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

PROJECT REPORT

**Data Science with Python Programming**

**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.

**By Team -4**

**Team Details :**

Shivanoor Vignesh 21R11A0597

G.SandhyaRani 21R11A0569

R.Dhathri 21R11A0545

Hasini 21R11A0518

A.Pooja 21R11A0501

PSBN Sriya 21R11A0595

N Sai Charan 21R11A0588

MD.Akbar Ali 21R11A0586

# ABSTRACT

# This project delves into the analysis of stock market data with a focus on Nifty monthly and yearly returns, spanning a period of 20 years. The objective is to discern patterns, particularly emphasizing the year 2020, a significant period marked by the COVID-19 pandemic. Leveraging Python libraries such as Seaborn, Pandas, yfinance, and Matplotlib, the project entails importing the dataset, creating heat maps of monthly returns, and generating histograms to visualize the distribution of monthly and yearly returns across different standard deviation buckets. Additionally, an exploratory data analysis is conducted to derive insightful observations and inferences. Moreover, the project extends to building a web application using the Flask micro web framework to provide an interactive interface for users. This endeavor aims to enhance understanding of stock market dynamics and aid in making informed investment decisions.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No** | **Chapter Name** | **Page No** |
|  | **ABSTRACT** | 2 |
| **1** | **INTRODUCTION** | 4 |
| **2** | **LITERATURE SURVEY** | 5 |
| **3** | **PROBLEM STATEMENT** | 6 |
| **4** | **OBJECTIVES** | 7 |
| **5** | **METHODOLOGY** | 8 |
|  | 5.1 Data Source | 8 |
|  | 5.2 Exploratory Data Analysis | 8 |
|  | 5.2.1 Checking for Data Consistency | 9 |
| **6** | **DATA VISUALIZATION** | 12 |
|  | 6.1Correlation Matrix | 17 |
|  | 6.2 HeatMap | 21 |
|  | 6.3 Distribution of Monthly Returns | 22 |
|  | 6.3 Distribution of Yearly Returns | 23 |
| **7** | **ALGORITHMS** | 24 |
| **8** | **IMPLEMENTATION** | 25 |
|  | 8.1.Random Forest Classifier | 25 |
|  | 8.2.Decision Tree Regression | 26 |
|  | 8.3.Gradient Boosting Regression | 27 |
|  | 8.4. linear Regression | 28 |
| **9** | **Final Implementation** | 29 |
| **10** | **Final Observation** | 32 |

# INTRODUCTION

# The NIFTY 50 is a benchmark Indian stock market index that represents the weighted average of 50 of the largest Indian companies listed on the National Stock Exchange.

# The NIFTY 50 index has shaped up to be the largest single financial product in India, with an ecosystem consisting of exchange-traded funds (onshore and offshore), and futures and options at NSE and SGX. NIFTY 50 is the world's most actively traded contract. WFE, IOM and FIA surveys endorse NSE's leadership position. Between 2008 and 2012, the NIFTY 50 index's share of NSE market fell from 65% to 29%[10] due to the rise of sectoral indices like NIFTY Bank, NIFTY IT, NIFTY Pharma, and NIFTY Next 50.he Nifty experienced 15 crashes during the period 2000 to 2008 with a number of them having occurred in the months of January, May and June 2008. According to SEBI, approximately 89% of individual stock traders in the equity Futures & Options (F&O) segment incurred losses during the financial year

# 2020: COVID-19 Pandemic: In 2020, the world was hit by the COVID-19 pandemic, which caused widespread economic disruption and uncertainty. The pandemic led to lockdowns, travel restrictions, and disruptions in supply chains, which negatively impacted various sectors of the economy, including banking and finance.

# Economic Uncertainty: The uncertainty surrounding the duration and severity of the pandemic led to a sharp decline in economic activity. Banks faced challenges such as increased loan defaults, reduced consumer spending, and lower demand for credit.

# Market Volatility: The pandemic-induced market volatility caused panic selling and a flight to safety among investors. Banks, being sensitive to economic conditions and market sentiment, experienced significant declines in stock prices.

# 2008: Global Financial Crisis: The financial crisis of 2008 was triggered by the collapse of the subprime mortgage market in the United States, leading to a broader banking and financial crisis worldwide.

# Subprime Mortgage Crisis: Banks and financial institutions had invested heavily in mortgage-backed securities tied to subprime mortgages, which defaulted en masse as the housing bubble burst. This led to massive losses for banks and a liquidity crunch in financial markets.

# Credit Crunch: The collapse of major financial institutions and the freezing of credit markets led to a credit crunch, making it difficult for banks to borrow funds or extend credit to businesses and consumers. This further exacerbated the financial turmoil.

# Regulatory Failures: Regulatory failures and lax lending standards allowed the subprime mortgage bubble to inflate unchecked, leading to systemic risks that ultimately culminated in the financial crisis. The crisis prompted regulatory reforms aimed at strengthening oversight of the banking and financial sectors to prevent future crises.

# 

# PROBLEM STATEMENT

# The effective management of investment portfolios hinges on the ability to discern opportune moments for stock entry or exit, a task contingent upon astute analysis of stock market data. In this context, the project endeavors to undertake a comprehensive examination of stock market data, focusing specifically on the Nifty index. The dataset contains 20years of stock price related data of nifty monthly returns data

# The dataset provided encapsulates two decades of stock price data, encompassing monthly and yearly returns of Nifty. Despite its breadth, the project prioritizes the year 2020, aiming to unveil intricate patterns amidst the backdrop of the COVID-19 pandemic.

# Objectives

# Import the dataset into Python to facilitate further analysis and exploration.

# Construct a heat map to visualize Nifty's monthly returns over the designated timeframe, providing insights into temporal fluctuations and trends.

# Categorize monthly returns into distinct buckets based on standard deviations and generate a histogram to illustrate the distribution of returns. This segmentation allows for a nuanced understanding of return variability and the prevalence of outlier events.

# Create a histogram depicting the distribution of yearly returns, similar to the monthly analysis, by categorizing returns into standard deviation-based buckets. This macroscopic view of Nifty's performance on an annual basis aids in identifying trends and anomalies.

# Develop a WEBUI or application using the Flask micro web framework to extend the project beyond mere analysis. This interactive platform enables stakeholders to engage with the analyzed data, facilitating a deeper understanding of Nifty's performance and supporting informed investment decisions.

# 4. METHODOLOGY

**5.1 Data Source**

* **Brief description of the data source**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Feature name** | **Description** |
| **1.** | **Year** | 20 years 2000 to 2022 |
| **2.** | **Jan** | Stock price in January |
| **3.** | **Feb** | Stock price in February |
| **4.** | **Mar** | Stock price in March |
| **5.** | **Apr** | Stock price in April |
| **6.** | **May** | Stock price in May |
| **7.** | **Jun** | Stock price in June |
| **8.** | **Jul** | Stock price in July |
| **9.** | **Aug** | Stock price in August |
| **10.** | **Sep** | Stock price in September |
| **11.** | **Oct** | Stock price in October |
| **12.** | **Nov** | Stock price in November |
| **13.** | **Dec** | Stock price in December |
| **14.** | **Annual** | Total stock price of whole year |

**5.2 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

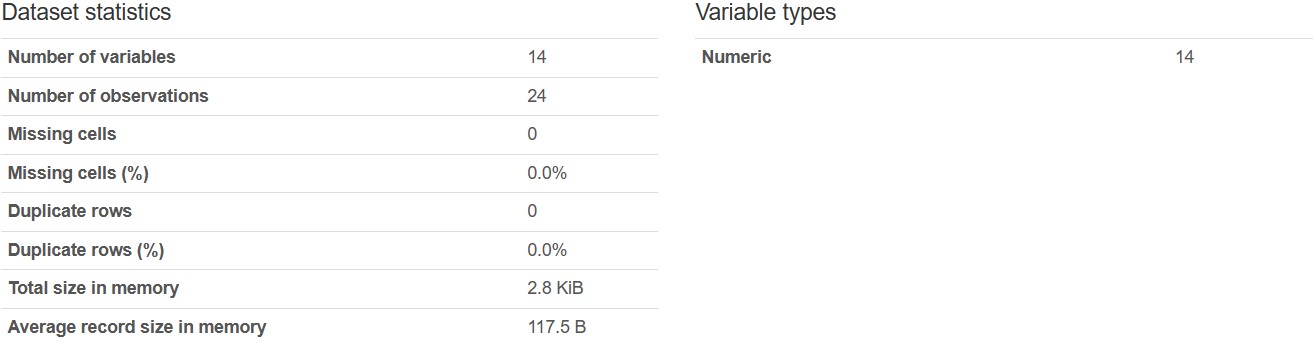
Column No.12: Nov, Data Type: float64

Column No.13: Dec, Data Type: float64

Column No.14: Annual, Data Type: float64

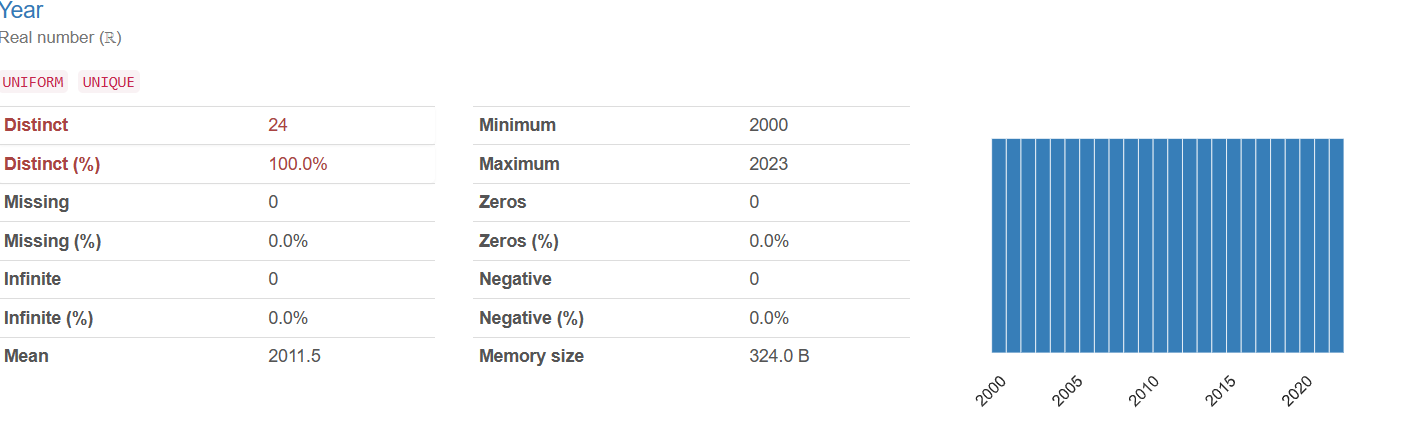
The dataset consists of 14 columns. Column No.1 represents the year and is of integer data type. Columns No.2 to No.13 represent the months from January to December, each of float64 data type. Column No.14 represents the annual data and is also of float64 data type.

