Signals and System(s)

Financial Analysis using Signal Processing Tools|

Made By-:

- Mukul Goel(220108037)
- Archie Jaipuria(220108009)
- Harnoor Kaur(220102125)
- Harpal Nirvedh Santosh(220108070)
- Achyut Chikhaliya(220108069)

INDEX-:

❖ INTRODUCTION

In this project, the focus is on using two important analytical tools, ARIMA (Autoregressive Integrated Moving Average) and Fourier Transforms, as key elements in a deep learning model for predicting financial outcomes.

ARIMA is a well-established method in the field of time-series analysis. It combines two key components:

- AutoRegression (AR), which considers the relationship between a current value and its past values, and moving average (MA), which accounts for the relationship between the current value and past forecast errors.
- The integrated (I) component reflects the number of differences needed to make the time series stationary. In essence, ARIMA models aim to capture patterns and dependencies in historical financial data, making it possible to project future values based on these patterns. It's a valuable tool for understanding trends and seasonality in financial time series.

➢ Why use Fourier?

Fourier Transforms are mathematical techniques that decompose time-series data into different frequency components. This is particularly useful when analyzing financial data, as it helps identify periodic patterns and hidden oscillations that might not be immediately apparent through traditional statistical analysis. By breaking down the time series into its constituent frequencies, analysts can gain insights into cyclic behaviors, which is vital in the financial world, where various assets and markets exhibit distinct periodic characteristics.

By incorporating these powerful techniques, along with popular technical indicators, into a deep learning model, we can enhance our ability to make accurate financial predictions. Deep learning models, which are a subset of machine learning, can learn complex relationships and patterns in the data, thus potentially improving forecasting accuracy. This approach provides an edge in the ever evolving and highly competitive field of finance, where making informed predictions is crucial for making investment decisions and managing risk effectively. The synergy of ARIMA, Fourier Transforms, and deep learning creates a robust analytical framework for financial prediction, helping analysts and traders navigate the dynamic and complex financial landscape.

❖ Downloading financial data:

All our data was downloaded from Yahoo Finance. Yahoo Finance provides historical stock data, which is a valuable resource for investors, traders, and analysts who want to analyze past stock prices and trends. Users can specify a date range to retrieve historical stock data for a specific period. Common options include daily, weekly, or monthly data. The opening price, closing price, high and low prices for each day or time period, as well as trading volume are available. Yahoo Finance allows users to download historical stock data in a downloadable format, such as CSV (Comma-Separated Values) or Excel, making it convenient for further analysis in other tools or software.

❖ Preprocessing:

We preprocessed the data by cleaning and organizing it into pandas dataframe that contains only the date and closing prices. Preprocessing financial data is essential for several reasons in data analysis and modeling within the finance domain:

1. Data Quality Assurance: Financial datasets often contain errors, missing values, or inconsistencies that can lead to inaccurate or unreliable analysis. Preprocessing helps identify and correct such issues, ensuring that the data is of high quality and suitable for analysis.
2. Outlier Detection: Financial data is susceptible to outliers (extreme values), which can distort analysis results. Preprocessing involves detecting and handling outliers to prevent them from unduly influencing the analysis or model.
3. Missing Data Handling: Missing data points are common in financial datasets due to holidays, trading suspensions, or data collection issues. Preprocessing techniques such as interpolation or imputation are used to address missing values and maintain data integrity.
4. Data Transformation: Data often requires transformation to be suitable for specific analysis or modeling techniques. For instance, returns may need to be calculated from price data, or time-series data may need to be differenced to make it stationary for time-series analysis.
5. Model Preparation: For predictive modeling in finance, preprocessing is essential to prepare the data for training and testing machine learning models. This involves data splitting into training and testing sets and ensuring the data is in a suitable format for the chosen model.
6. Dimension Reduction: In some cases, financial datasets can be large and complex. Preprocessing may involve dimension reduction techniques to extract relevant features or variables, simplifying subsequent analysis.

In summary, preprocessing financial data is necessary to ensure data quality, consistency, and suitability for analysis and modeling. It helps in cleaning, transforming, and organizing data, making it ready for meaningful insights, investment decisions, risk assessment, and financial predictions.

❖ ARIMA:

An AutoRegressive Integrated Moving Average (ARIMA) is a powerful statistical analysis model used for understanding time series data and making predictions about future trends within that data. It is a combination of three key components:

1. AutoRegressive (AR): The "autoregressive" part of ARIMA refers to the model's ability to predict future values based on past values within the same time series. It assumes that the future behavior of the data is influenced by its own past behavior. For example, an ARIMA model for stock price prediction might consider the stock's past price movements and trends to forecast future prices.
2. Integrated (I): The "integrated" component indicates that the time series data is differenced to achieve stationarity. Stationarity is a key concept in time series analysis, and differencing helps make the data's statistical properties constant over time. In simpler terms, it involves subtracting each data point from the previous one to remove trends or seasonality.
3. Moving Average (MA): The "moving average" part involves considering past forecast errors (the difference between predicted and actual values) to predict future values. It accounts for the influence of past errors on future data points. For example, in a financial context, it can be used to assess how past forecasting errors affect future earnings predictions for a company.

- What Is ARIMA Used for?

ARIMA is a method for forecasting or predicting future outcomes based on a historical time series. It is based on the statistical concept of serial correlation, where past data points influence future data points.

- How is ARIMA related to signal processing?

ARIMA (AutoRegressive Integrated Moving Average) is a statistical method used for time series forecasting and modeling. Although ARIMA is not a signal processing technique, it can be applied to analyze and forecast time series data, including signals.
Here's how ARIMA is related to signal processing:

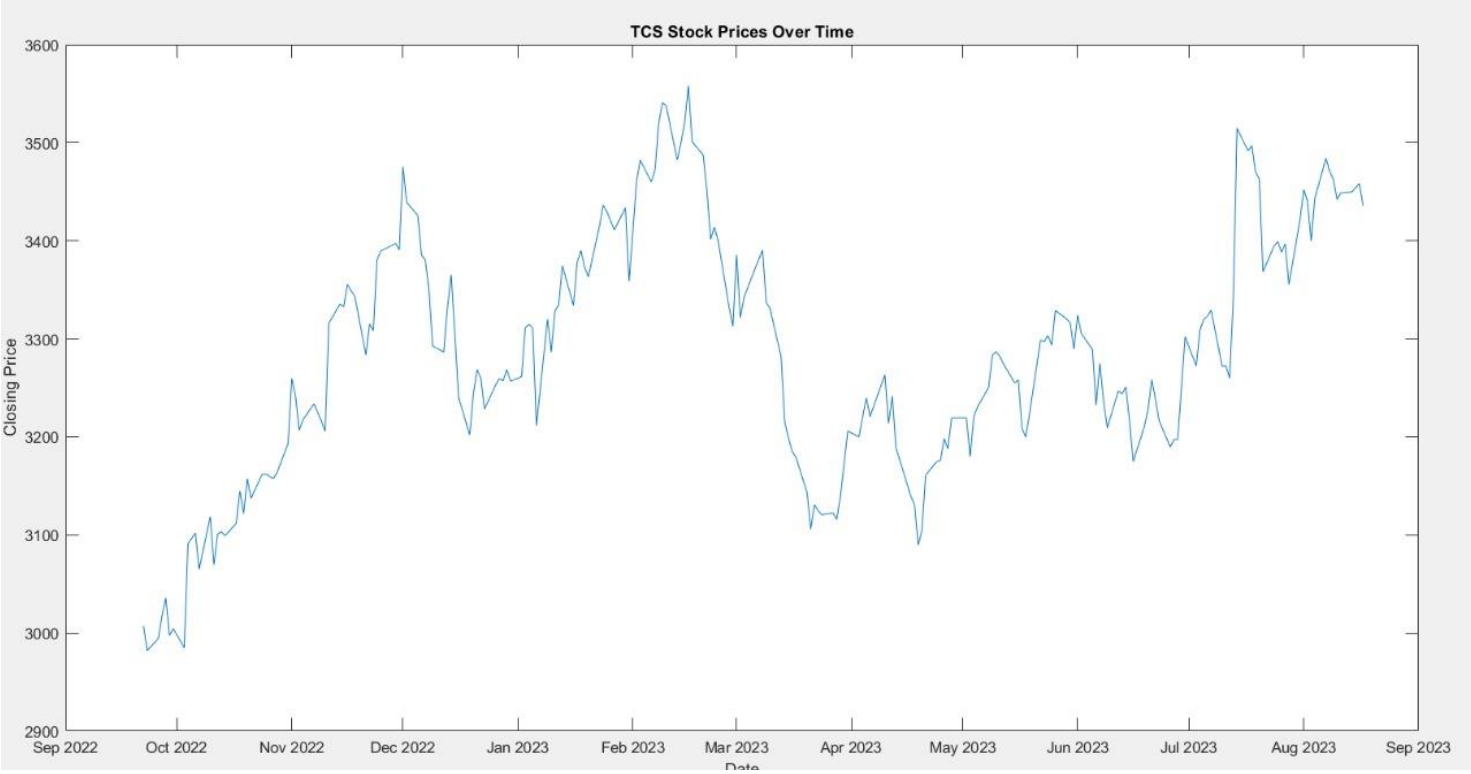
1. Signal Analysis: In signal processing, you often deal with time-domain signals, such as audio, video, sensor data, and more. ARIMA can be applied to analyze these signals by modeling their underlying time series components, such as trends, seasonality, and noise. By decomposing signals into these components, you can gain insights into the underlying patterns and characteristics of the signal.
2. Noise Reduction: ARIMA can be used to filter out noise or unwanted components from a signal. By modeling the noise component of a signal using ARIMA, you can remove it and extract the desired signal components, which can be useful in denoising applications.
3. Forecasting and Prediction: Signal processing often involves predicting future values of a signal. ARIMA models are excellent for time series forecasting and can be used to make predictions about future signal values. This is particularly useful in applications like speech recognition, where predicting the next audio sample is crucial.
4. System Identification: In some signal processing applications, it's necessary to identify the underlying system dynamics that generate a signal. ARIMA models can help in system identification by providing a mathematical description of the signal's behavior. This information can be valuable in control systems and system analysis.
5. Anomaly Detection: ARIMA models can be used to detect anomalies or unusual patterns in signals. By comparing the actual values to the ARIMA model's predictions, you can identify deviations that may indicate anomalies in the signal.

