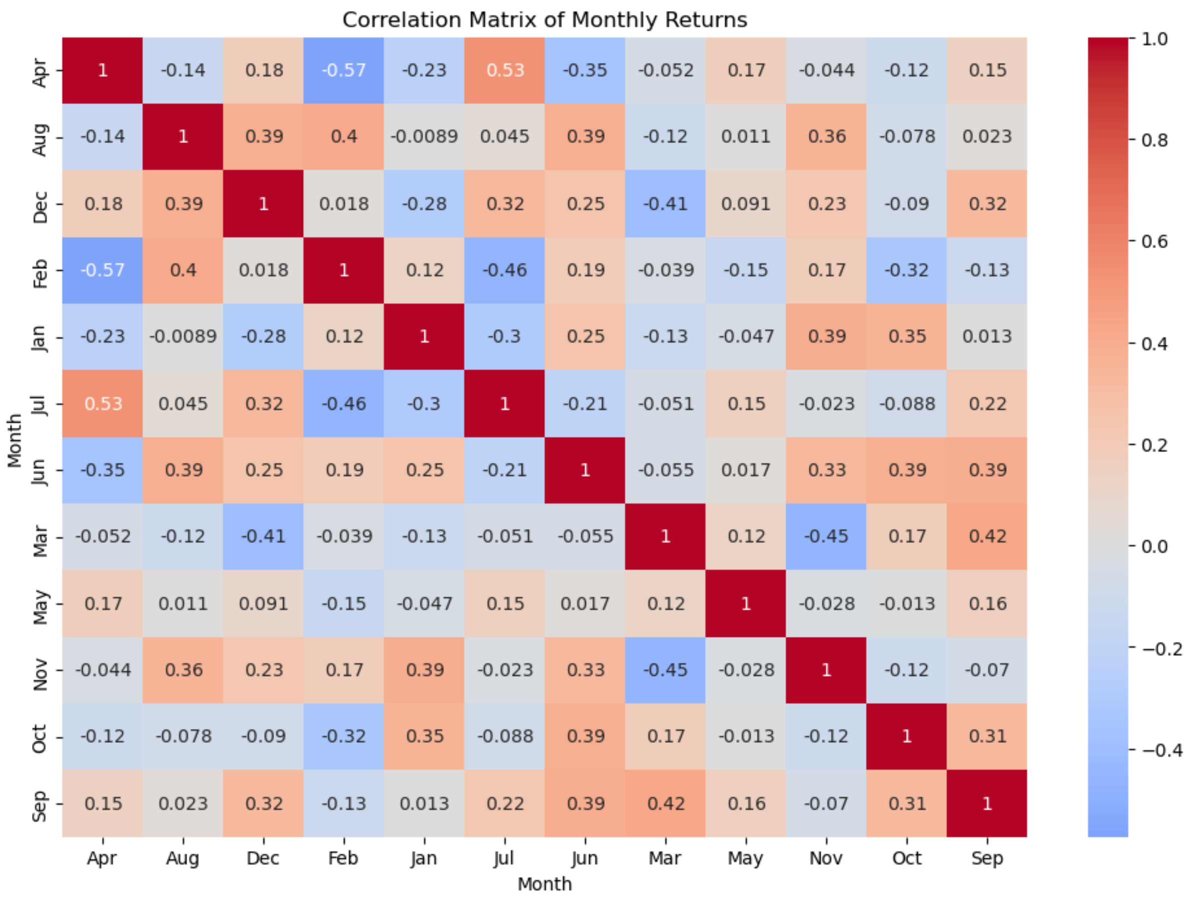
The distribution of monthly returns is bell-shaped and appears to approximate a normal distribution, with a concentration of values around the mean. There are tails on both ends of the distribution, indicating that there are months with significantly higher and lower returns, although these are less frequent. Boxplot of Monthly Returns:

The boxplot shows a median near zero, with a fairly even spread of returns above and below the median. There are outliers present on both the high and low ends, indicating months with exceptionally high or low returns. These could correspond to periods of market stress or exuberance. Bar Chart of Average Monthly Returns:

This bar chart suggests seasonality in the returns, with certain months (like December) showing higher average returns. This could be due to year-end effects such as the "Santa Claus rally." Some months (like January, February, and March) exhibit lower average returns, which could invite further investigation into phenomena like the "January effect." Time Series of Monthly Returns:

The time series chart shows volatility in returns, with no clear long-term upward or downward trend, which is consistent with the random walk hypothesis of stock prices. Notable sharp drops or spikes could be associated with specific market events, financial crises, or economic news releases. These points in time may warrant a closer look to understand the factors affecting such significant moves.

5.1 Correlation matrix



Month	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	Oct	Sep
Apr	1	-0.14	0.18	-0.57	-0.23	0.53	-0.35	-0.052	0.17	-0.044	-0.12	0.15
Aug	-0.14	1	0.39	0.4	-0.0089	0.045	0.39	-0.12	0.011	0.36	-0.078	0.023
Dec	0.18	0.39	1	0.018	-0.28	0.32	0.25	-0.41	0.091	0.23	-0.09	0.32
Feb	-0.57	0.4	0.018	1	0.12	-0.46	0.19	-0.039	-0.15	0.17	-0.32	-0.13
Jan	-0.23	-0.0089	-0.28	0.12	1	-0.3	0.25	-0.13	-0.047	0.39	0.35	0.013
Jul	0.53	0.045	0.32	-0.46	-0.3	1	-0.21	-0.051	0.15	-0.023	-0.088	0.22
Jun	-0.35	0.39	0.25	0.19	0.25	-0.21	1	-0.055	0.017	0.33	0.39	0.39
Mar	-0.052	-0.12	-0.41	-0.039	-0.13	-0.051	-0.055	1	0.12	-0.45	0.17	0.42
May	0.17	0.011	0.091	-0.15	-0.047	0.15	0.017	0.12	1	-0.028	-0.013	0.16
Nov	-0.044	0.36	0.23	0.17	0.39	-0.023	0.33	-0.45	-0.028	1	-0.12	-0.07
Oct	-0.12	-0.078	-0.09	-0.32	0.35	-0.088	0.39	0.17	-0.013	-0.12	1	0.31
Sep	0.15	0.023	0.32	-0.13	0.013	0.22	0.39	0.42	0.16	-0.07	0.31	1

From the heatmap provided, here are a few observations we can infer:

High Volatility: There are years with high volatility where the monthly returns vary significantly, as indicated by the presence of both warm and cool colors in the same row. For instance, the year 2009 shows months with both high positive and negative returns, suggesting a turbulent market.

Periods of Growth and Decline: There are clear periods of growth and decline. For example, 2003 shows predominantly warm colors, indicating a generally positive year for returns. Conversely, 2008 shows a significant period of cooler colors, particularly towards the end of the year, which corresponds with the global financial crisis.

Extreme Values: The presence of extreme positive or negative values in certain months, such as in May and June 2004, where there's a significant negative value, or in March 2020 with a large negative value likely related to the onset of the COVID-19 pandemic.