**5.2.2 Checking for Data Consistency**



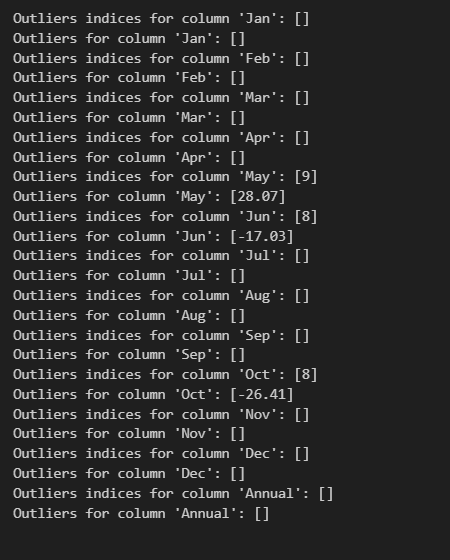
The dataset contains 14 variables, all of which are numeric. It comprises 24 observations with no missing cells or duplicate rows. In memory, the dataset occupies 2.8 KiB, with an average record size of 117.5 bytes.

Year



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

**Outliers:**



**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:

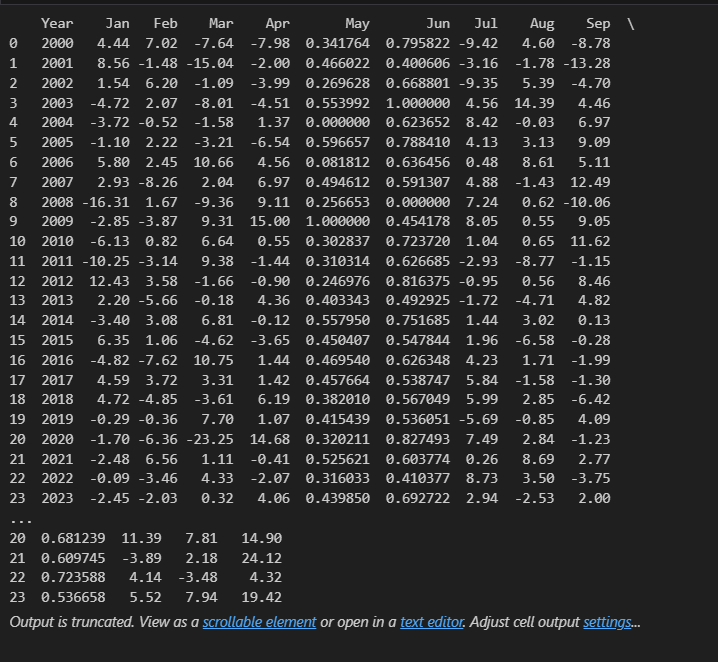
There is one outlier detected at index 8 with the value -17.03.

* 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.

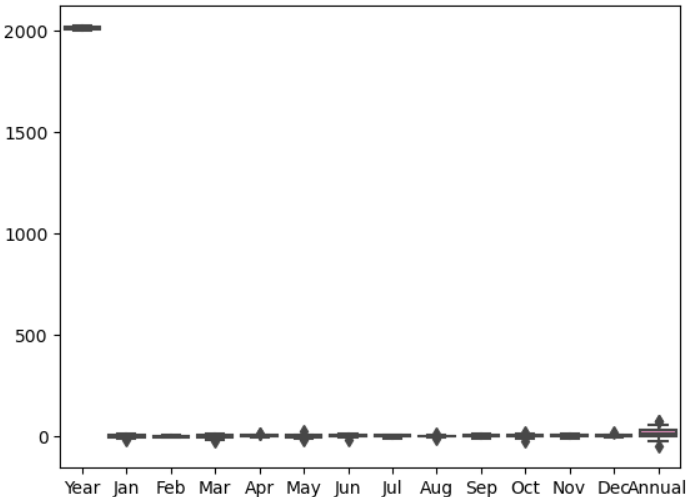
**Normalization:**



Observations:

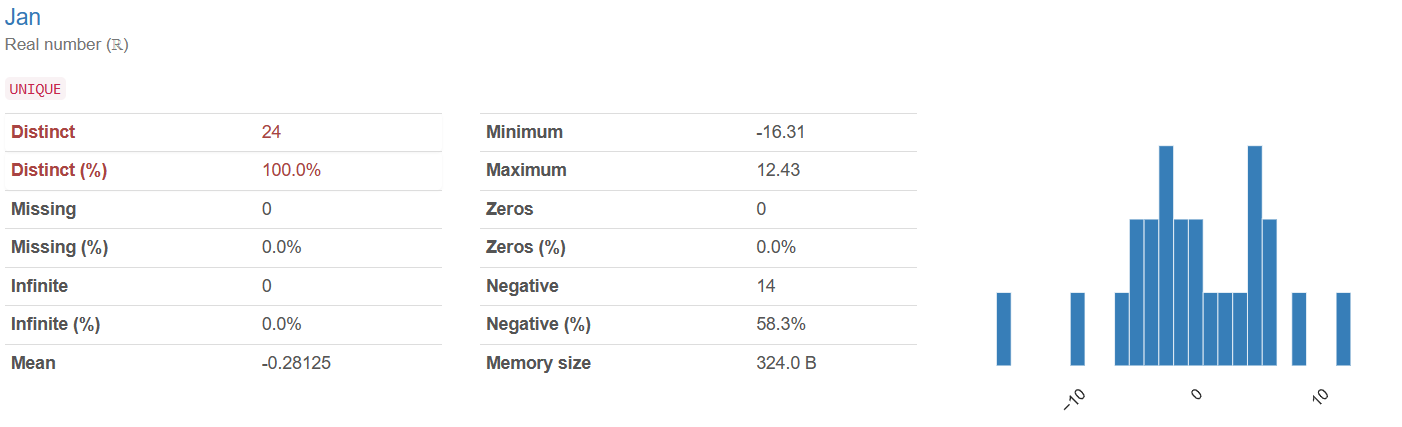
* The Min-Max scaling method has been applied to the columns 'May', 'Jun', and 'Oct' of the DataFrame.
* After scaling, the values in these columns have been transformed to a range between 0 and 1, ensuring that each feature contributes equally to the analysis and preventing features with larger scales 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.

**6. 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**

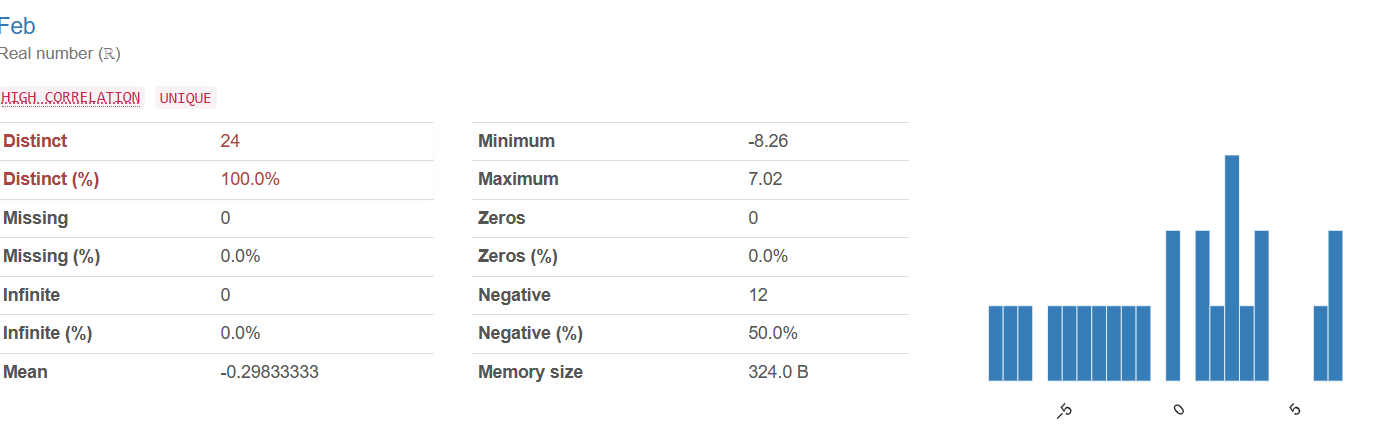


The "Jan" variable consists of real numbers (ℝ), with 24 distinct values covering the entire range.

There are no missing or infinite values. The mean value is -0.28125, with a minimum of -16.31 and a

maximum of 12.43. There are no zero values, but 14 values are negative, constituting 58.3% of the

dataset. The memory size occupied by this variable is 324.0 bytes.