❖ How to build an ARIMA Model?

To begin building an ARIMA model for an investment, you download as much of the price data as you can. Once you've identified the trends for the data, you identify the lowest order of differencing (d) by observing the autocorrelations. If the lag-1 autocorrelation is zero or negative, the series is already differenced. You may need to difference the series more if the lag-1 is higher than zero.
Next, determine the order of regression (p) and order of moving average (q) by comparing autocorrelations and partial autocorrelations. Once you have the information you need, you can choose the model you'll use.

❖ ARIMA Implementation-:

To get an idea about ARIMA and AR models, we decided to implement them in MATLAB.

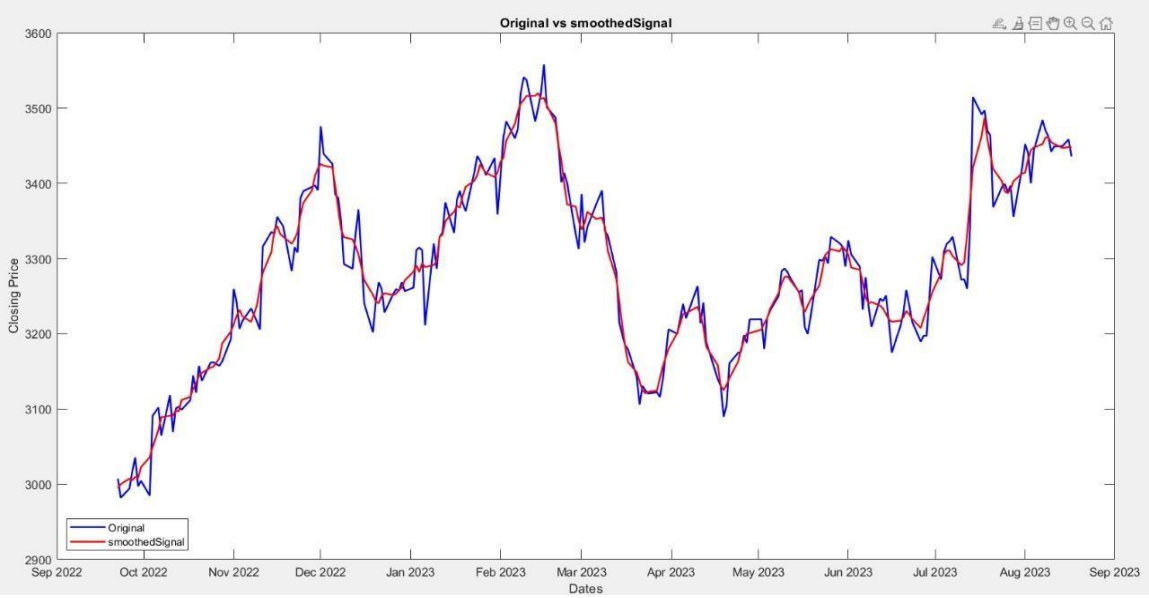


- Code for loading the Data and Plotting:

```
% Read the data from a CSV file into
a table
data = readtable('TCS.NS.csv');
data.Date = datetime(data.Date);
figure;
plot(data.Date, data.Close);
title('TCS Stock Prices Over Time');
xlabel('Date');
ylabel('Closing Price');
```

- Smoothing the Plot -:

  Smoothing a signal refers to the process of reducing noise and variability in the data, making the underlying trends or patterns more apparent. It involves applying mathematical techniques, such as moving averages or filtering algorithms, to create a less jagged and more continuous version of the signal. Smoothing is commonly used in data analysis to enhance visualization, reveal long-term trends, and improve the accuracy of subsequent analyses or predictions.



- Code for smoothing the plot:

```
% Smooth the signal using a moving average filter
windowSize = 5;
% Size of the moving average window
SmoothedSignal = movmean(signal,windowSize);
figure;
plot(data.Date,data.Close,'b','LineWidth',1.5);
hold on;
plot(data.Date,smoothedSignal,'r','LineWidth',1.5;
title('Original vs smoothedSignal');
xlabel('Dates');
ylabel('Closing Price');
legend({'Original','smoothedSignal'},'Location','southwest');
```

- The above code is meant to display a comparison between the original closing prices of a financial instrument (likely a stock) and a smoothed version of those prices using a red line.

❖ Data cleaning -:

Assumption: Consider the proficiency of the website that we fetch data from, there will not be simple mistakes such as typos.

- Stationary Time Series:

A Stationary time series maintains constant statistical properties like mean, variance, and autocorrelation over time, lacking trends or seasonality, simplifying analysis and modeling. Stationarity serves as a crucial assumption in techniques like ARIMA, ensuring reliable predictions by assuming consistent data patterns. This is vital because stationary processes are easier to analyze and model compared to their broader counterparts, simplifying investigations and predictions. The predictability of how these processes change makes them valuable, offering insights into real-world phenomena and enhancing the accuracy of predictions, forming a fundamental basis for reliable time series analysis. We embark upon a certain hypothesis with our project that is-

- Hypothesis: In the realm of statistics and research, a hypothesis is a precise, clear, and testable statement or prediction about the relationship between variables. It's a fundamental part of the scientific method, guiding researchers to design experiments and collect data to either support or refute the hypothesis. Hypotheses are often framed as null hypothesis (H0), which states that there is no significant effect or relationship, and alternative hypothesis (H1), which posits a specific effect or relationship between variables. Researchers use statistical tests to evaluate data and determine whether to accept or reject the null hypothesis, thereby drawing conclusions about the relationship being studied.

Unit Roots: It indicates non-stationary time series data with unpredictable, persistent patterns. Detecting unit roots is crucial, especially in econometrics, as they complicate accurate modeling. Techniques like differencing are used to transform non-stationary data into a more predictable form, enabling better analysis and forecasting.

- Augmented Dickey Fuller test (ADF Test)

  In statistics, an Augmented Dickey–Fuller test (ADF) tests the null hypothesis that a unit root is present in a time series sample. The alternative hypothesis is different depending on which version of the test is used but is usually stationarity or trend-stationarity. The augmented Dickey–Fuller (ADF) statistic, used in the test, is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence. The ADF test belongs to a category of tests called 'Unit Root Test'.

  ➢ Results from ADF test:

  H0: The time series is non-stationary
  p-value = 0.828396

  Thus, we can't reject the null hypothesis that the origin time series data is non-stationary. To satisfy the stationary assumption, we need to do further data preprocessing.

  - CODE FOR ADF TEST:

```
[h, pValue] = adftest(data.Close);
display(h);
display(pValue);
```

- Differencing:

  Differencing is a common technique in time series analysis used to stabilize the mean of a time series and make the series stationary. Stationary time series have constant statistical properties over time, making them easier to model and analyze. Differencing involves computing the differences between consecutive observations in a time series.

- Code for differencing:

```
differencedData = diff(data.Close);
(ADF Test ⇒ H1: The time series is stationary)
```

- The above code calculates the first difference of a time series.