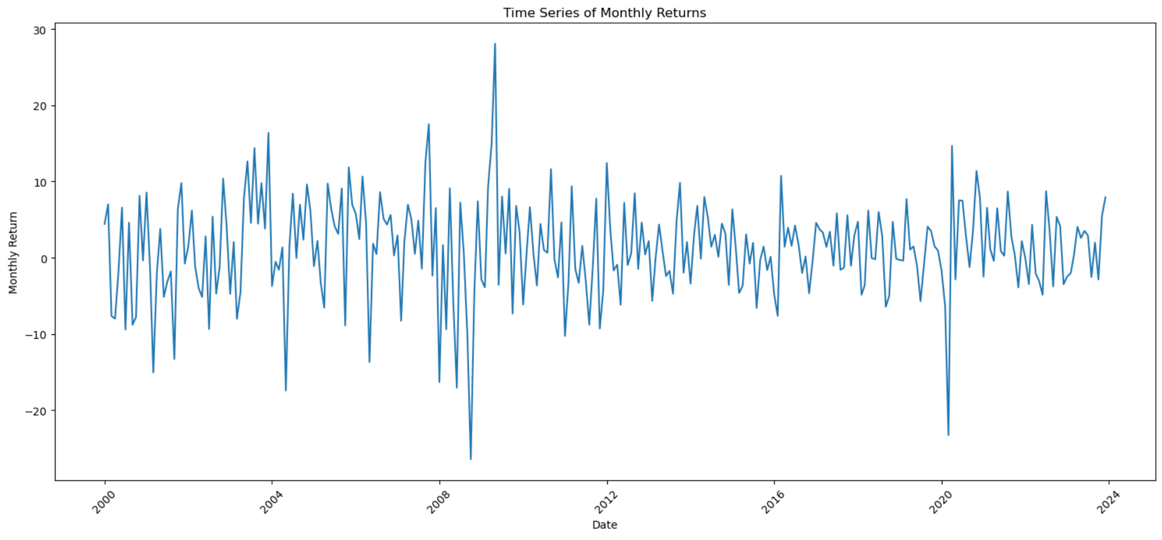
Overall Trend: While there are years of decline, the general trend over the two decades appears to have more periods of positive returns than negative ones, as there are more warm-toned squares than cool-toned.

Recent Performance: The most recent years, such as 2020 and 2021, show a mix of positive and negative returns, with some extreme values. This suggests that while there has been recovery from the initial shock of the COVID-19 pandemic, the market still faces instability and uncertainty.

Annual Perspective: It's also interesting to note the annual columns on the far right, which provide a quick visual summary of the overall yearly performance. This tells us that despite monthly fluctuations, the annual returns may still end up positive, as seen in several years.

Historical Context: Specific historical events can be correlated with the patterns observed. For example, market crashes, economic booms, or global crises are often reflected in the color patterns and can be cross-referenced with historical data for more detailed insights.

5.2 Time Series Monthly Returns



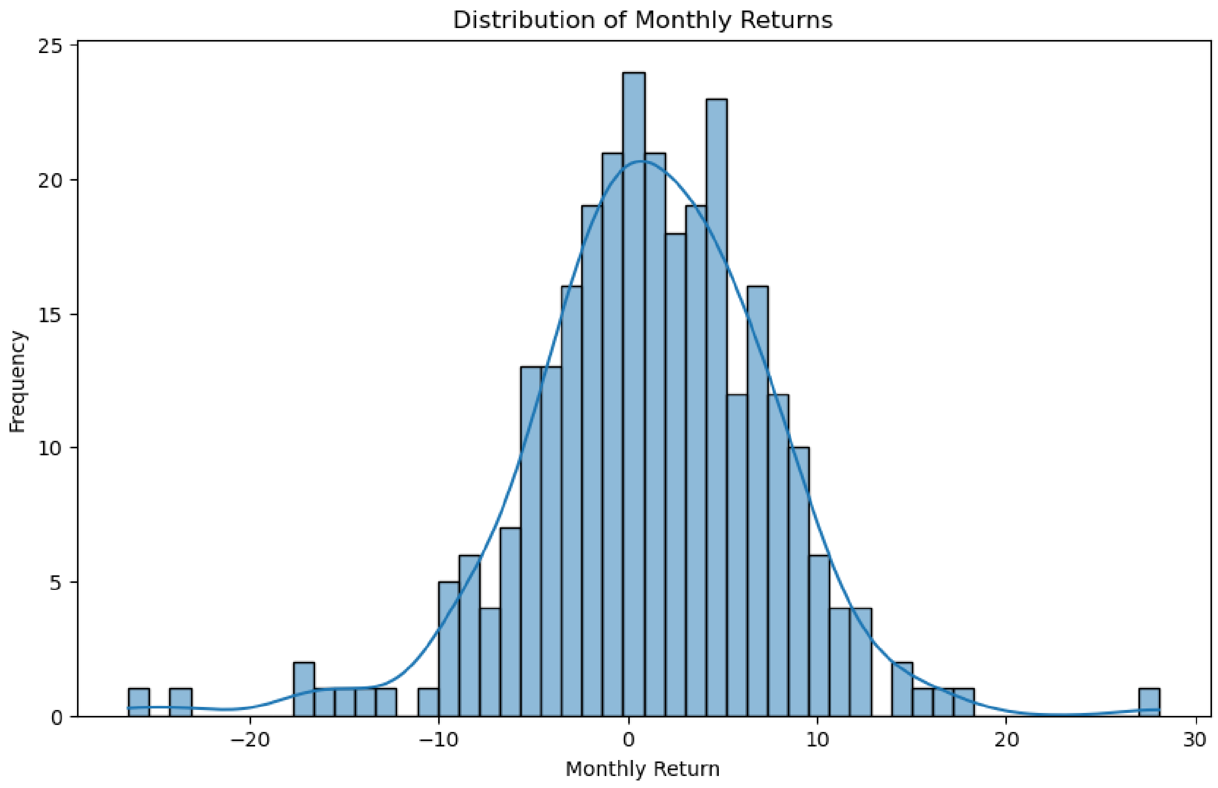
Trend Over Time: By observing the overall direction of the line plot, we can identify whether there is a consistent trend in monthly returns over the specified period. An upward slope suggests increasing returns over time, while a downward slope indicates decreasing returns.