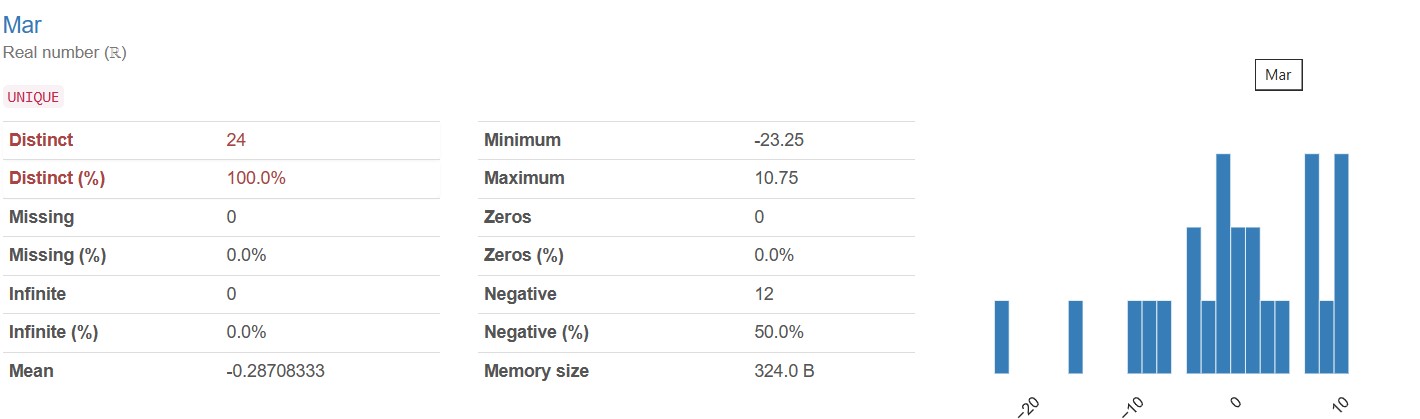


The "Feb" variable consists of real numbers (ℝ) with high correlation, and all 24 values are unique.

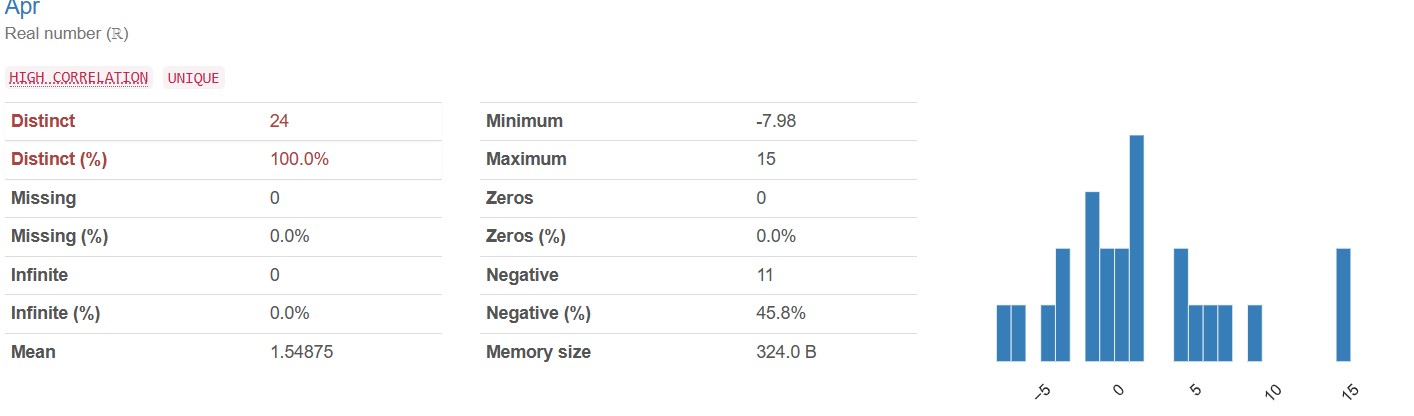
There are no missing or infinite values. The mean value is -0.29833333, with a minimum of -8.26 and a

maximum of 7.02. There are no zero values, but 12 values are negative, accounting for 50.0% of the

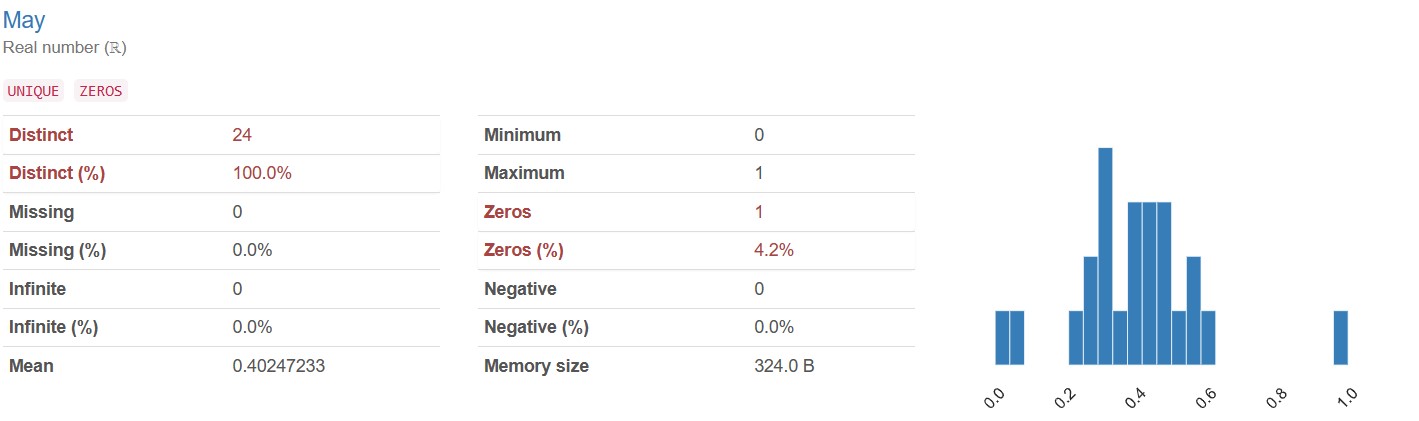
dataset. The memory size occupied by this variable is 324.0 bytes.



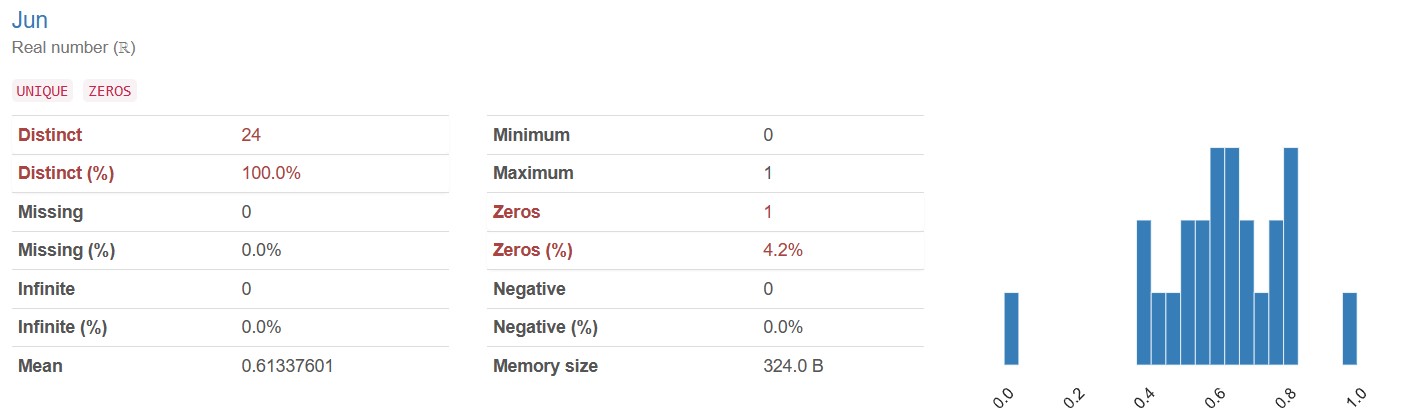
The "Mar" variable consists of real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is -0.28708333, with a minimum of -23.25 and a maximum of 10.75. There are no zero values, but 12 values are negative, making up 50.0% of the dataset. The memory size occupied by this variable is 324.0 bytes.



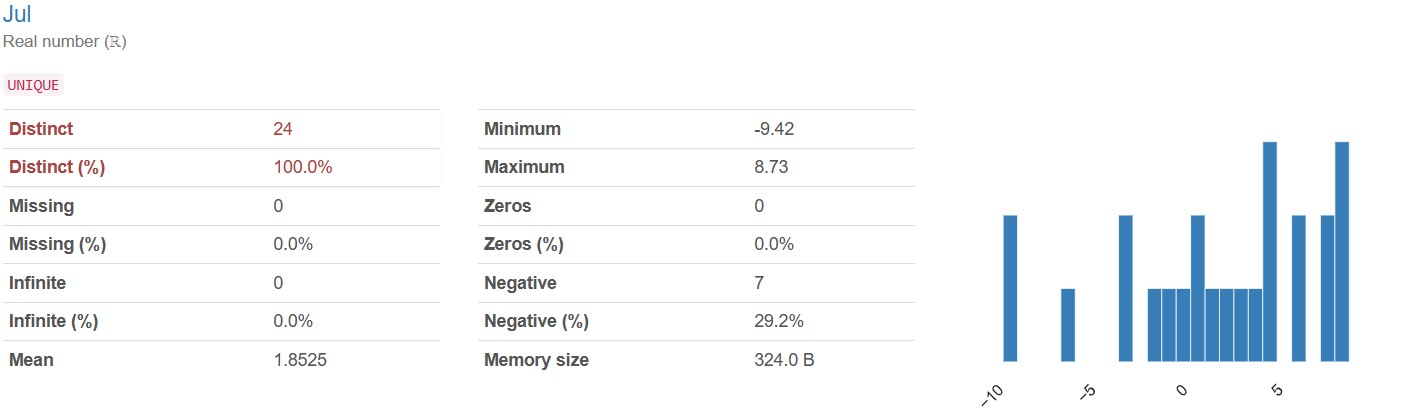
The "Apr" variable consists of real numbers (ℝ) with high correlation, and all 24 values are unique. There are no missing or infinite values. The mean value is 1.54875, with a minimum of -7.98 and a maximum of 15. There are no zero values, but 11 values are negative, accounting for 45.8% of the dataset. The memory size occupied by this variable is 324.0 bytes.



