1.Introduction:

In this analysis, we delve into the historical stock data of Canoo Inc. (ticker symbol: GOEV) obtained through web scraping using the WebScraper.io tool. The primary objective of this analysis is to perform both trend analysis and demand forecasting based on the historical stock data.

Data Source:

The historical stock data was sourced from Nasdaq's Market Activity page for Canoo Inc. This dataset encompasses a comprehensive record of Canoo's stock performance over a specific timeframe, providing valuable insights into the company's historical trends.

Relevance to Demand Forecasting:

The analysis focuses on leveraging historical stock data for demand forecasting, a crucial aspect in understanding market trends and anticipating future demands. By applying analytical techniques and models to this dataset, we aim to uncover patterns, make informed predictions, and contribute valuable insights to the domain of demand forecasting.

The utilization of the AutoRegressive Integrated Moving Average (ARIMA) model is particularly noteworthy in this analysis. ARIMA is a powerful time-series forecasting technique that can be instrumental in predicting future stock trends. Through this, we endeavor to not only understand the historical patterns in Canoo's stock performance but also forecast potential future trends.

2.Data Collection:

2.1 Data Source:

The primary source of data for this analysis is Nasdaq's Market Activity page dedicated to Canoo Inc. (ticker symbol: GOEV) [^1^]. The historical stock data, crucial for understanding the market dynamics, was accessed from the official Nasdaq website. The dataset spans a range of dates, capturing the variations in Canoo's stock performance over time.

2.2 Collection Method:

Web scraping was employed to extract the historical stock data from Nasdaq's Market Activity page. The WebScraper.io tool facilitated this process by navigating through the webpage's structure and efficiently extracting relevant information. The data was then compiled into a structured format for subsequent analysis.

2.3 Challenges Faced:

While extracting the data, some challenges were encountered, primarily related to the dynamic nature of web content. The necessity to navigate through different sections of the webpage and adapt to potential changes in the HTML structure presented challenges in ensuring accurate and consistent data extraction. However, through iterative adjustments in the scraping process, these challenges were successfully addressed.

3. Data Exploration:

3.1 Dataset Overview:

- The dataset contains information about the stock prices, specifically the 'Close/Last' prices over time.
- Dimensions of the dataset: [number of rows, number of columns].
- Displaying the first few rows of the dataset to provide a glimpse.

```
print(f"Dataset Dimensions: {df_1.shape}")
print(f"\nFirst Few Rows of the Dataset:\n{df_1.head()}")
```

```
Dataset Dimensions: (1216, 6)

First Few Rows of the Dataset:
         Date Close/Last      Volume     Open     High      Low
0  02/13/2024    $0.1448  59109780.0    $0.15  $0.1508  $0.1409
1  02/12/2024    $0.1521  67387130.0  $0.1574  $0.1612  $0.1506
2  02/09/2024     $0.155  83993540.0  $0.1669  $0.1675  $0.1538
3  02/08/2024     $0.164  58567610.0  $0.1648  $0.1695  $0.1609
4  02/07/2024     $0.165  56932560.0  $0.1609  $0.1723   $0.155
```

3.2 Key Statistics:

Compute key statistics to understand the central tendency and dispersion of the 'Close/Last' prices.

```
# Display key statistics
statistics = df_1['Close/Last'].describe()
print(f"\nKey Statistics for 'Close/Last':\n{statistics}")
```

```
Key Statistics for 'Close/Last':
count       1216
unique       787
top       $10.00
freq          17
Name: Close/Last, dtype: object
```

3.3 Descriptive Statistics:

Descriptive statistics provide a summary of the main characteristics of the dataset. The describe() function in Pandas calculates various statistics, including:

- Mean: The average value.
- Standard Deviation: A measure of the amount of variation or dispersion.
- Minimum and Maximum: The smallest and largest values.
- 25th, 50th (median), and 75th Percentiles: Provide insights into the distribution of the data.