- Finding parameter (p, q, r):
  ARIMA(p,d,q)

  i. Integrated/Mean Distributed Lag parameter(d): The minimum difference order to make the sequence stationary.
  ii. Autoregressive (AR) parameter(p): Indicates relationship with history value.
  iii. Moving average (MA) parameter(q): Indicates relationship with AR prediction error.

  * p,q will be selected by grid search using criteria of AIC/BIC.

By adjusting these parameters and fitting the model to the specific characteristics of the time series data, ARIMA models can capture and model various patterns, such as trends, seasonality, and cyclic behavior. This makes ARIMA a valuable tool for time series forecasting and analysis, helping analysts and researchers gain insights and make informed decisions based on historical data.

- Code for finding p, q, r:

```
%Loop over different combinations of p, d, and q values
for p = 1:3
for d = 1:2
for q =1:3
% Create an ARIMA model with the specified p, d, and q values
arimaModel = arima(p, d, q);
% Estimate the ARIMA model and obtain the fitting, log likelihood
[fit, ~, logL] = estimate(arimaModel, data);
aic = -2 * logL + 2 * (p + d + q + 1);
bic = -2 * logL + log(numel(data)) * (p + d + q + 1);
fprintf('ARIMA(%d,%d,%d) - AIC: %.2f, BIC: %.2f\n', p, d, q, aic, bic);
end
 end
end
```

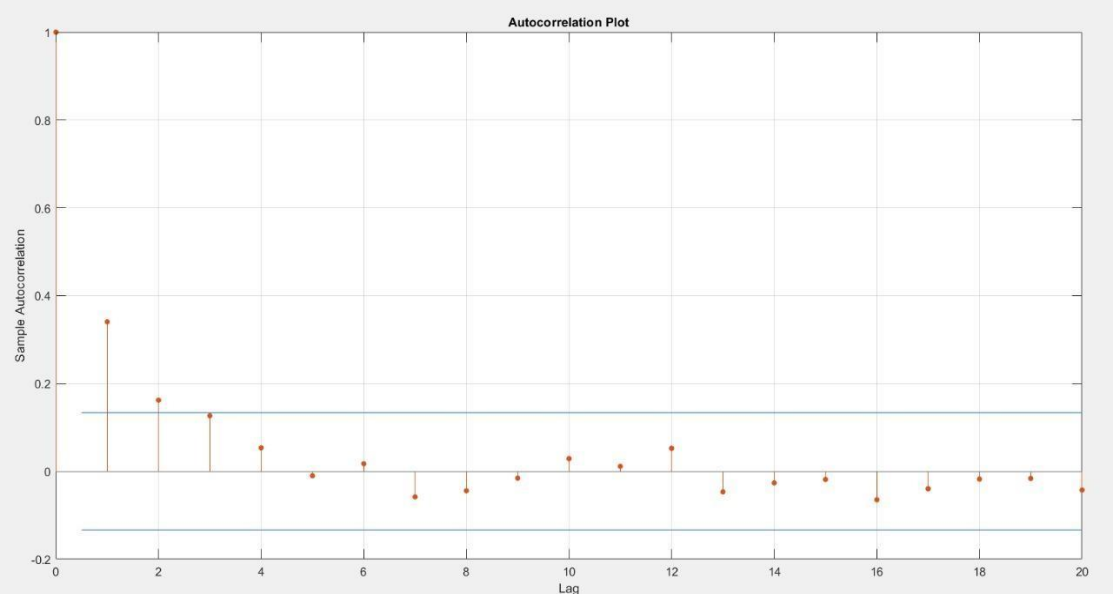
- The above code provides us with a MATLAB script that performs a grid search for ARIMA model selection. It tries different combinations of model orders (p, d, and q) and computes two commonly used information criteria, the AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion), to assess the goodness of fit for each combination.

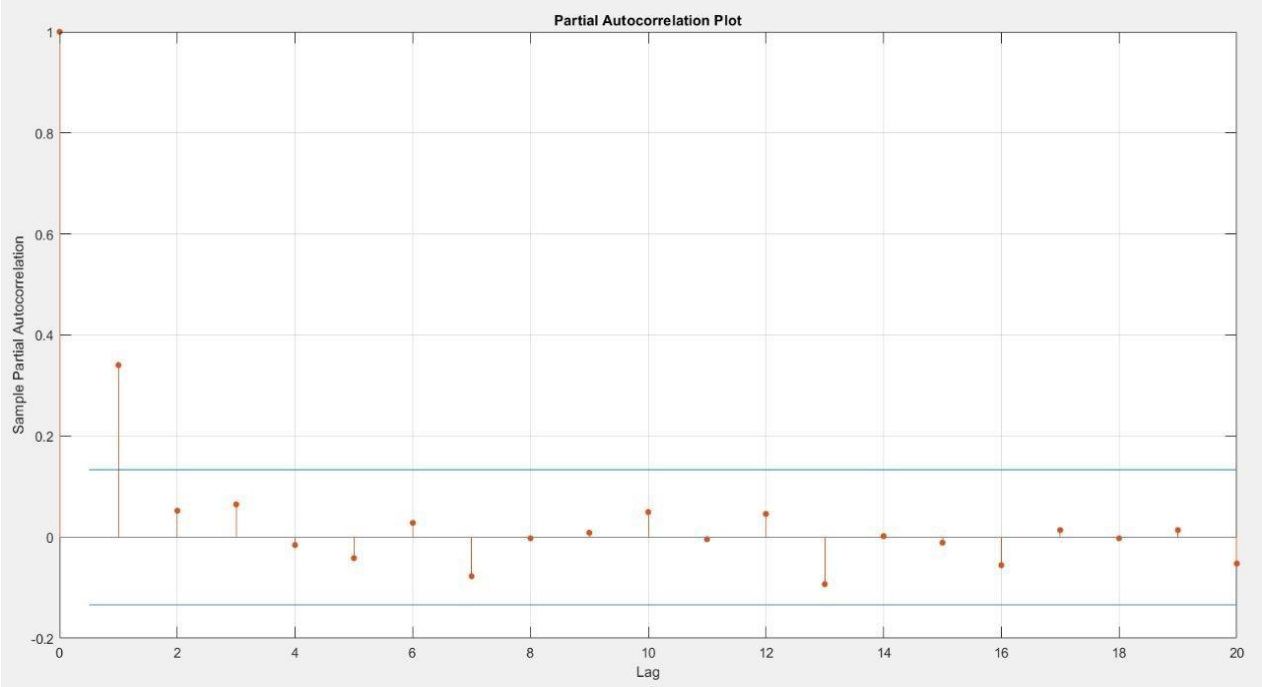
➢ We get Value p=1, q=1, d=1.

• Verifing Parameters:

- ACF: It measures the correlation between a time series and its lagged values. It helps identify the presence of seasonality or trend in the data. It is particularly useful in identifying the order of Moving Average(q).
- PACF: It measures the correlation between a data point and its lagged values, removing the influence of shorter lagged observations. It is particularly useful in identifying the order of an AR(p) process in ARIMA modeling.

• PACF/ACF Plot for initial data-:



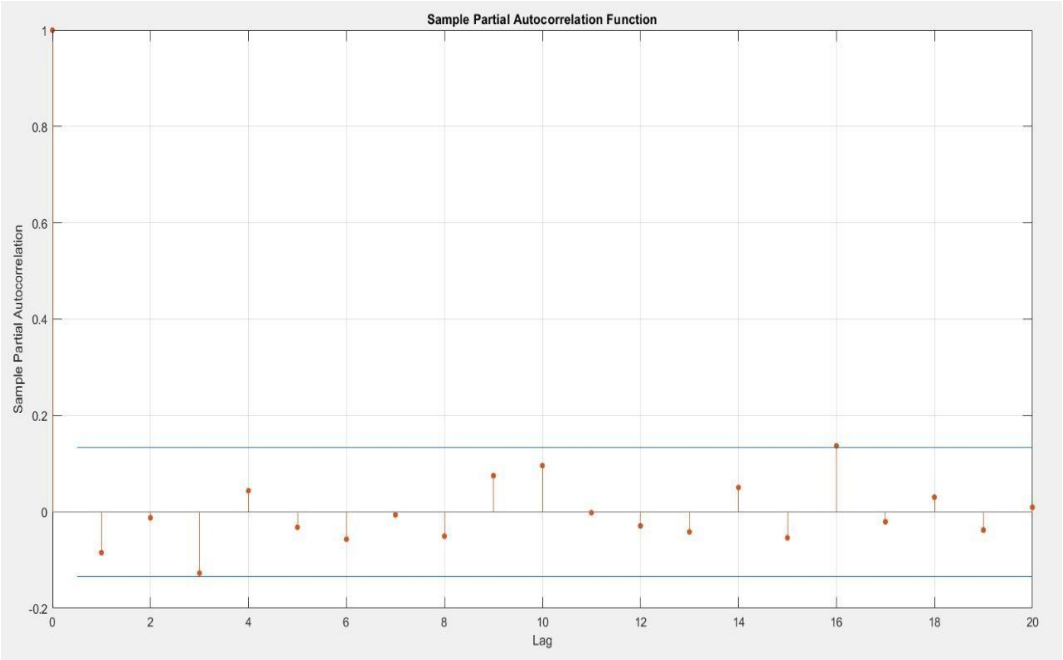


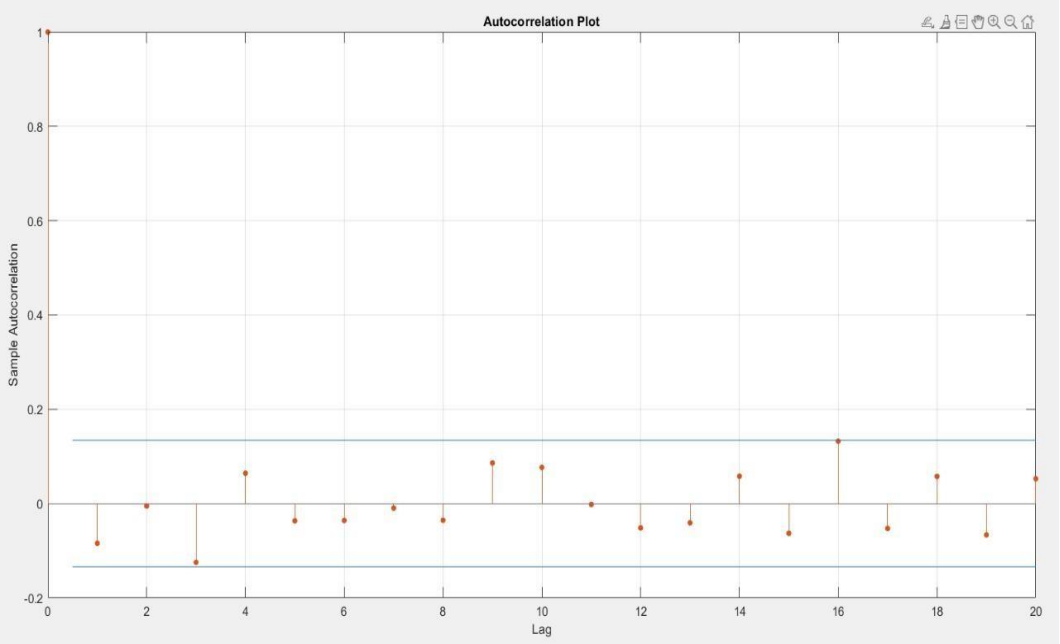
- Code for ACF/PACF:

```
figure;
autocorr(data);
title('Autocorrelation Plot');
figure;
parcorr(data);
title('Partial Autocorrelation Plot')
```

- The above code figure generates two separate figures in MATLAB. The first figure displays the autocorrelation plot, showing the correlation between data points and their lagged values. The second figure presents the partial autocorrelation plot, revealing correlations while removing intervening values' effects, aiding in determining the order of autoregressive models.

- ACF/PACF plot for differentiated plot:





- Code for ACF/ PACF for differenced data:

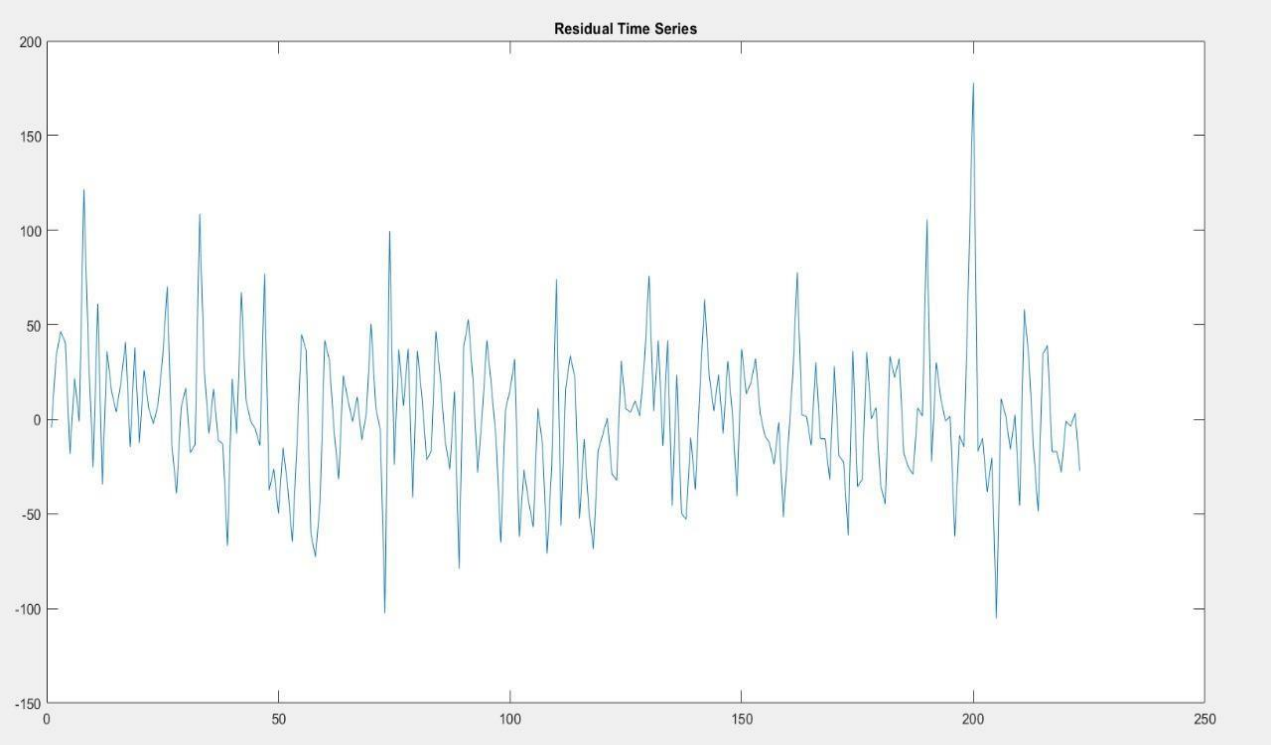
```
differencedData = diff(data.Close); figure;
autocorr(differencedData); title('Autocorrelation Plot');
figure; parcorr(differencedData);
title('Partial Autocorrelation Plot')
```

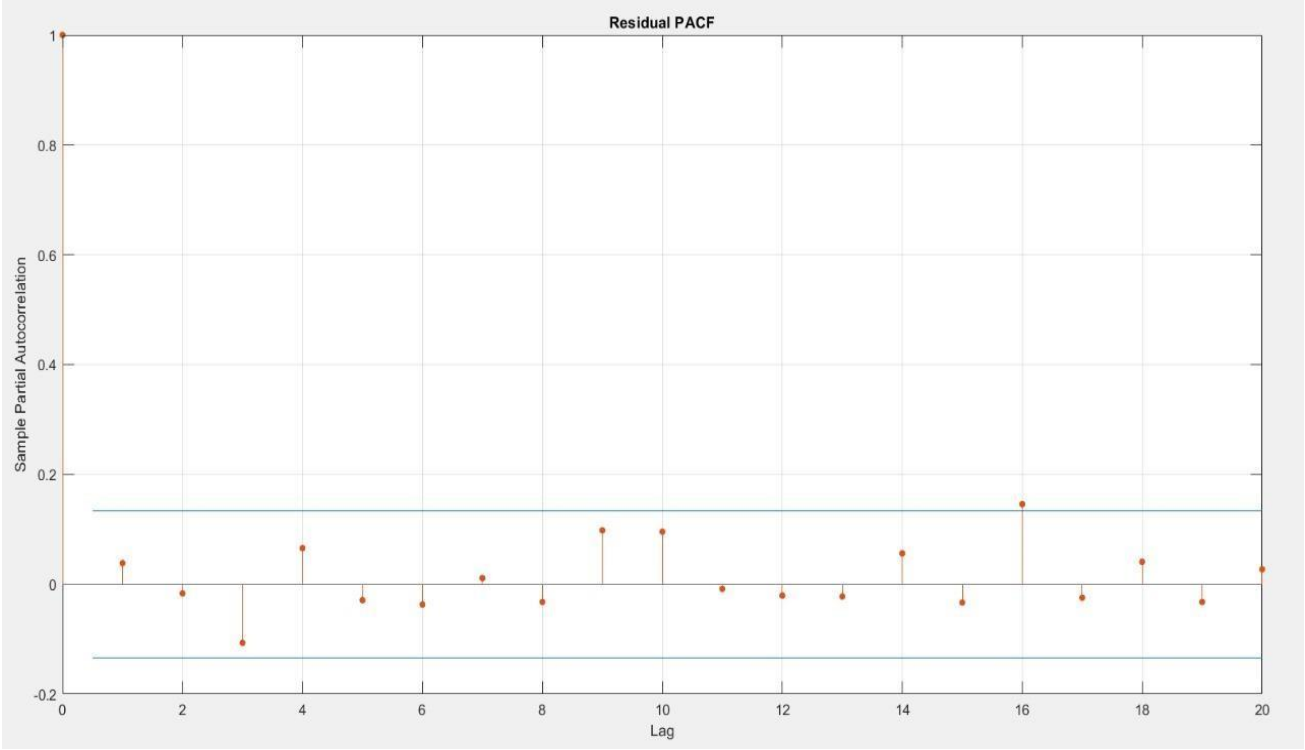
- The code calculates the first-order differences of the 'Close' prices, creating differenced Data. It then generates autocorrelation and partial autocorrelation plots for the differenced data, aiding in identifying lagged correlations and determining time series model parameters.