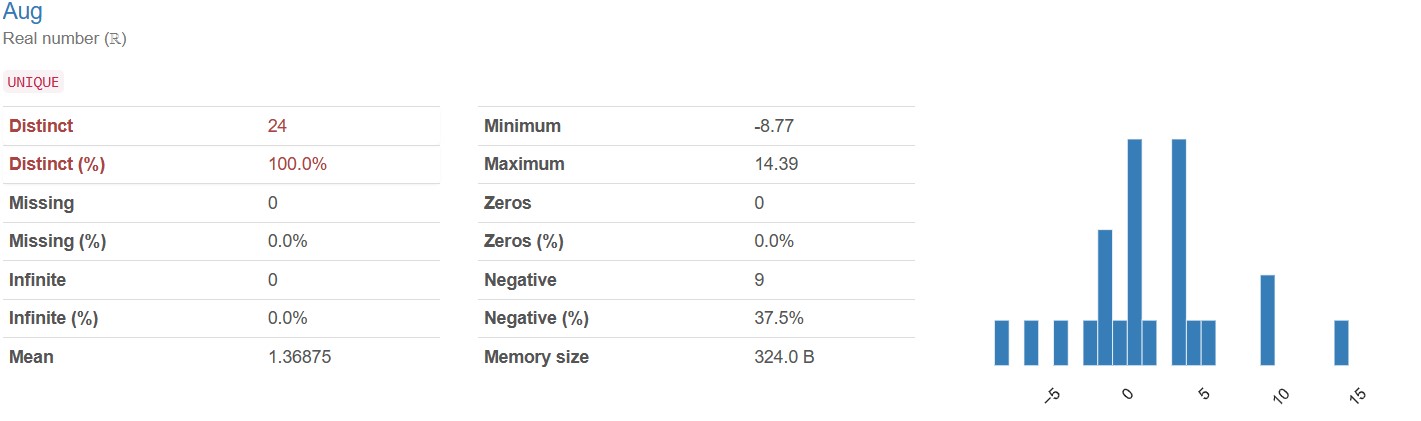
The "May" variable consists of real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is 0.40247233, with a minimum of 0 and a maximum of 1. There is one zero value, which constitutes 4.2% of the dataset. There are no negative values present. The memory size occupied by this variable is 324.0 bytes.



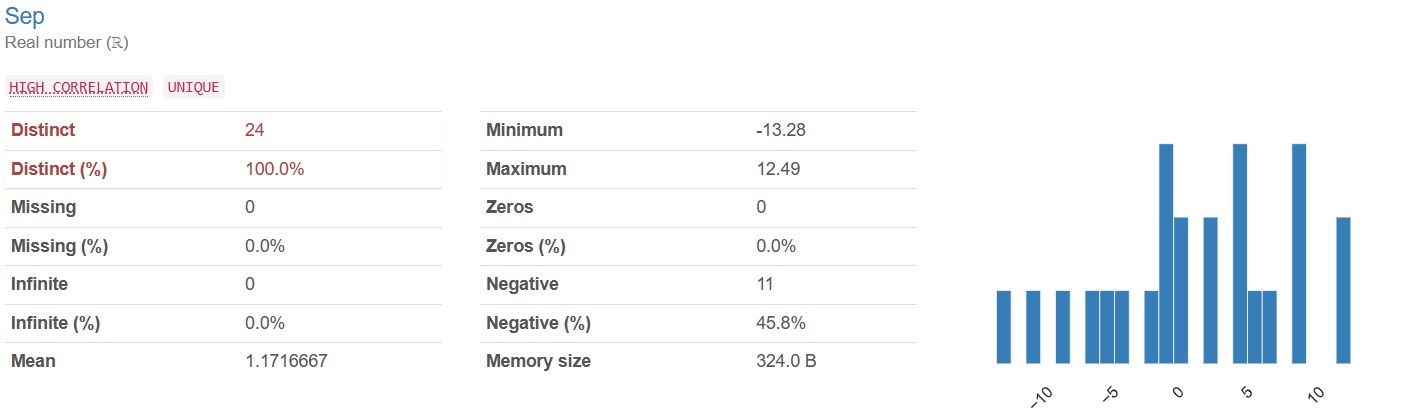
The "Jun" variable comprises real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is 0.61337601, with a minimum of 0 and a maximum of 1. There is one zero value, accounting for 4.2% of the dataset. No negative values are present. The memory size occupied by this variable is 324.0 bytes.



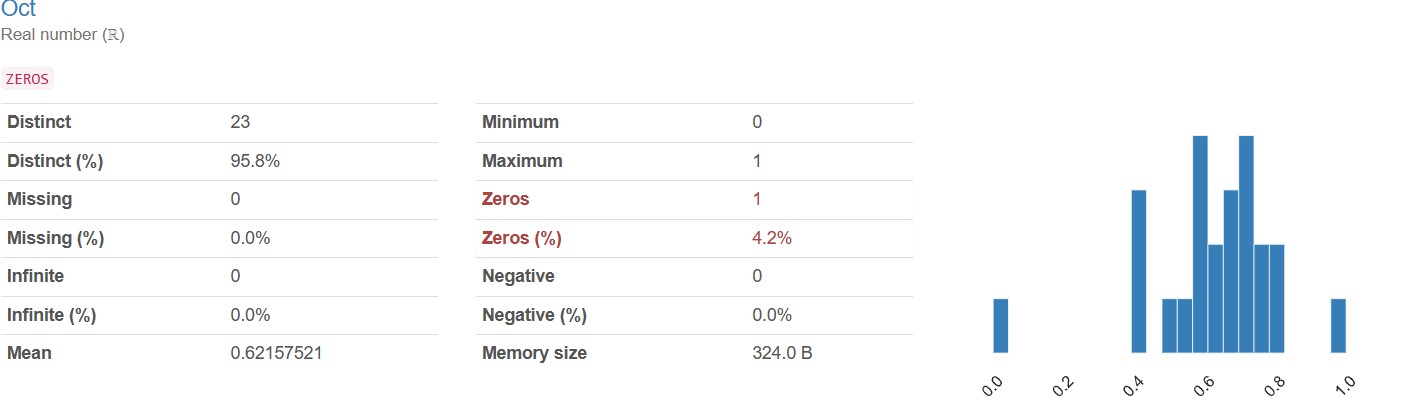
The "Jul" variable consists of real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is 1.8525, with a minimum of -9.42 and a maximum of 8.73. There are no zero values present, but 7 values are negative, making up 29.2% of the dataset. The memory size occupied by this variable is 324.0 bytes.



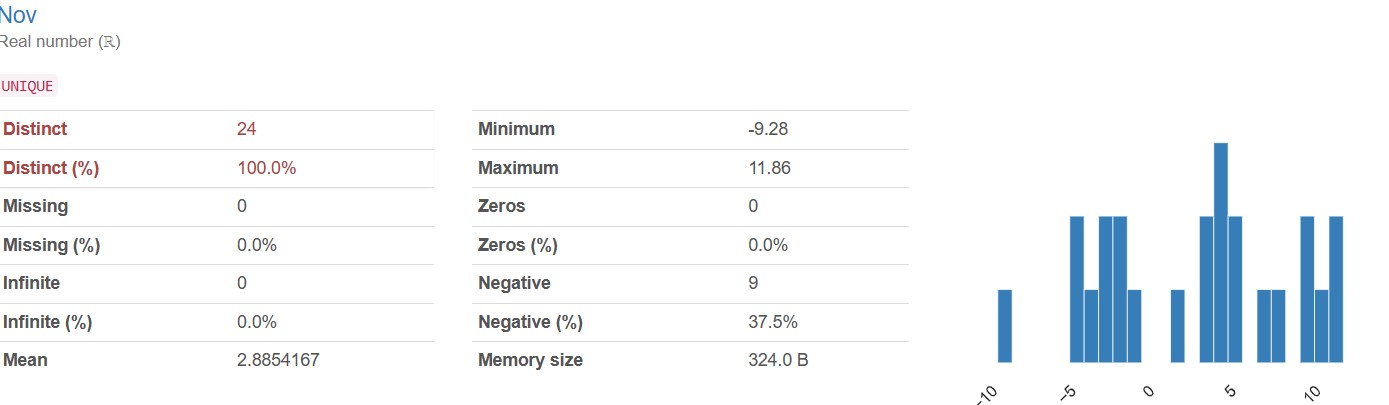
The "Aug" variable comprises real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is 1.36875, with a minimum of -8.77 and a maximum of 14.39. There are no zero values present, but 9 values are negative, accounting for 37.5% of the dataset. The memory size occupied by this variable is 324.0 bytes.



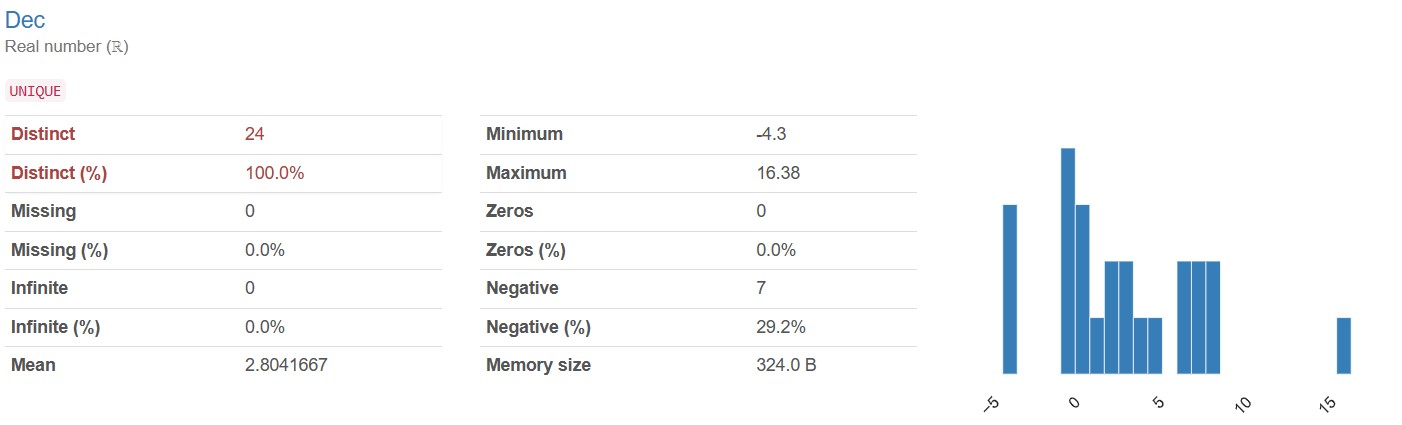
The "Sep" variable consists of real numbers (ℝ) with high correlation, and all 24 values are unique. There are no missing or infinite values. The mean value is 1.1716667, with a minimum of -13.28 and a maximum of 12.49. There are no zero values present, but 11 values are negative, comprising 45.8% of the dataset. The memory size occupied by this variable is 324.0 bytes.