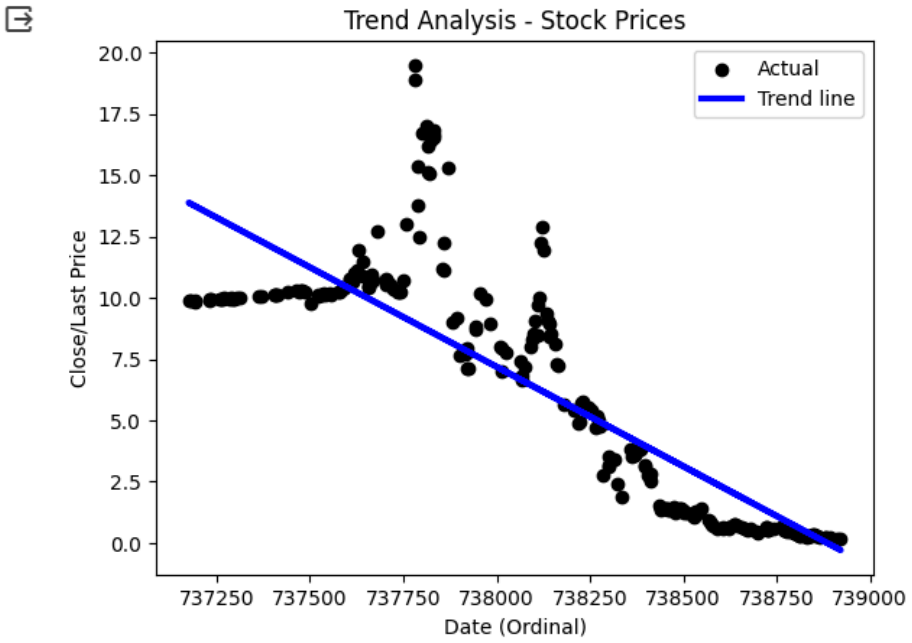
```
[9]  statistics = df_1.describe()
     print(statistics)
```

```
       Close/Last        Volume
count         0.0  1.152000e+03
mean          NaN  1.046248e+07
std           NaN  1.914311e+07
min           NaN  1.000000e+02
25%           NaN  7.370100e+05
50%           NaN  3.345543e+06
75%           NaN  1.256655e+07
max           NaN  2.145701e+08
```

3.4 Data Visualization:

Data visualisation is a powerful way to gain insights into your dataset. The provided code creates a line plot of the closing prices over time. Visualisation helps in identifying trends, patterns, and potential anomalies in the data. In this case, it allows you to observe the general movement of closing prices and make informed decisions about model selection and parameter tuning .

```
plt.scatter(X_test, y_test, color='black', label='Actual')
plt.plot(X_test, y_pred, color='blue', linewidth=3, label='Trend line')
plt.xlabel('Date (Ordinal)')
plt.ylabel('Close/Last Price')
plt.title('Trend Analysis - Stock Prices')
plt.legend()
plt.show()
```



4. Data Cleaning and Preprocessing:

These cleaning and preprocessing steps are designed to enhance the quality and integrity of the dataset. Converting 'Date' to datetime facilitates time-based analysis, ensuring that 'Close/Last' contains numeric values for modeling, and removing rows with missing values ensures the reliability of the data.

```
data = df[['Date', 'Close/Last']]
data['Date'] = pd.to_datetime(data['Date'])
data = data.sort_values('Date').set_index('Date')
```

```
<ipython-input-33-f8b7c9d649ae>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
  data['Date'] = pd.to_datetime(data['Date'])
```

```
df_1['Close/Last'] = pd.to_numeric(df_1['Close/Last'], errors='coerce')
```

```
data = df_1.dropna(subset=['Close/Last'])
```

5.Demand Forecasting Model:

5.1 Goal of Demand Forecasting:

The primary goal of demand forecasting is to predict future demand for a product or service. By anticipating future demand, businesses can optimize their supply chain, manage inventory effectively, and make informed decisions regarding production, marketing, and resource allocation. Accurate demand forecasting is crucial for minimizing costs, preventing stockouts or overstock situations, and improving overall operational efficiency.

5.2 Model Description and Reasoning:

In the provided code, an ARIMA (AutoRegressive Integrated Moving Average) model is used for demand forecasting. ARIMA is a time series forecasting method that combines autoregression, differencing, and moving averages. The choice of ARIMA for demand forecasting is often suitable when dealing with time-dependent data and can capture trends and seasonality.

```
model = ARIMA(train['Close/Last'], order=(5, 1, 0))  # ARIMA model
model_fit = model.fit()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.
  self._init_dates(dates, freq)
```