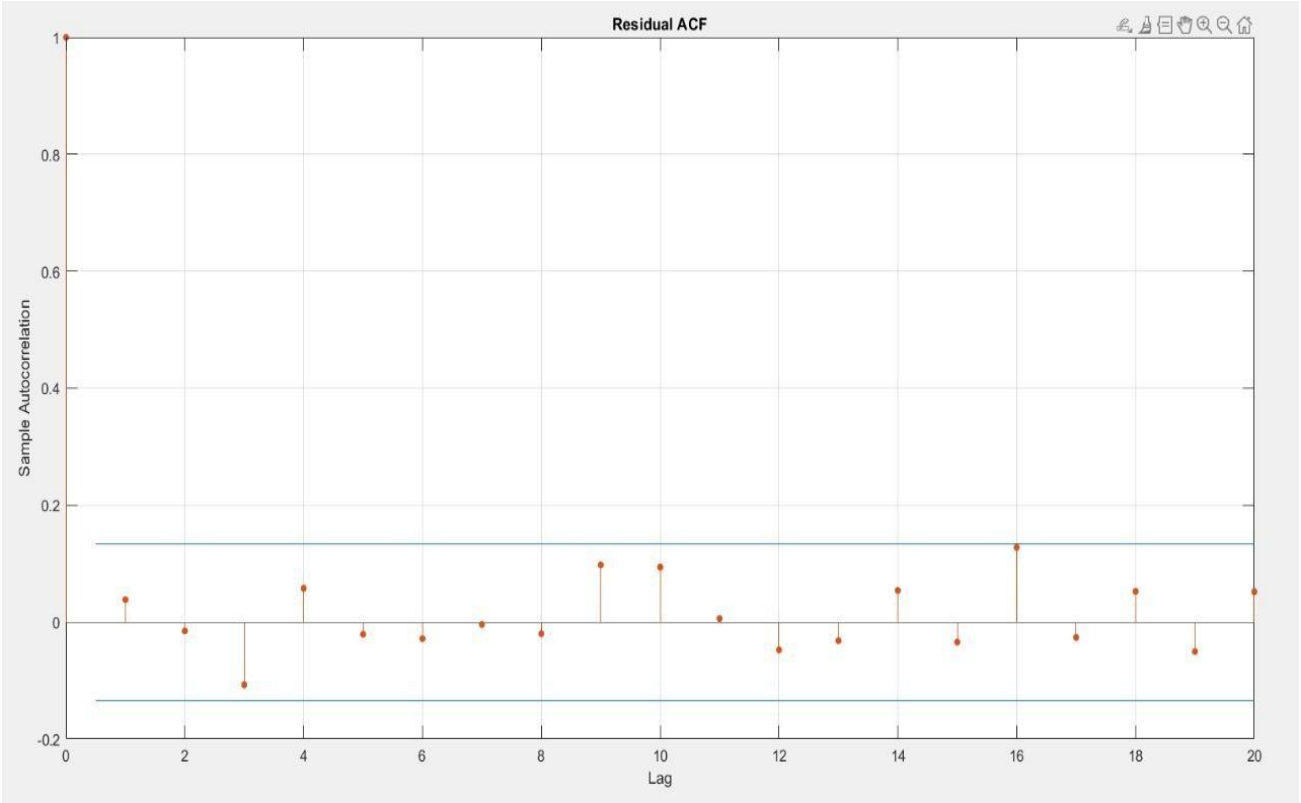
❖ Dealing with Residual-:

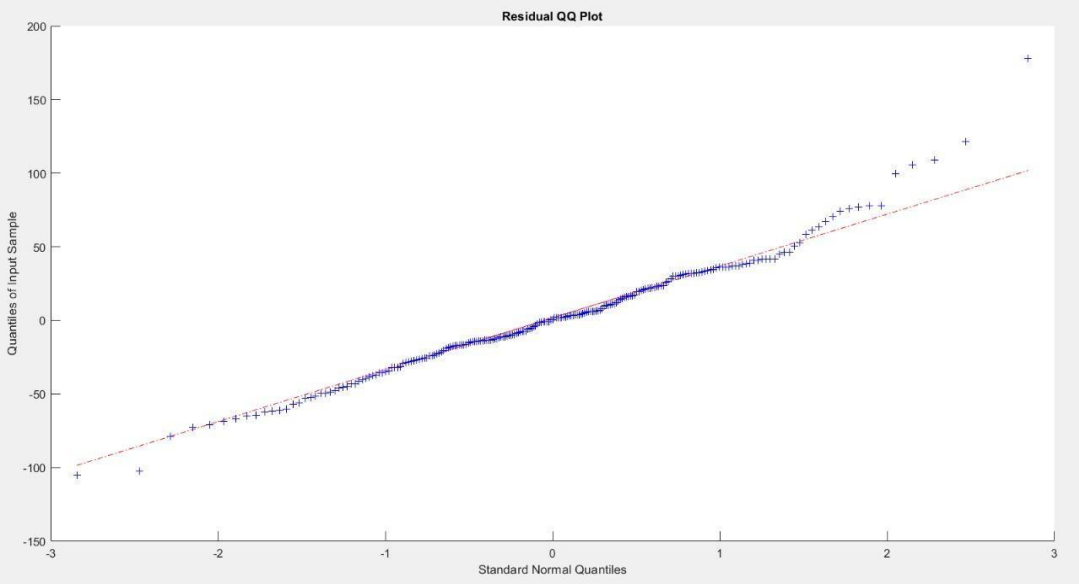
In time series forecasting, residuals refer to the difference between the actual value of a data point and the value predicted by a model. They are also known as the errors of the model. These residuals can be used to evaluate the performance of a model and to identify any patterns or trends in the data that the model may have missed. They can also be used to detect outliers or anomalies in the data.

- Residual Time Series Plot: A visual representation of the residuals over time.
- Residual ACF and PACF: Autocorrelation and partial autocorrelation of the residuals help identify any remaining patterns in the model.
- Residual QQ Plot: A quantile-quantile plot comparing the distribution of residuals against a theoretical normal distribution. It helps assess the normality of residuals.




Residual PACF
Sample Partial Autocorrelation
Lag
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14
16
18
20


Residual ACF
Sample Autocorrelation
Lag
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14
16
18
20



- Code for Plotting Residuals:

```
% Differencing the data
differencedData = diff(data.Close);
% Estimating the ARIMA model using differenced data
arimaModel = arima(p, d, q);
arimaFit = estimate(arimaModel, differencedData);
residuals = infer(arimaFit, differencedData);
figure;
autocorr(residuals);
title('Residual ACF');
figure;
parcorr(residuals);
title('Residual PACF');
figure;
plot(residuals);
title('Residual Time Series');
figure;
qqplot(residuals);
title('Residual QQPlot');
```

- The above code performs differencing on 'Close' prices, fits an ARIMA (p, d, q) model, and analyzes residuals. It generates autocorrelation, partial autocorrelation, time series, and QQ plots, aiding in evaluating the model's fit and identifying potential issues.