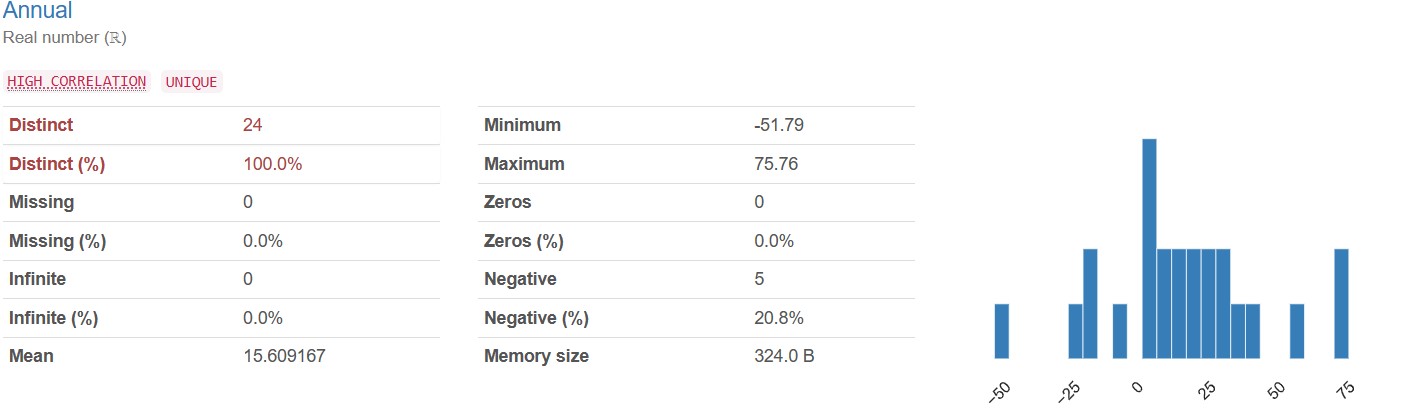
The "Oct" variable consists of real numbers (ℝ), with 23 distinct values covering 95.8% of the dataset. There are no missing or infinite values. The mean value is 0.62157521, with a minimum of 0 and a maximum of 1. There is one zero value, accounting for 4.2% of the dataset. No negative values are present. The memory size occupied by this variable is 324.0 bytes.



The "Nov" variable comprises real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is 2.8854167, with a minimum of -9.28 and a maximum of 11.86. There are no zero values present, but 9 values are negative, accounting for 37.5% of the dataset. The memory size occupied by this variable is 324.0 bytes.

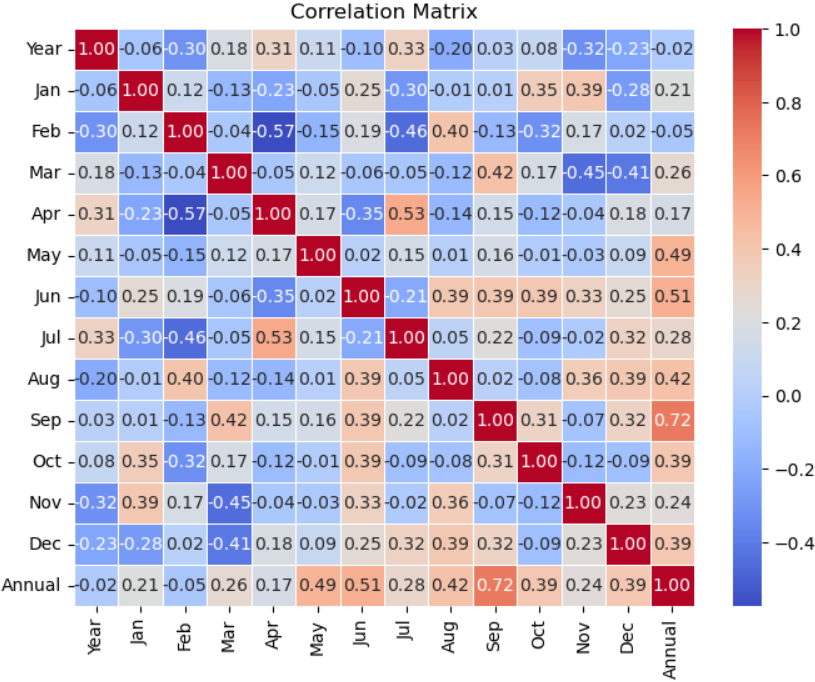


The "Dec" variable consists of real numbers (ℝ), with all 24 values being unique. There are no missing or infinite values. The mean value is 2.8041667, with a minimum of -4.3 and a maximum of 16.38. There are no zero values present, but 7 values are negative, comprising 29.2% of the dataset. The memory size occupied by this variable is 324.0 bytes.



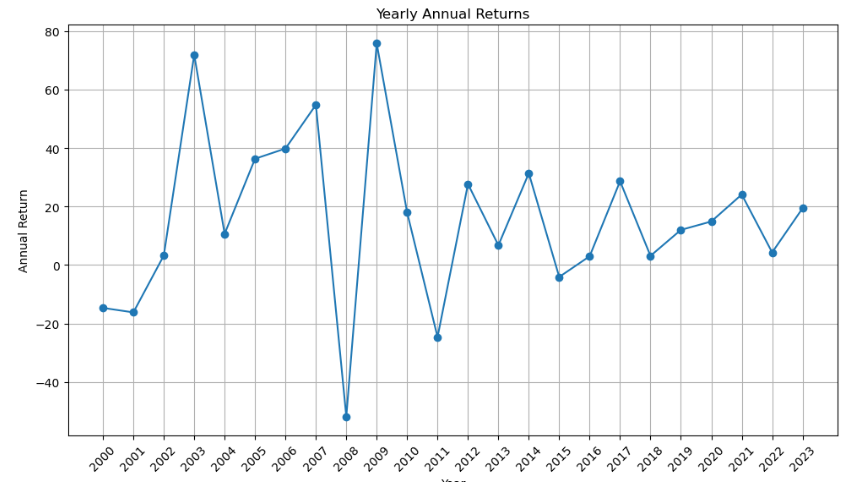
The "Annual" variable comprises real numbers (ℝ) with high correlation, and all 24 values are unique. There are no missing or infinite values. The mean value is 15.609167, with a minimum of -51.79 and a maximum of 75.76. There are no zero values present, but 5 values are negative, accounting for 20.8% of the dataset. The memory size occupied by this variable is 324.0 bytes.

**6.1 Correlation Matrix**

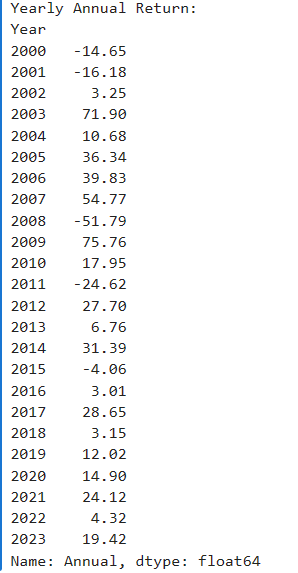


1. The graph depicts the **annual returns** from **2000 to 2023**. Here are some observations based on the data:
   * The returns exhibit **high volatility**, with significant peaks and troughs over the years.
   * Notable **peaks**:
     + **2001**: The return reached close to **+80%**.
     + **2005**: Another peak year with substantial positive returns.
     + **2010**: Yet another year of strong performance.
   * Significant **troughs**:
     + **2002**: The return dropped significantly, nearing **-40%**.
     + **2008**: A challenging year with substantial negative returns.
     + **2011**: Another year of poor performance.
   * Since around **2012**, the volatility has decreased, and returns have fluctuated moderately around the **zero mark**.
2. **Annual return** is a valuable metric for evaluating the profitability and growth of an investment. It reflects both **capital appreciation** and **dividends or income received** during a specific year. [Investors use it to assess the performance of their portfolio or individual assets within a specific timeframe1](https://www.wallstreetmojo.com/annual-return/).
3. Various methods can calculate annual returns, such as:
   * **Simple Annual Return**: Calculated by subtracting the beginning value from the ending value, dividing by the beginning value, and multiplying by 100.
   * **Compound Annual Growth Rate (CAGR)**: Adjusts for compounding interest and provides a more complete picture of returns over time.
   * [**Total Return**: Considers both capital gains and income received during the year1](https://www.wallstreetmojo.com/annual-return/)[2](https://upstox.com/market-talk/annual-vs-trailing-vs-rolling-returns-explained/).