Seasonality: Look for recurring patterns or cycles in the plot, which may indicate seasonality in monthly returns. Seasonal fluctuations could be related to specific time periods, such as months of the year or quarters.

Volatility: Identify periods of high or low volatility in monthly returns based on the fluctuations in the line plot. Sharp peaks and valleys indicate periods of high volatility, while smoother segments suggest more stable returns.

Outliers or Anomalies: Any significant spikes or dips in the plot may indicate outliers or anomalies in monthly returns, which could be caused by extraordinary events or market conditions.

5.3 Distribution of Monthly Returns



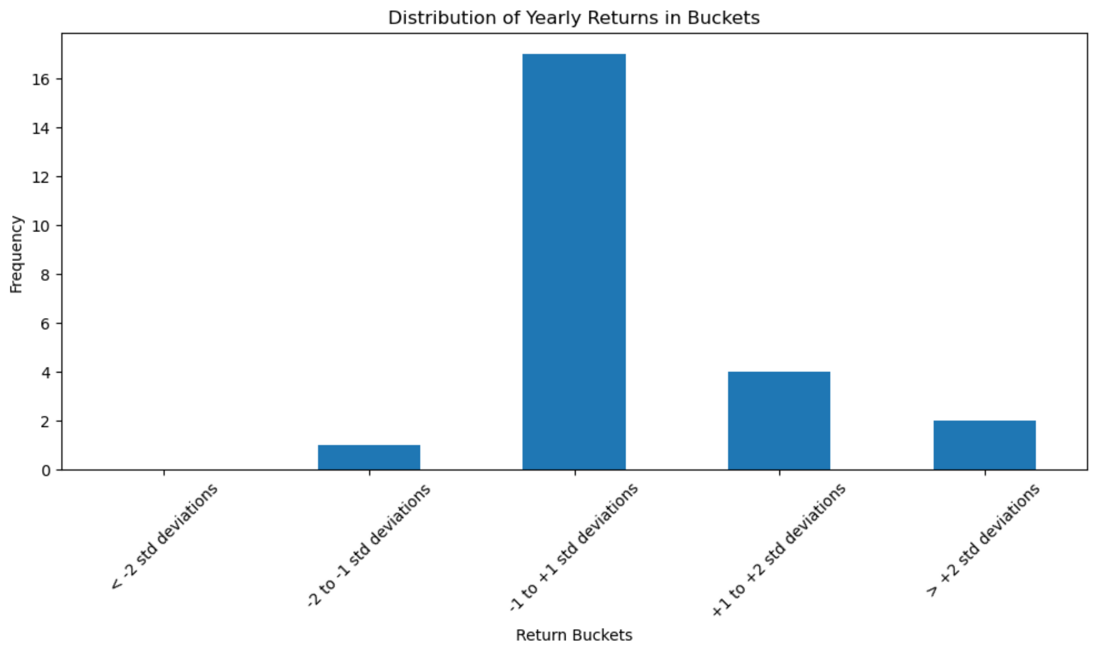
Negative Returns: The histogram displays negative returns on the left side of the plot, with bars extending below the zero line. The tallest bar in the negative returns range indicates the most frequent range of negative monthly returns observed in the dataset.