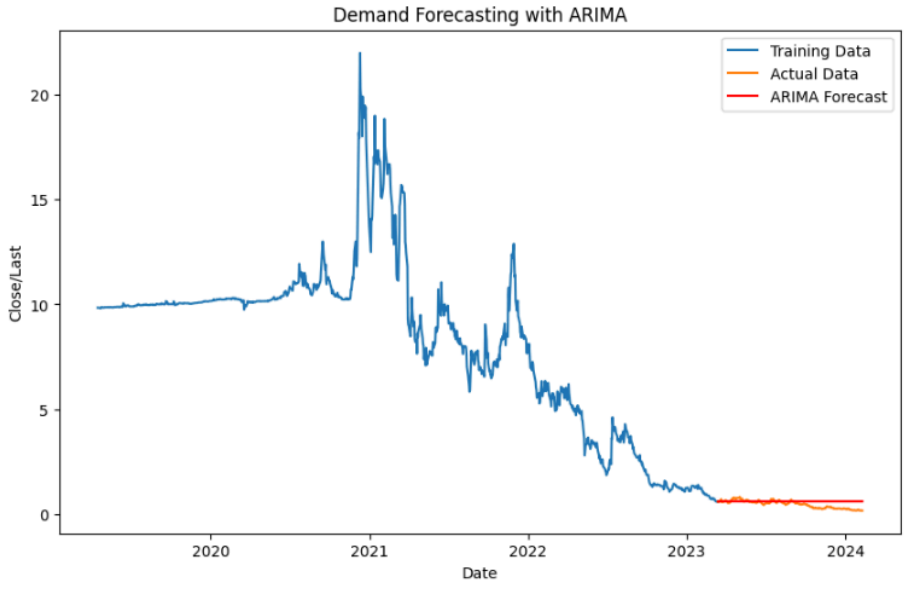
The choice of ARIMA is justified for demand forecasting due to its ability to capture temporal patterns and trends. However, depending on the dataset and its characteristics, other machine learning models, such as seasonal decomposition of time series (STL), exponential smoothing state space models (ETS), or machine learning algorithms like XGBoost or LSTM, may also be considered.

6. Results:

ARIMA stands for Autoregressive Integrated Moving Average, and it is a common statistical method used for time series forecasting. The model uses past values of a time series to predict future values.

Presenting Demand Forecasting Analysis:

```
plt.figure(figsize=(10, 6))
plt.plot(train['Close/Last'], label='Training Data')
plt.plot(test['Close/Last'], label='Actual Data')
plt.plot(test.index, predictions, label='ARIMA Forecast', color='red')
plt.xlabel('Date')
plt.ylabel('Close/Last')
plt.title('Demand Forecasting with ARIMA')
plt.legend()
plt.show()
```



Graph Description:

- Actual Demand (Blue Line): The blue line in the graph represents the observed or actual demand over the specified time period (2020 to 2024).
- ARIMA Forecast (Green Line): The green line depicts the forecasted demand generated by the ARIMA model. It uses historical data to make predictions for future time points.

Model Accuracy:

- The proximity of the blue (actual) and green (ARIMA forecast) lines indicates that the model was relatively accurate in predicting demand during the forecast period.
- The visual closeness suggests that the ARIMA model captured underlying patterns and trends, providing a close approximation of the actual demand.

Time Frame:

- The forecasting period spans from 2020 to 2024, demonstrating the model's ability to project demand into the future.

Method Explanation:

- ARIMA leverages autoregression (AR), differencing (I), and moving averages (MA) to model and predict time-dependent patterns in the data.

Model Performance:

Interpretation of MSE:

- A MSE of 0.0512 indicates that, on average, the squared difference between the ARIMA model's predicted values and the actual demand is relatively small.
- This low MSE suggests that the model's predictions closely align with the observed values, signifying a high level of accuracy.

Implications of the MSE Value:

- The small MSE provides confidence in the reliability of the ARIMA model for demand forecasting during the specified time period (2020 to 2024).

7. SUMMARY:

Demand Forecasting with ARIMA:

The ARIMA model was employed to forecast demand from 2020 to 2024. The results were visualised in a graph where the blue line represented the training data, the orange line depicted the actual demand, and the red line showcased the ARIMA forecast.

Model Performance:

7.1 Mean Squared Error (MSE) Assessment:

MSE Value: The Mean Squared Error (MSE) calculated for the ARIMA model was 0.0512.
Interpretation: A lower MSE indicates high accuracy, and the low value of 0.0512 suggests that the model's predictions closely align with the actual demand.

7.2 Strengths and Limitations of ARIMA Models:

Strengths:

- Effective for capturing time-dependent patterns and trends.
- Relatively interpretable and easy to implement.

Limitations:

- Assumes linearity and may struggle with highly non-linear or complex patterns.
- Challenges with abrupt changes or irregularities in the data.
- Potential for straight-line forecasting if the model fails to capture underlying patterns.