❖ Forecasting using ARIMA Model:

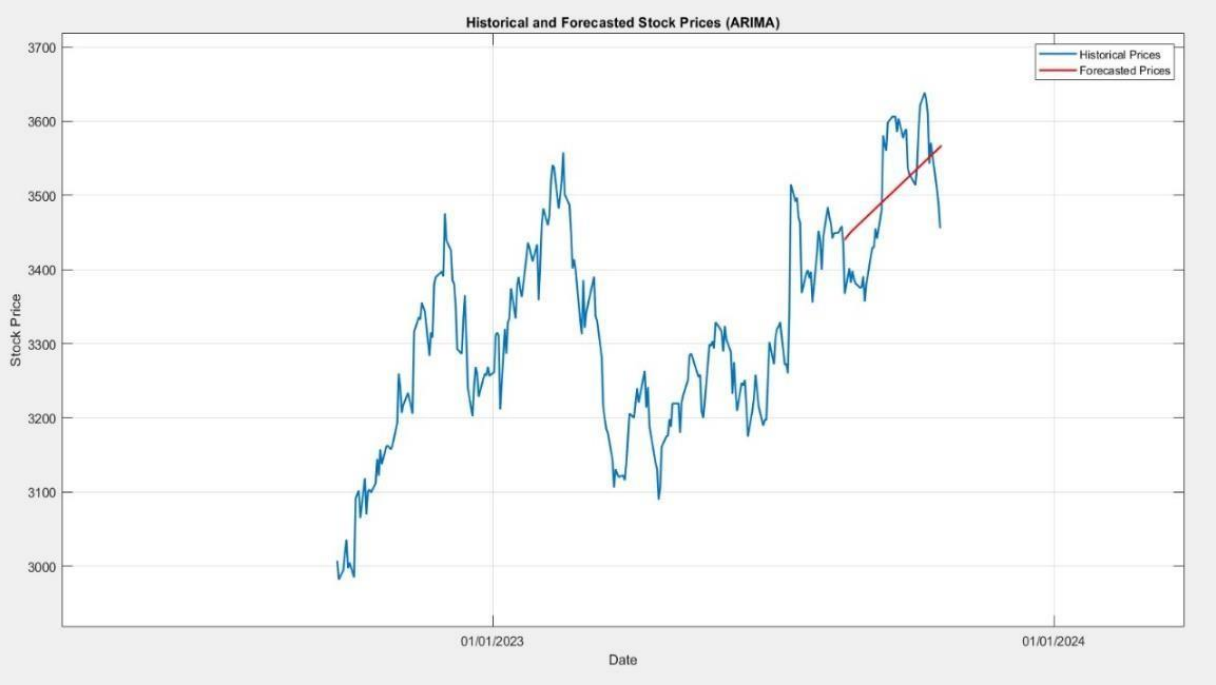
ARIMA (AutoRegressive Integrated Moving Average) is a powerful time series forecasting model that combines autoregressive (AR), differencing(I) and moving average (MA) components. ARIMA is suitable for non-stationary time series data, where the mean and variance change over time.

- **AutoRegressive (AR) Component:** ARIMA models capture the relationship between a value and its past values in a linear regression manner, denoted by the "p" parameter. It considers correlations between the current value and its lagged (past) values.
- **Integrated (I) Component:** The "I" in ARIMA represents differencing, where the time series data is transformed to achieve stationarity (constant mean and variance) by differencing the series. The differencing order is denoted by the "d" parameter.
- **Moving Average (MA) Component:** The MA component represents the correlation between a time series value and residual errors from a moving average model applied to lagged values of the series. It is denoted by the "q" parameter.

Mathematical Expression for ARIMA Model

$$yt = c + \phi 1y(t-1) + \phi 2y(t-2) + \ldots + \phi py(t-p) + \varepsilon t + \theta 1\varepsilon(t-1) + \theta 2\varepsilon(t-2) + \ldots + \theta q\varepsilon(t-q) + \varepsilon t$$

Plot Of ARIMA Model:



- ## Code for forecasting using ARIMA Model:

```
data2 = readtable('TCS.NS.csv');
dates2 = datenum(data2.Date);
prices2 = data2.Close;
data = readtable('TCS.NS2.csv');
dates = datenum(data.Date);
prices = data.Close;
 p = 1;
d = 1;
q = 1;
% Create an ARIMA model object with the specified parameters
arimaModel = arima(p, d, q);
% Estimate the ARIMA model using the prices data
estimatedModel = estimate(arimaModel, prices);
% Define the number of periods for which to forecast future prices
```

```
numForecastPeriods = 64;
% Forecast future prices using the fitted ARIMA model
forecast = forecast(estimatedModel, numForecastPeriods, 'Y0', prices);
figure;
plot(dates2, prices2, 'LineWidth', 1.5);
hold on;
forecastDates = dates(end) + 1:dates(end) + numForecastPeriods;
plot(forecastDates, forecast, 'r', 'LineWidth', 1.5);
% Convert serial dates to readable dates on the x-axis
datetick('x', 'mm/dd/yyyy');
xlabel('Date');
ylabel('Stock Price');
title('Historical and Forecasted Stock Prices (ARIMA)');
legend('Historical Prices', 'Forecasted Prices');
grid on;
```

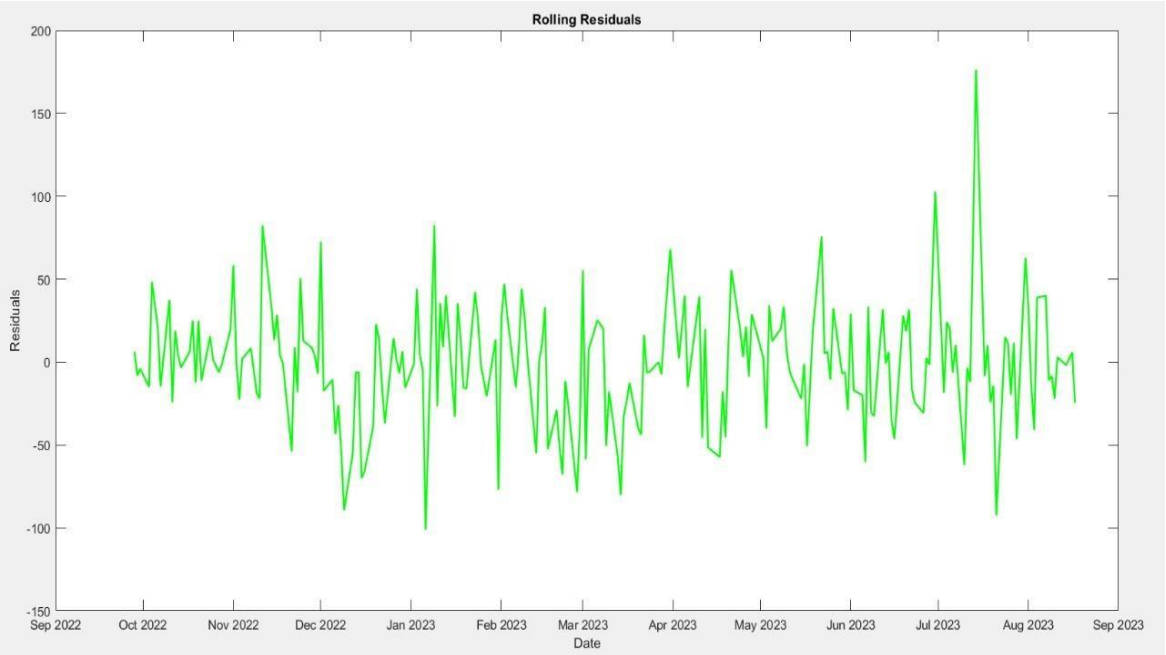
- The above MATLAB code demonstrates a time series forecasting task using an ARIMA (AutoRegressive Integrated Moving Average) model.

We could not accurately implement the ARIMA model. Our model just joins the initial and final points on our dataset and thus is not accurate at all. Hence, we used rolling forecast ARIMA model.