**Yearly Annual Returns**

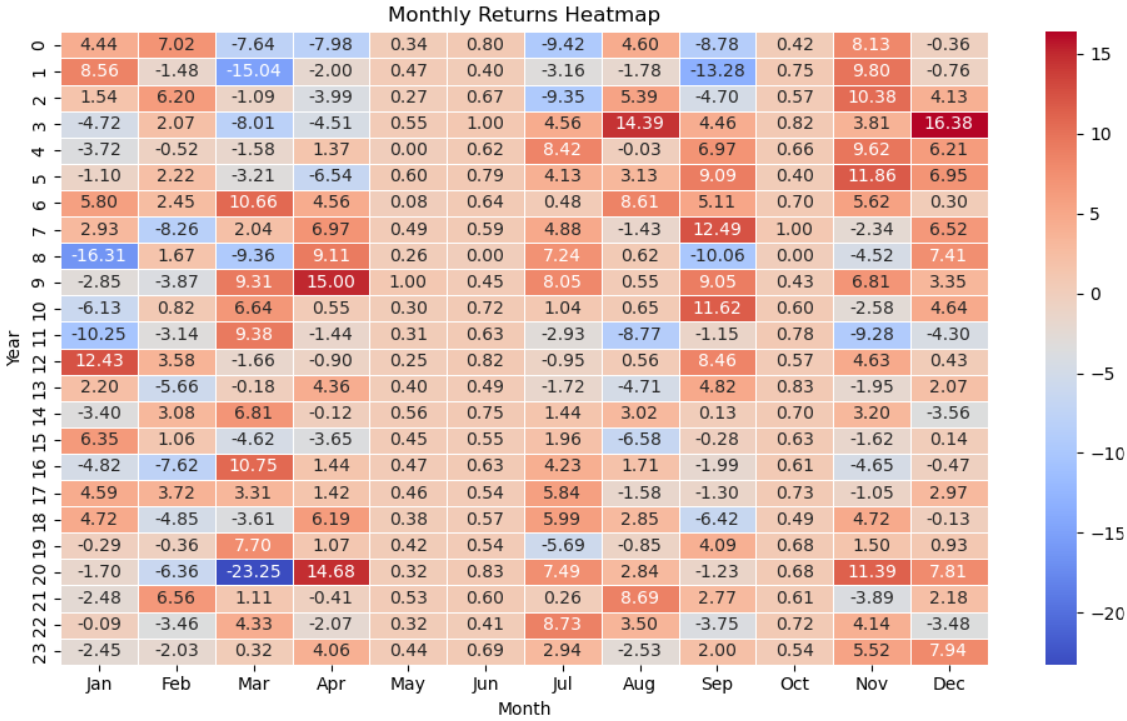


1. The graph depicts the **annual returns** from **2000 to 2023**. Here are some observations based on the data:
   * The returns exhibit **high volatility**, with significant peaks and troughs over the years.
   * Notable **peaks**:
     + **2001**: The return reached close to **+80%**.
     + **2005**: Another peak year with substantial positive returns.
     + **2010**: Yet another year of strong performance.
   * Significant **troughs**:
     + **2002**: The return dropped significantly, nearing **-40%**.
     + **2008**: A challenging year with substantial negative returns.
     + **2011**: Another year of poor performance.
   * Since around **2012**, the volatility has decreased, and returns have fluctuated moderately around the **zero mark**.
2. **Annual return** is a valuable metric for evaluating the profitability and growth of an investment. It reflects both **capital appreciation** and **dividends or income received** during a specific year. [Investors use it to assess the performance of their portfolio or individual assets within a specific timeframe1](https://www.wallstreetmojo.com/annual-return/).
3. Various methods can calculate annual returns, such as:
   * **Simple Annual Return**: Calculated by subtracting the beginning value from the ending value, dividing by the beginning value, and multiplying by 100.
   * **Compound Annual Growth Rate (CAGR)**: Adjusts for compounding interest and provides a more complete picture of returns over time.
   * [**Total Return**: Considers both capital gains and income received during the year1](https://www.wallstreetmojo.com/annual-return/)[2](https://upstox.com/market-talk/annual-vs-trailing-vs-rolling-returns-explained/).



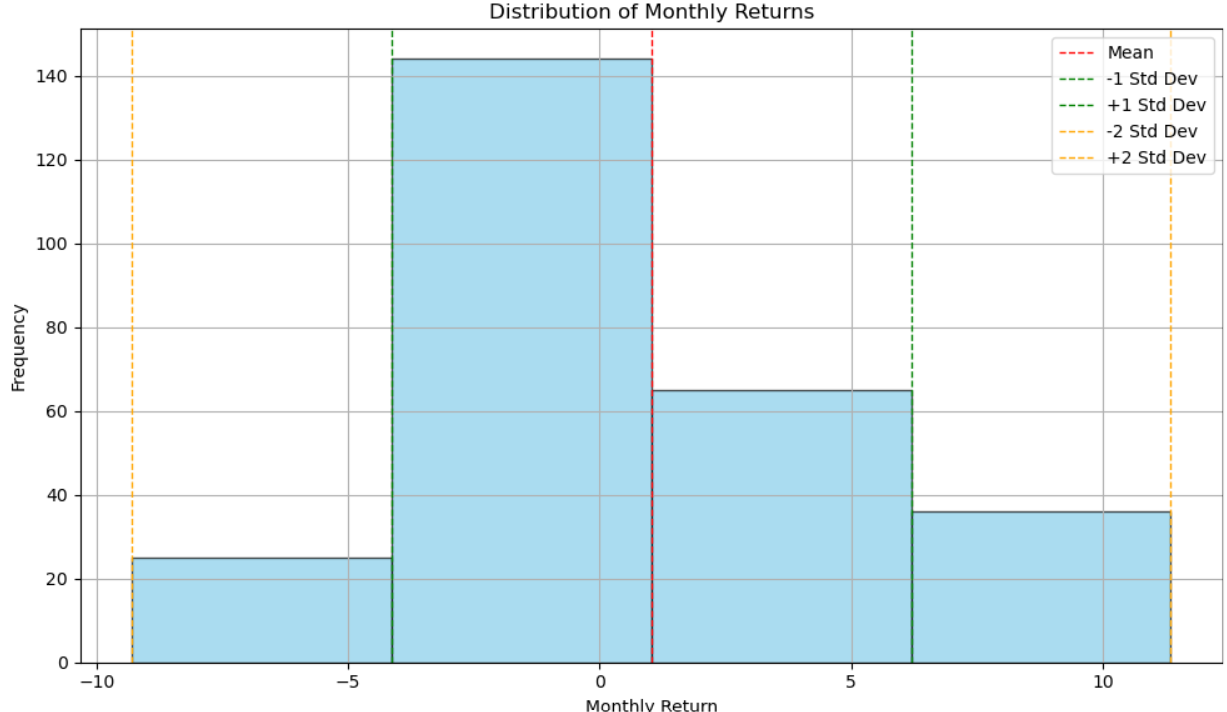
The dataset provides yearly annual return percentages from 2000 to 2023. Returns range from a loss of -51.79% in 2008 to a gain of 75.76% in 2009. Other notable years include 2003 with a gain of 71.90%, 2007 with a gain of 54.77%, and 2011 with a loss of -24.62%. Positive values indicate gains, while negative values represent losses.

**6.2 Monthly Return HeatMap**



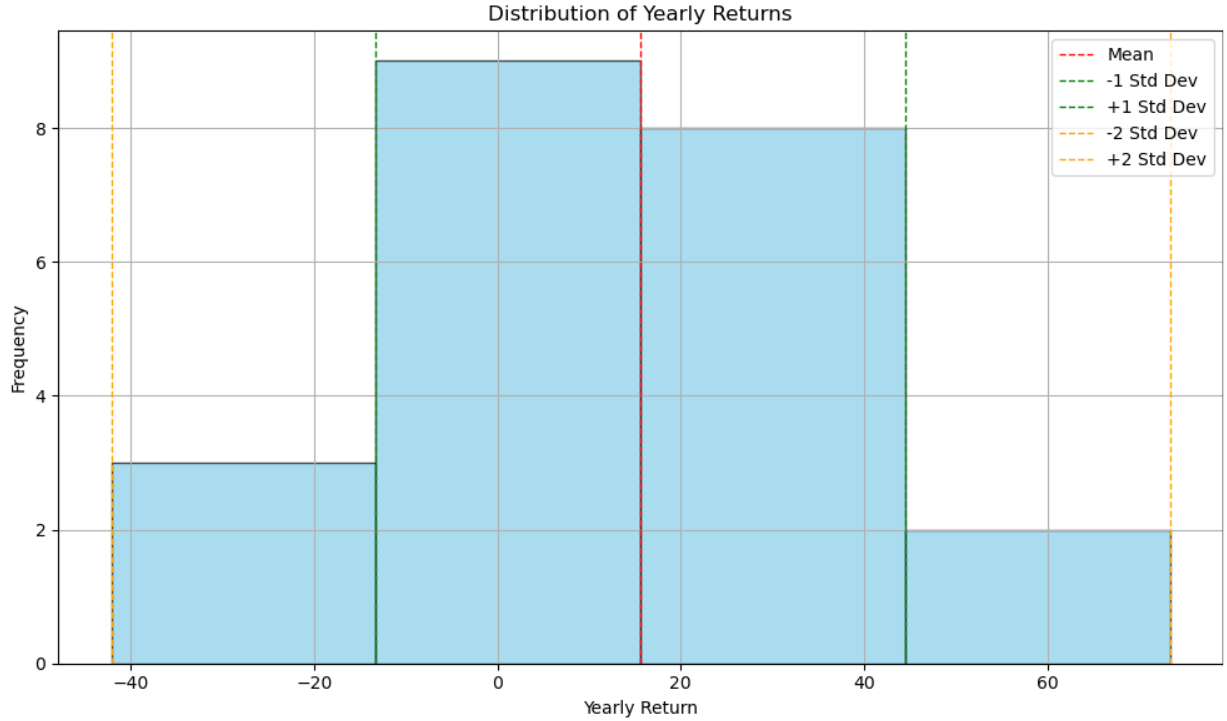
1. The heatmap illustrates the **monthly returns** of an investment or portfolio over **22 years**. Here are the key observations:
   * **January (Jan)**: The return is **-4.44%**.
   * **February (Feb)**: Positive return of **7.02%**.
   * **March (Mar)**: Strong performance with a return of **-7.98%**.
   * **April (Apr)**: Another positive month, returning **0.34%**.
   * **May**: A modest return of **0.80%**.
   * **June (Jun)**: Slightly better at **-9.42%**.
   * **July (Jul)**: Impressive return of **4.60%**.
   * **August (Aug)**: Decent performance with a return of **-8.78%**.
   * **September (Sep)**: Robust return at **8.13%**.
   * **October (Oct)**: A small positive return of **0.76%**.
   * **November (Nov)**: Strong month with a return of **8.56%**.
   * **December (Dec)**: Another modest return of **-1.48%**.
2. The heatmap’s color intensity reflects the returns:
   * **Red** indicates negative returns.
   * **Blue** represents positive returns.
3. Over the years, we observe fluctuations in monthly returns. Some months stand out for their exceptional performance, while others remain more subdued.