Positive Returns: Positive returns are displayed on the right side of the plot, with bars extending above the zero line. The tallest bar in the positive returns range represents the most frequent range of positive monthly returns observed in the dataset.

Magnitude of Returns: The height of the bars indicates the frequency of returns within each range. Taller bars represent higher frequencies of returns falling within the corresponding range.

Skewness: If one side of the histogram (positive or negative returns) has significantly taller bars compared to the other side, it suggests skewness in the distribution of monthly returns towards that side..

5.4 Distribution of Yearly Returns in Buckets

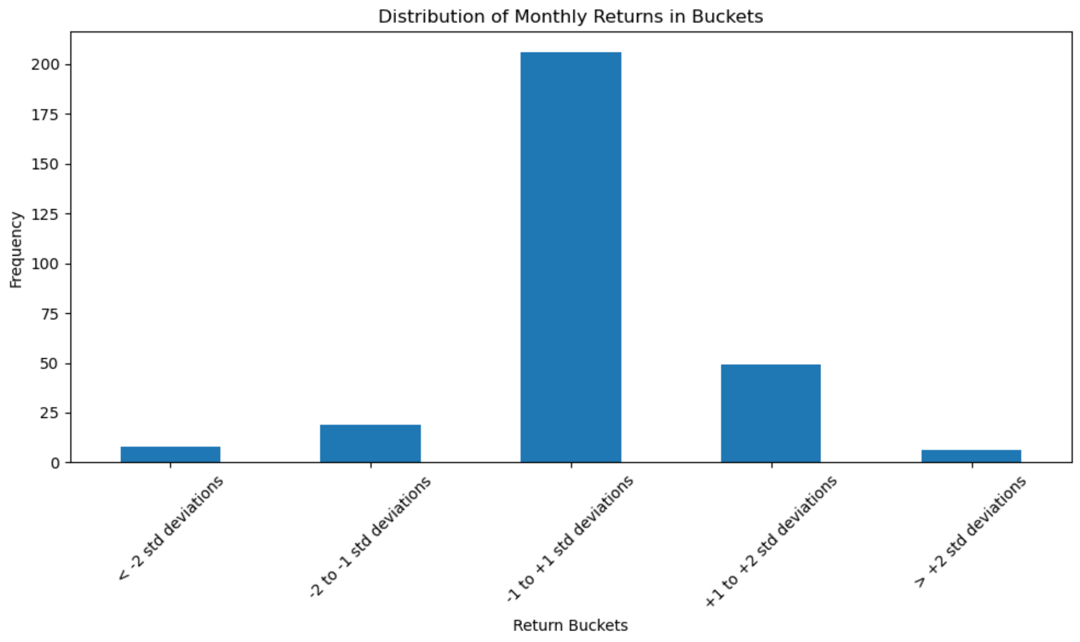


• X-Axis: The x-axis represents the return buckets, with each bucket labeled accordingly. These labels likely correspond to different ranges or categories of yearly returns, such as "Low Returns," "Moderate Returns," "High Returns," etc.

- Y-Axis: The y-axis represents the frequency or count of yearly returns. It indicates how many data points fall into each return bucket category.