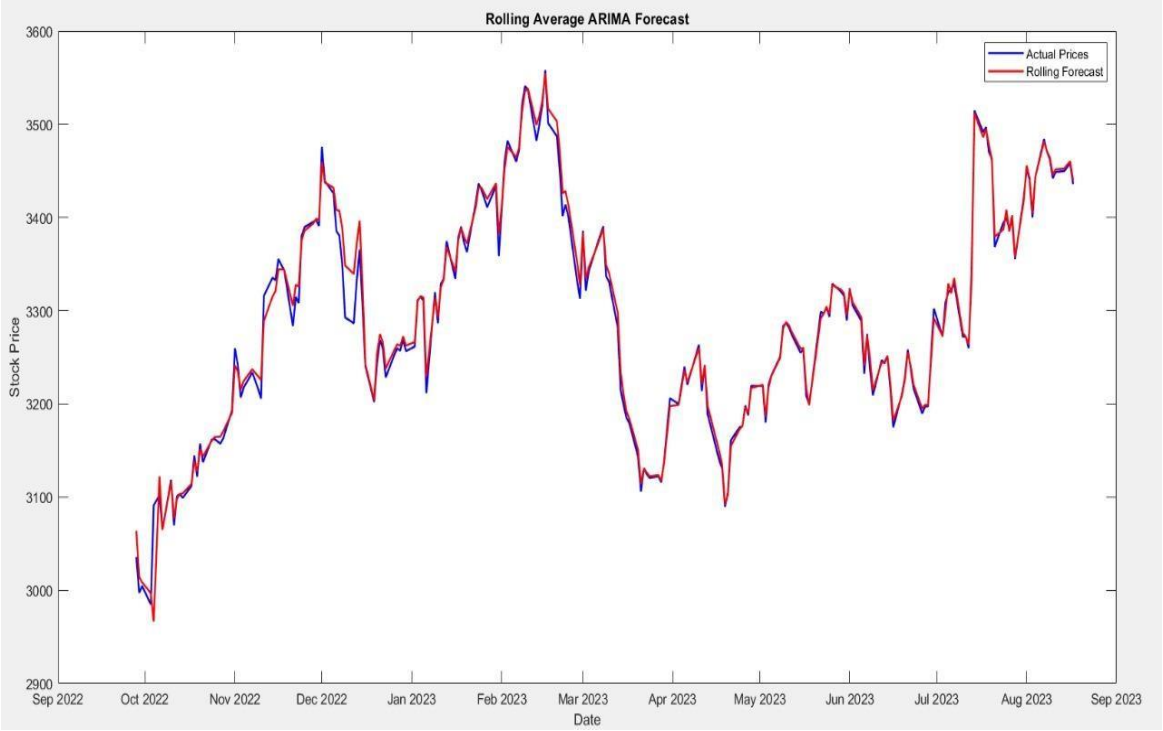
• Rolling Forecast ARIMA Model:

- Rolling forecasting involves updating the model at regular intervals as new data becomes available, allowing for more accurate and up-to-date predictions.
- Rolling Update: As new data becomes available, the model is updated by re-fitting the ARIMA model with the most recent data. This allows the model to adapt to any changes in the underlying patterns or trends in the time series.
- Prediction Refinement: After updating the model, new forecasts are generated for the upcoming period, providing more accurate predictions based on the most recent data.

- Plot of Rolling Reidues:



- Plot for Rolling Average ARIMA Forecast:



- Code for Rolling Average ARIMA Forecast:

```
data2 = readtable('TCS.NS.csv');
dates2 = datetime(data2.Date);
prices2 = data2.Close;
data = readtable('TCS2.NS.csv');
dates = datetime(data.Date);
prices = data.Close;
windowSize = 5; % You can adjust this as needed
% Initialize arrays to store forecasted values and residuals
rollingForecast = [];
rollingResiduals = [];
for i = windowSize:length(prices)
% Select a subset of data for each iteration
subsetData = data(1:i, :);
% Fit ARIMA model for the subset data
%valures of p, d, q are 1, 1, 1
arimaModel = arima(p, d, q);
arimaFit = estimate(arimaModel, subsetData.Close);
% Forecast the next value using the fitted ARIMA model
[forecastedValue, forecastedError] = forecast(arimaFit, 1, 'Y0',
subsetData.Close);
% Append the forecasted value to the rolling forecast array
rollingForecast = [rollingForecast; forecastedValue];

% Calculate residuals and append to the rolling residuals array
currentResiduals = infer(arimaFit, subsetData.Close);
rollingResiduals = [rollingResiduals; currentResiduals(end)];
end
% Plot the rolling forecast and residuals
figure;
subplot(2,1,1);
plot(dates(windowSize:end), prices(windowSize:end), 'b', 'LineWidth', 1.5);
hold on;
plot(dates(windowSize:end), rollingForecast, 'r', 'LineWidth', 1.5);
xlabel('Date');
ylabel('Stock Price');
title('Rolling Average ARIMA Forecast');
legend('Actual Prices', 'Rolling Forecast');
subplot(2,1,2);
plot(dates(windowSize:end), rollingResiduals, 'g', 'LineWidth', 1.5);
```

```
xlabel('Date');
ylabel('Residuals');
title('Rolling Residuals');
```

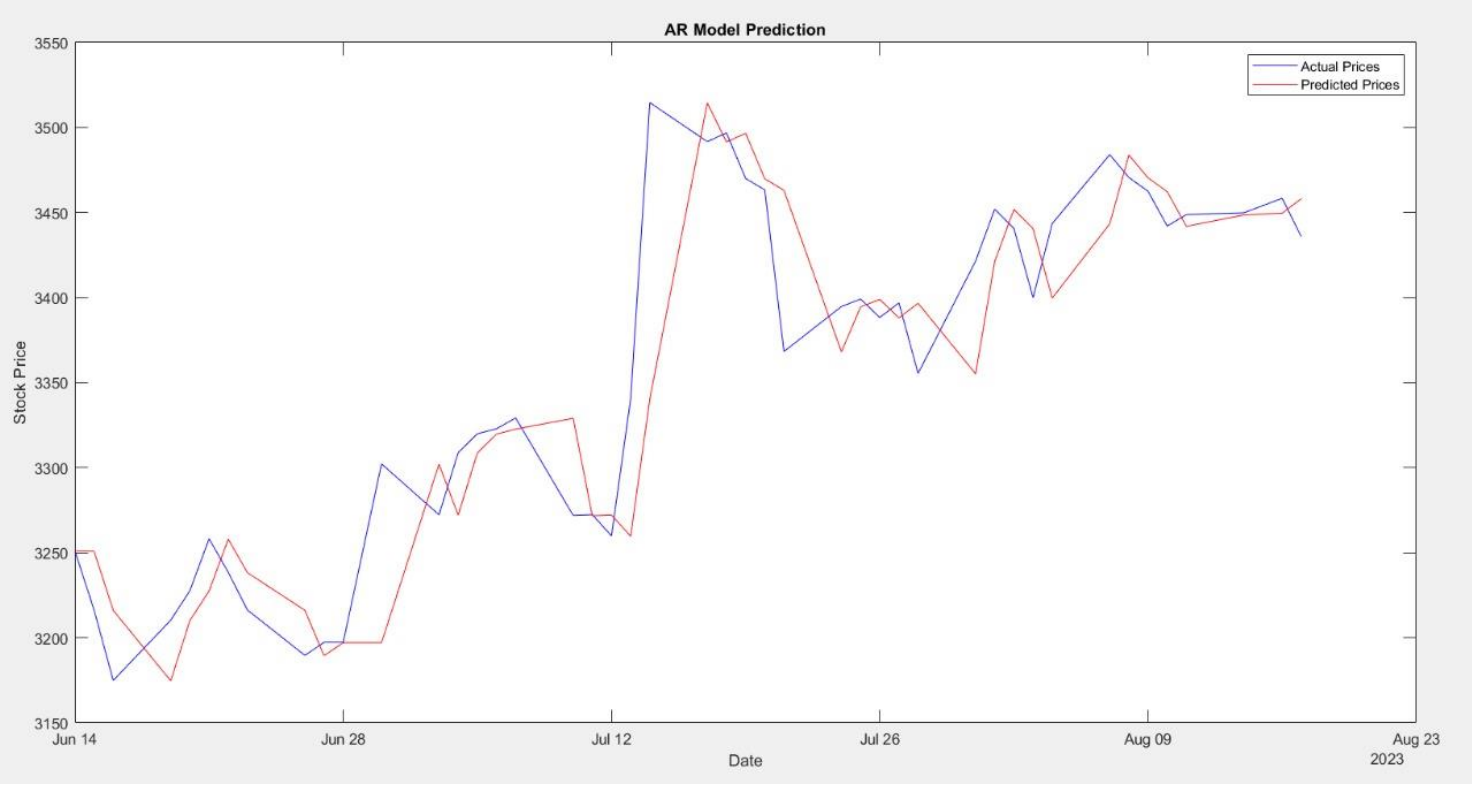
- Forecasting using AR model:

An AR(AutoRegressive) model is a type of time series model where the current value in the series is linearly dependent on its previous values. In other words, the value at any time t is a weighted sum of its past values, with the weights determined by the model's parameters.