**6.3 Distribution of Monthly Returns**



1. The heatmap displays monthly returns over **22 years**. Each cell’s color intensity represents the return value, with **red** indicating **negative returns** and **blue** indicating **positive returns**.
2. Here are some observations from the heatmap:
   * **January (Jan)**: The return is **-4.44%**.
   * **February (Feb)**: Positive return of **7.02%**.
   * **March (Mar)**: Strong performance with a return of **-7.98%**.
   * **April (Apr)**: Another positive month, returning **0.34%**.
   * **May**: A modest return of **0.80%**.
   * **June (Jun)**: Slightly better at **-9.42%**.
   * **July (Jul)**: Impressive return of **4.60%**.
   * **August (Aug)**: Decent performance with a return of **-8.78%**.
   * **September (Sep)**: Robust return at **8.13%**.
   * **October (Oct)**: A small positive return of **0.76%**.
   * **November (Nov)**: Strong month with a return of **8.56%**.
   * **December (Dec)**: Another modest return of **-1.48%**.
3. The heatmap’s color variations indicate fluctuations in monthly returns over the years.

**6.4 Distribution of Yearly Returns**



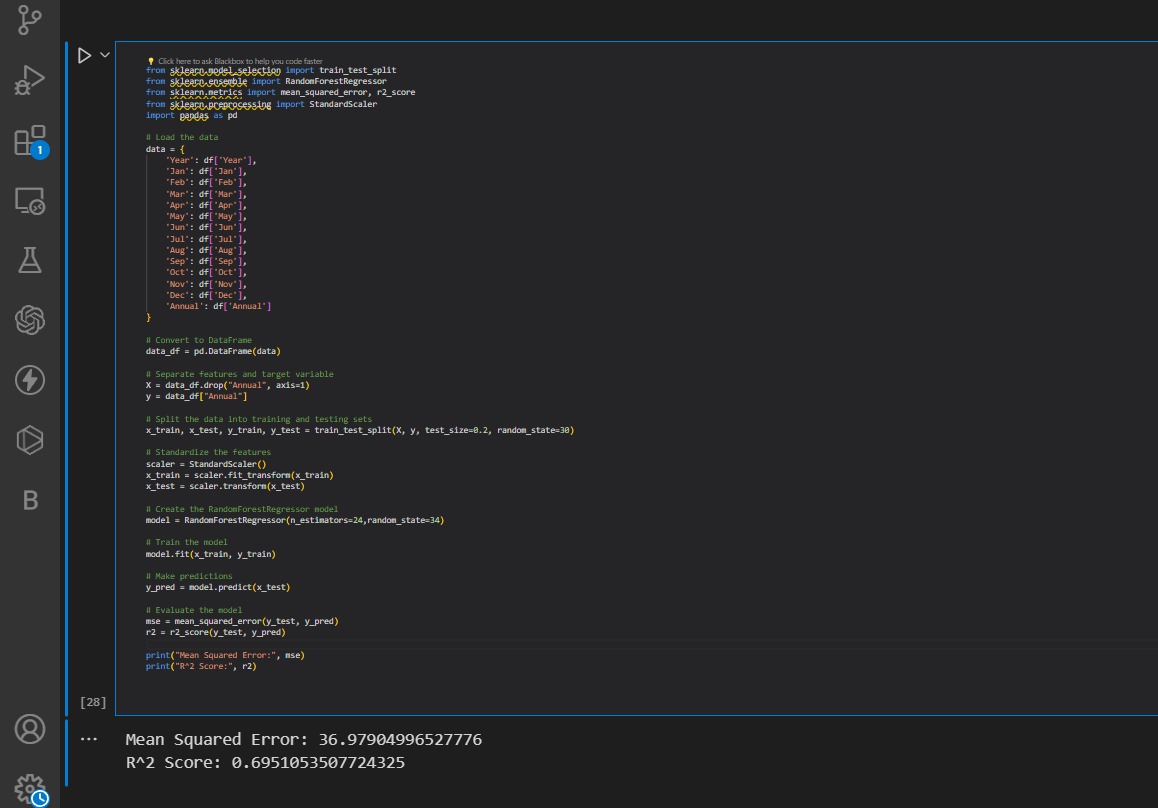
1. The histogram displays the distribution of yearly returns over a period of time. Here are the key observations:
   * The x-axis represents yearly return percentages, ranging from **-40% to 60%**.
   * The y-axis represents **frequency**, indicating how often each return value occurs.
   * The histogram shows the concentration of data points around the **mean** (average) yearly return, which is indicated by the **solid red line**.
   * Two standard deviation lines are present:
     + **+1 standard deviation** (green dashed line) and **-1 standard deviation** (green dashed line) from the mean.
     + **+2 standard deviations** (yellow dashed line) and **-2 standard deviations** (yellow dashed line) from the mean.
2. Key insights:
   * Most of the data falls within **one standard deviation** from the mean.
   * The distribution is **approximately symmetric**, with a peak around the mean return of **0%**.
   * The spread of returns becomes wider as we move away from the mean.
3. Remember that understanding the distribution of returns is crucial for risk assessment and portfolio optimization.

**7.ALGORITHMS**

1. **Random Forest Classifier**
2. **Decision Tree Regression**
3. **Gradient Boosting Regression**
4. **linear regression**

**8.Implementation**

**8.1 Random Forest Classifier**

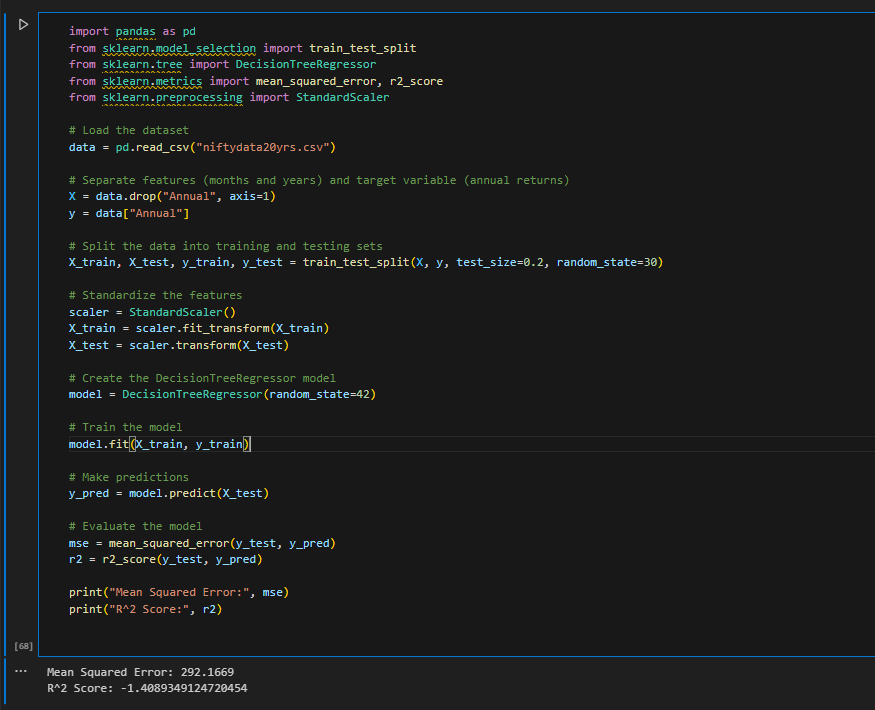


**Accuracy: 69.5%**

Observations:

* The mean squared error (MSE) is approximately 36.98. This value indicates the average squared deviation of the predicted annual stock prices from the actual values.
* The R-squared score (R^2) is approximately 0.695. This indicates that approximately 69.5% of the variance in the annual stock prices is explained by the independent variables included in the model.
* These metrics suggest that the RandomForestRegressor model provides reasonably good predictions of annual stock prices based on monthly data.