- Bar Heights: The height of each bar corresponds to the frequency of yearly returns within the respective bucket. Higher bars indicate a higher frequency of returns falling into that bucket, while shorter bars indicate a lower frequency.

5.5 Distribution of Monthly Returns in Buckets



X-Axis: Represents the return buckets or categories. Each bucket is labeled accordingly, likely indicating different ranges or classifications of monthly returns.

Y-Axis: Displays the frequency or count of monthly returns. It shows how many data points fall into each return bucket category.

This bar plot offers a visual summary of how monthly returns are distributed across different buckets or categories, allowing viewers to quickly grasp the frequency distribution and identify any patterns or trends in the data.

6.ALGORITHMS

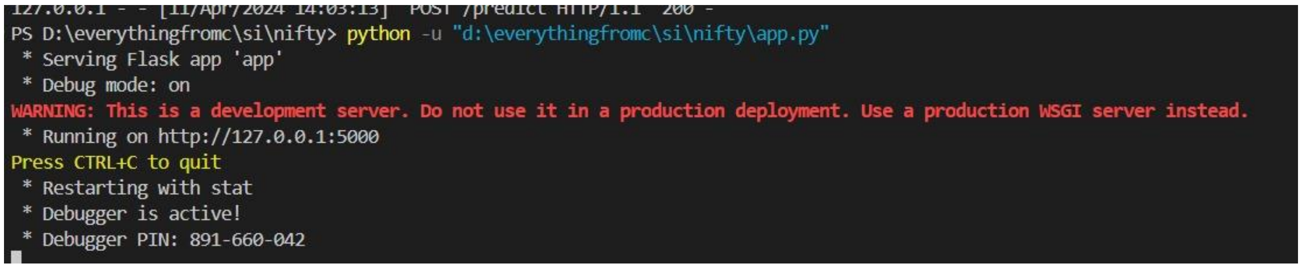
1. SARIMAX MODEL

Among the various approaches available, the SARIMAX (Seasonal Autoregressive Integrated Moving Average + exogenous variables) model stands out as a powerful tool for modeling and forecasting both trends and seasonal variations in temporal data, while incorporating exogenous variables into the analysis to improve prediction accuracy.

The SARIMAX model represents a significant advance in time-series analysis by allowing the integration of covariates. By incorporating external variables to enrich the analysis, this model enables us to better understand future trends and predictions. However, as with any methodology, it is crucial to master the model parameters and understand the results to obtain relevant and reliable predictions.

7.Implementation

Step 1: run the python code app.py



```
127.0.0.1 - - [11/Apr/2024 14:03:13] "POST /predict HTTP/1.1" 200 -
PS D:\everythingfromc\si\nifty> python -u "d:\everythingfromc\si\nifty\app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 891-660-042
```

Create Flask App:

- Import the Flask class and instantiate it to create a Flask app instance.
- Define routes using the @app.route('/myurl') decorator to specify URL endpoints for different functionalities.

Create Index.html File:

- Create an HTML file named index.html to serve as the user interface for inputting data.
- Design the HTML page to contain input fields for 2 attributes required for the Annual return prediction.