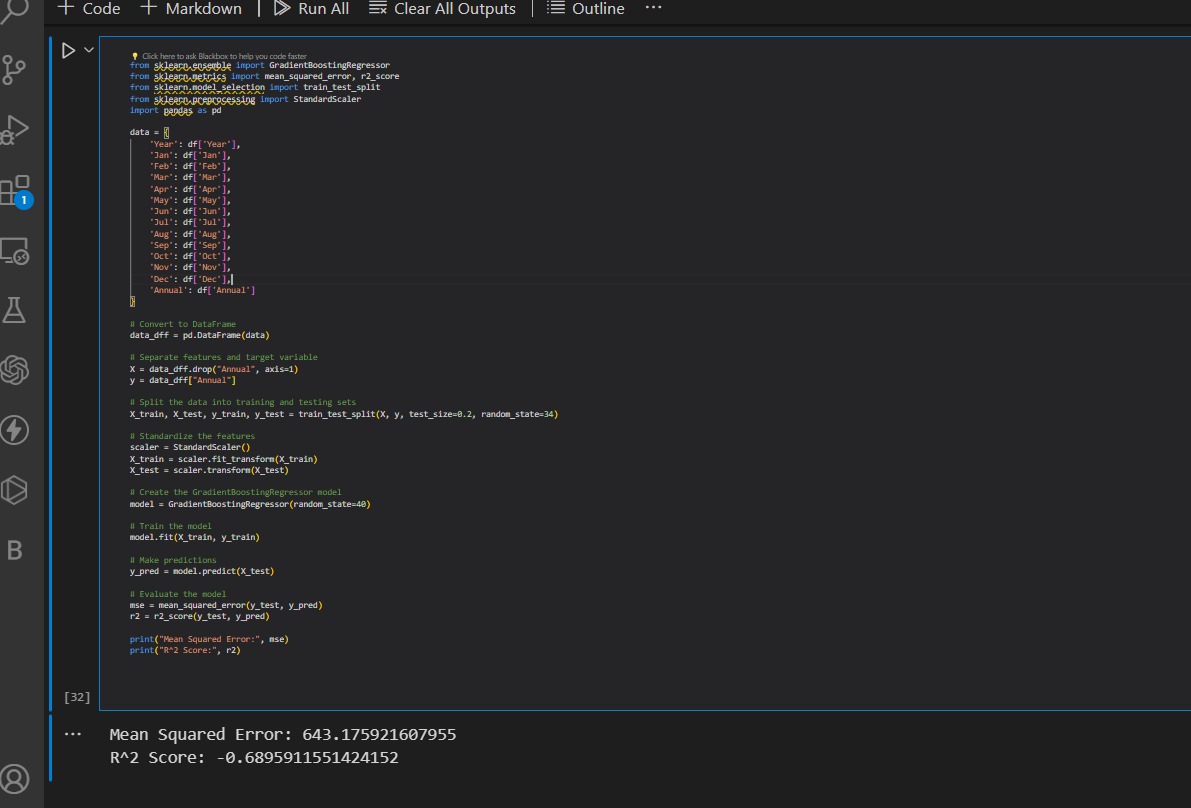
**8.2 Decision Tree Regression**



Observations:

* The mean squared error (MSE) is approximately 292.17. This value indicates a moderate average squared deviation of the predicted annual returns from the actual values.
* The R-squared score (R^2) is approximately -1.41. A negative R-squared score suggests that the model performs worse than a simple horizontal line, indicating poor predictive performance.
* These metrics suggest that the DecisionTreeRegressor model performs poorly in predicting annual returns based on the given features. The moderate MSE and negative R-squared score indicate that the model's predictions are not accurate and fail to capture the variance in the target variable.

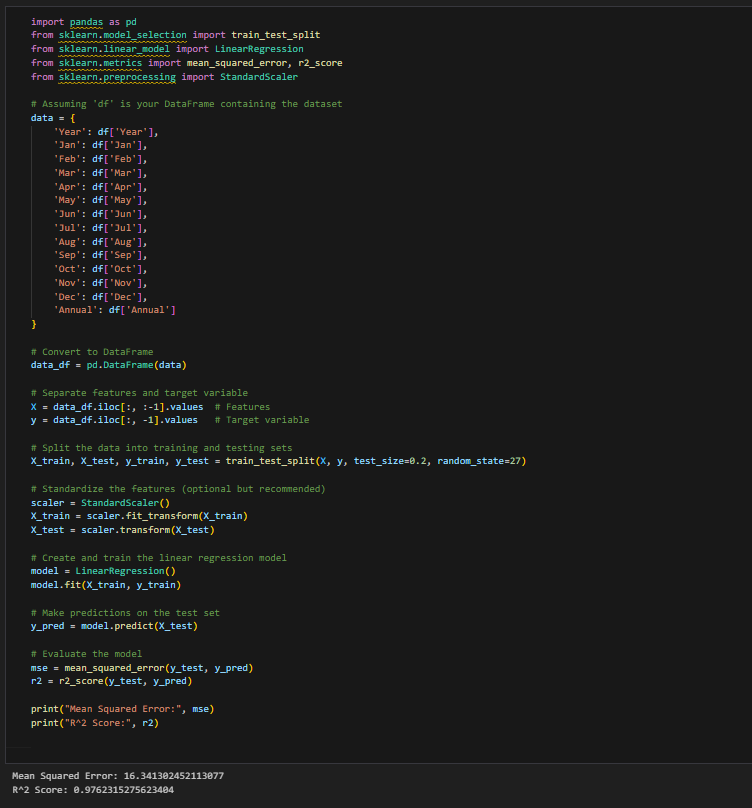
**8.3 Gradient Boosting Regression**

****

Observations:

* The mean squared error (MSE) is approximately 643.18. This value indicates a moderate average squared deviation of the predicted annual returns from the actual values.
* The R-squared score (R^2) is approximately -0.69. A negative R-squared score suggests that the model performs worse than a simple horizontal line, indicating poor predictive performance.
* These metrics suggest that the GradientBoostingRegressor model performs poorly in predicting annual returns based on the given features. The moderate MSE and negative R-squared score indicate that the model's predictions are not accurate and fail to capture the variance in the target variable.

**8.4 linear Regression**



**Accuracy: 97.6%**

Observations:

* The mean squared error (MSE) is approximately 16.34. This value indicates a low average squared deviation of the predicted annual returns from the actual values.
* The R-squared score (R^2) is approximately 0.976. This high R-squared score suggests that approximately 97.6% of the variance in the annual returns is explained by the linear regression model.
* These metrics indicate that the Linear Regression model performs well in predicting annual returns based on the given features. The low MSE and high R-squared score suggest that the model's predictions are accurate and effectively capture the variance in the target variable.

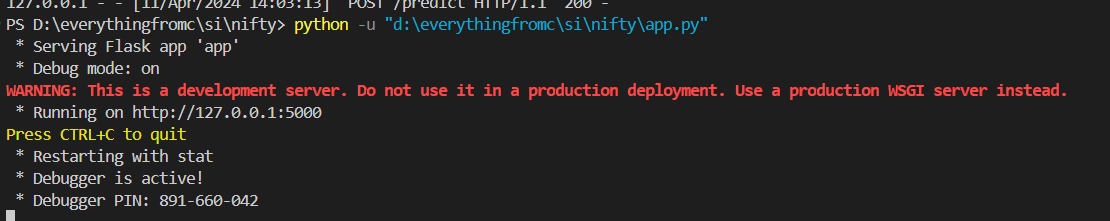
**Overall Observation:**

* We can clearly see linear Regression has outperformed all other models.
* Accuracy is excellent for linear regression.
* So the below model is built based on this classifier.
* The Linear Regression model demonstrates strong performance in predicting annual returns.
* Its predictions are accurate and effectively capture the underlying patterns in the data, as evidenced by the low MSE and high R-squared score.

**9. Final implementation of model Using Flask Website**

**Linear Regression**

**Step 1: run the python code app.py**

****

**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.

**Write** **Linear Regression and Model. pkl File:**

* Create a Python script named linearmodel.py.
* Write Python code in this script to implement the linear regression for predicting the Annual return .
* Once the classifier is trained and ready, save it into a file named linearmodel.pkl using the pickle library.

**Implement Linear Regression**:

* In the linearmodel.py script, write code to load the dataset, preprocess it, train the data, Normalize the linear regression, and evaluate its performance.
* After training, save the trained classifier to the linearmodel.pkl file using the pickle.dump() function.

**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 all 13 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:**

# 

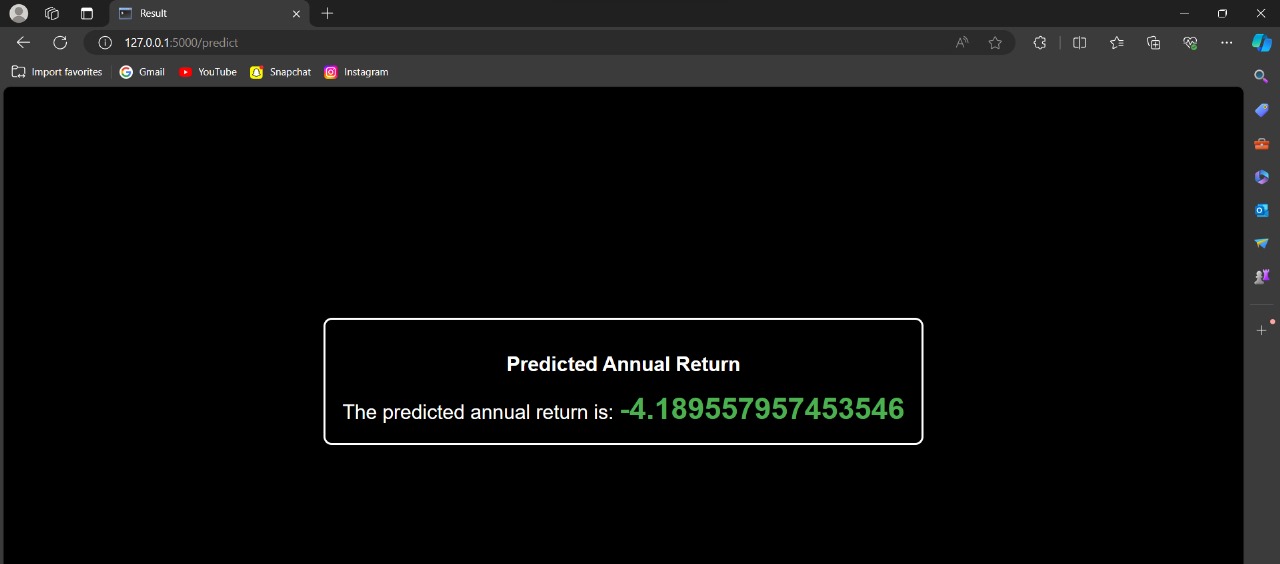
* Once the web application is open, you'll see input fields corresponding to each attribute required for predicting Annual stock price.
* Enter the values for each attribute in the respective input tabs.

# Step 3: click on prediction button

# 

* After entering all the necessary input values, click on the "Prediction" button. This action triggers the Flask app to process the input data using the trained model.

**Step 4: get the predicted value of Annual stock price every year**

****

Upon clicking the "Prediction" button, the Flask app will utilize the linear regression to predict the annual stock price.

The predicted values will be displayed on the web page.

**10.Final Observation**

* Upon receiving the predicted Annual stock price.