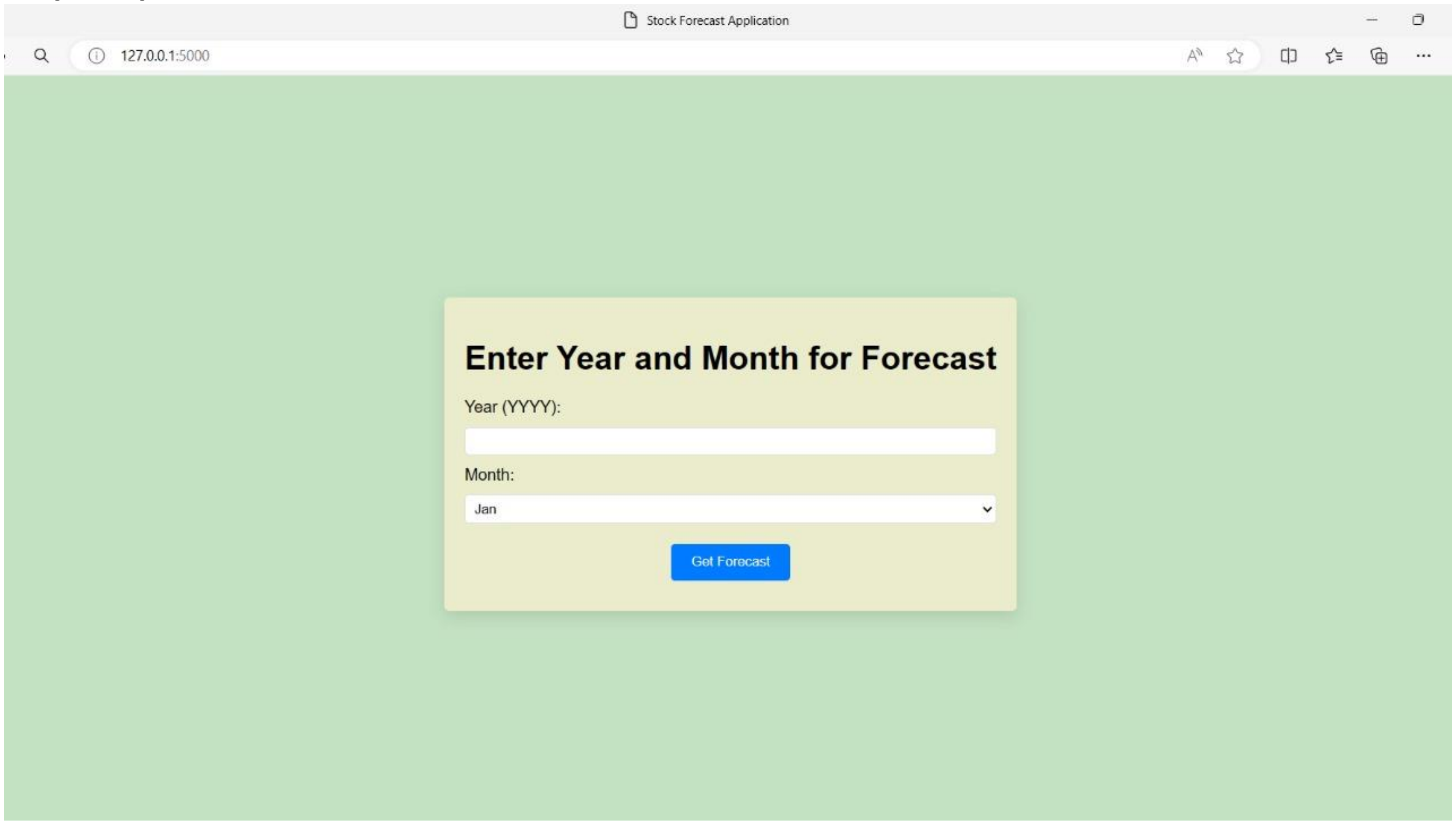
Implement Flask Routes:

- Define Flask routes to handle requests from the client-side.
- For example, create a route to render the index.html template and another route to handle form submissions.

Run Flask App:

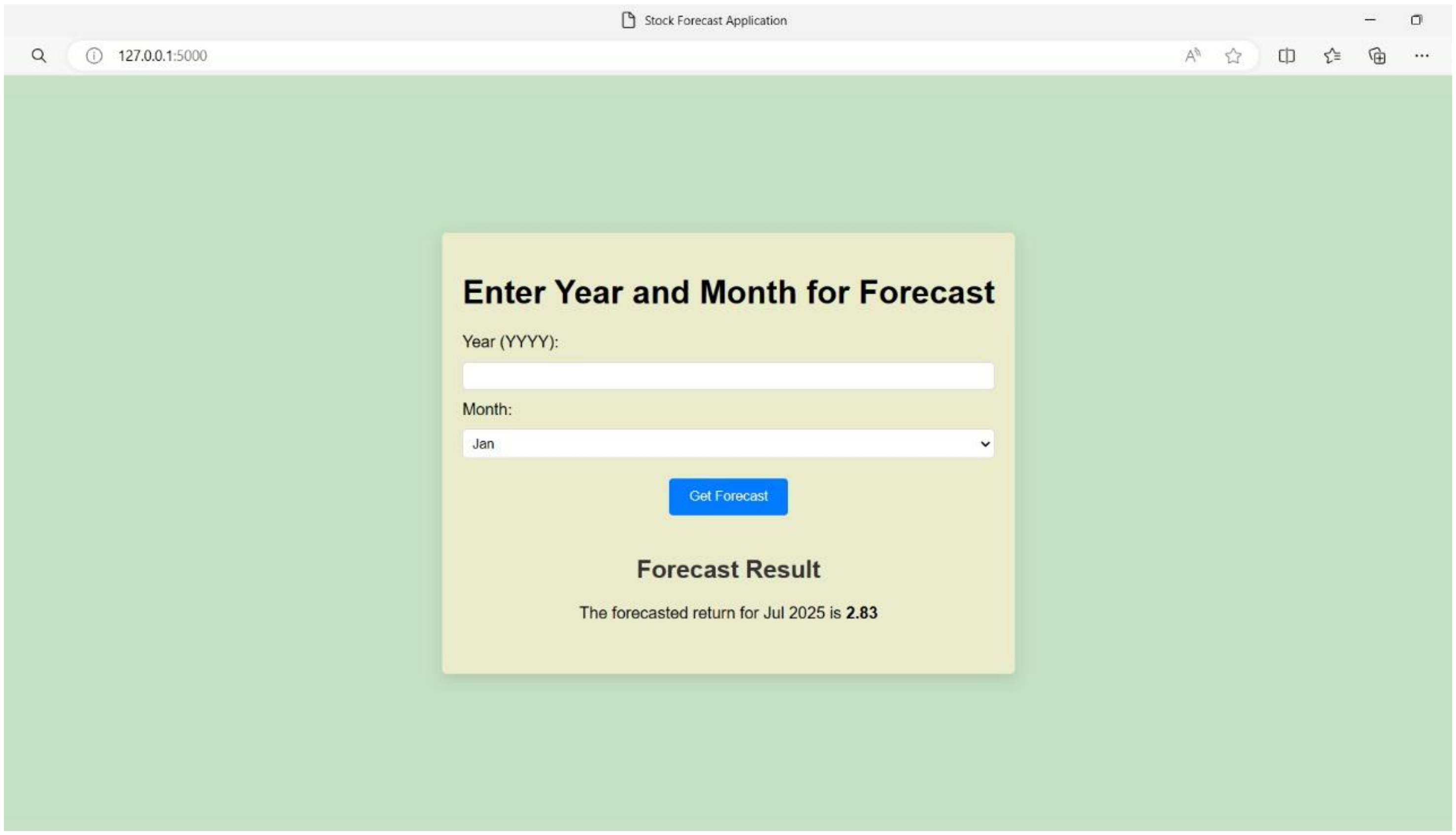
- Run the Flask app using the flask run command or by executing the Python script containing the Flask app.
- Access the Flask app through a web browser to interact with the user interface and make predictions.

Step 2: open the link in browser:



- Once the web application is open, you'll see input fields corresponding to attribute required for predicting Annual stock price.
- Enter the values for attribute in the respective input tabs.
- Select the month whatever You want to predict.

Step 3: click on “Get Forecast” button



Enter Year and Month for Forecast

Year (YYYY):

Month:

Jan

Forecast Result

The forecasted return for Jul 2025 is **2.83**

- After entering the necessary input values, click on the "Get Forecast" button. This action triggers the Flask app to process the input data using the trained model.
- The predicted value will be displayed on the web page.

9.Final Observation

- We receiving the predicted Annual stock price.