Mathematical Expression for AR Model:

$$Xt = c + \phi 1X(t-1) + \phi 2X(t-2) + \ldots + \phi pX(t-p) + \varepsilon t$$

- Plot of AR Model:



- Code of forcasting using AR Model:

```
% Read the data from a CSV file into a table
data = readtable('TCS.NS.csv');
% Convert the 'Date' column to a datetime format for better handling of
    dates
data.Date = datetime(data.Date);
plot(data.Date, data.Close);
title('TCS Stock Prices Over Time'); xlabel('Date');
ylabel('Closing Price');
% Define the split ratio for training and testing data
splitRatio = 0.8;
% Get the total number of observations in the dataset
numObs = size(data, 1);
% Calculate the number of observations for training data based on the
    split ratio
numTrainObs = round(splitRatio * numObs);
% Divide the data into training and testing sets
trainData = data(1:numTrainObs, :);
testData = data(numTrainObs+1:end, :);
% Fit an autoregressive (AR) model of order 1 using the closing prices of
    the training data
arModel = ar(trainData.Close, 1);
% Predict future values using the AR model based on the test data
[predictedValues, predictionError] = predict(arModel, testData.Close, 1);
figure;
plot(testData.Date, testData.Close, 'b', 'DisplayName', 'Actual Prices');
    hold on;
plot(testData.Date,   predictedValues, 'r', 'DisplayName', 'Predicted
    Prices');
xlabel('Date');
ylabel('Stock Price');
title('AR Model Prediction');
legend('show');
```

- This above code is a MATLAB script that demonstrates how to work with time series data and perform autoregressive (AR) modeling for stock price prediction.

❖ Conclusion of ARIMA implementation:

We learned a lot about the ARIMA and AR models through this. We decided to take a step further by using Python and including Fourier transforms and technical indicators in addition to ARIMA predictions.

❖ Fourier Transforms:

In addition to using ARIMA, we can also incorporate Fourier Transforms into our predictive model to analyze and forecast time series data. Fourier transforms a mathematical technique that decomposes a signal into its underlying frequencies. By analyzing the frequency components of a signal, we can identify patterns and trends that may be difficult to see in the original data.

- But what is a Fourier transform?

❖ Discrete Fourier Transform (DFT):

In mathematics, the Discrete Fourier Transform (DFT) converts a finite sequence of equally spaced samples of an function into a same-length sequence of equally spaced samples of the Discrete Time Fourier Transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence. An Inverse DFT (IDFT) is a Fourier series, using the DTFT samples as coefficients of complex sinusoids at the corresponding DTFT frequencies. It has the same sample-values as the original input sequence. The DFT is therefore said to be a frequency domain representation of the original input sequence. If the original sequence spans all the non-zero values of a function, its DTFT is continuous (and periodic), and the DFT provides discrete samples of one cycle. If the original sequence is one cycle of a periodic function, the DFT provides all the non-zero values of one DTFT cycle.

The discrete Fourier transform transforms a sequence of N complex numbers
{xn}: =x0,x1,…,xN-1 into another sequence of complex numbers,
{Xk}: =X0,X1,…,XN-1, which is defined by

$$X_{2\pi}(\omega) = \sum_{n=-\infty} x[n]\, e^{-i\omega n}.$$

It is related to signal processing as a discrete version (i.e. samples) of the Discrete-time Fourier transform (DTFT), which is a continuous and periodic function. The DFT computes N equally spaced samples of one cycle of the DTFT. Since it deals with a finite amount of data, DFT can be implemented in computers by numerical algorithms or even dedicated hardware. These implementations usually employ efficient Fast Fourier transform (FFT) algorithms.
To apply Fourier transforms to our financial data, we will first calculate the Fourier transform of the closing prices using the NumPy library and then plotted the Fourier transforms alongside the actual closing prices to visualize the frequency components of the data.

❖ Fast Fourier Transform (FFT):

The Fast Fourier Transform (FFT) is an efficient algorithm to calculate the DFT of a sequence. It is a divide and conquer algorithm that recursively breaks the DFT into smaller DFTs to bring down the computation. As a result, it successfully reduces the complexity of the DFT from $O(n^2)$ to O(nlogn), where n is the size of the data. This reduction in computation time is significant especially for data with large N, therefore, making FFT widely used in engineering, science and mathematics. The FFT algorithm is the Top 10 algorithm of the 20th century by the journal Computing in Science & Engineering.

- ### Symmetries in the DFT:

The answer to how FFT speeds up the computing of DFT lies in the exploitation of the symmetries in the DFT. From the definition of the DFT equation:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N}$$

we can calculate the

$$X_{k+N} = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi(k+N)n/N} = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi n} \cdot e^{-i2\pi kn/N}$$

Note that, $e^{-i2\pi n}=1$, therefore, we have

$$X_{k+N} = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} = X_k$$

with a little extension, we can have

$$X_{k+i\cdot N} = X_k, \text{ for any integer } i$$

This means that within the DFT, we clearly have some symmetries that we can use to reduce the computation.

- ### Tricks in FFT:

Since we know there are symmetries in the DFT, we can consider using it to reduce the computation, because if we need to calculate both X_k and X_{k+N}, we only need to do this once. This is exactly the idea behind the FFT. Cooley and Tukey showed that we can calculate DFT more efficiently if we continue to divide the problem into smaller ones. Let's first divide the whole series into two parts, i.e. the even number part and the odd number part:

$$
\begin{aligned}
X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \\
&= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i2\pi k(2m)/N} + \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i2\pi k(2m+1)/N} \\
&= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i2\pi km/(N/2)} + e^{-i2\pi k/N} \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i2\pi km/(N/2)}
\end{aligned}
$$

We can see that the two smaller terms which only have half of the size (N/2) in the above equation are two smaller DFTs. For each term, the 0≤m≤ N/2, but 0≤k≤N, therefore, we can see that half of the values will be the same due to the symmetric properties we described above. Thus, we only need to calculate half of the fields in each term. Of course, we don't need to stop here; we can continue to divide each term into half with the even and odd values until it reaches the last two numbers; then calculation will be really simple. This is how FFT works using a recursive approach.

- FFT Applications:

In signal processing, FFT forms the basis of frequency domain analysis (spectral analysis) and is used for signal filtering, spectral estimation, data compression, and other applications. Variations of the FFT such as the short-time Fourier transform also allow for simultaneous analysis in time and frequency domains. These techniques can be used for a variety of signals such as audio and speech, radar, communication, and other sensor data signals. FFT is also sometimes used as an intermediate step for more complex signal processing techniques.

- FFT in Finance:

Fourier analysis attempts to identify patterns or cycles in a time series data set which has already been normalized. In particular, it seeks to simplify complex or noisy data by decomposing it into a series of trigonometric or exponential functions, such as sine waves. Each of these sine waves would have a specific cycle length, amplitude, and phase relationship with the other sine waves, which then could be added back together to reconstruct the observed data.

By first identifying and removing any effects of spurious trends or other complicating factors from the data set, the effects of periodic cycles or patterns can be identified more accurately, leaving the analyst with a better estimate of the direction that the data under analysis will take in the future.

Fourier analysis methods are frequently implemented in algorithmic trading as a technical analysis tool for forecasting market direction and trends. Numerous studies have explored Fourier analysis for practical value in forecasting stock market prices. Because Fourier analysis seeks to break down repetitive waveforms into harmonic components and the stock

market doesn't move in a well-defined and repetitive manner, results are mixed, as most similar strategies are.

❖ Technical Indicators

We incorporated technical indicators as features in our deep learning model to enhance its robustness. Specifically, we will be utilizing several indicators, including the Exponential Moving Average (EMA) with period lengths of 20, 50, and 100, the Relative Strength Index (RSI), the Moving Average Convergence Divergence (MACD), and the On-Balance-Volume (OBV).

More details on these Technical Indicators:

- EMA-:

I. It is commonly used indicator that calculates the average price of an asset over a specified time period, with more recent prices given greater weight

II. The EMA is a moving average that places a greater weight and significance on the most recent data points.

III. Like all moving averages, this technical indicator is used to produce buy and sell signals based on crossovers and divergences from the historical average.

IV. Traders often use several different EMA lengths, such as 10-day, 50-day, and 200-day moving averages.

Formula for EMA:

$$EMA_{\text{Today}} = \left(\text{Value}_{\text{Today}} * \left(\frac{\text{Smoothing}}{1+\text{Days}}\right)\right) + EMA_{\text{Yesterday}} * \left(1 - \left(\frac{\text{Smoothing}}{1+\text{Days}}\right)\right)$$

- RSI-:

I. It is used to determine whether an asset is overbought or oversold and is calculated using the average gain and loss of the asset's price over a given time period.

II. The RSI provides technical traders with signals about bullish and bearish price momentum, and it is often plotted beneath the graph of an asset's price.

III. An asset is usually considered overbought when the RSI is above 70 and oversold when it is below 30.

I. The RSI line crossing below the overbought line or above the oversold line is often seen by traders as a signal to buy or sell.

Formula for RSI:

$$RSI_{\text{step one}} = 100 - \left[\frac{100}{1+\frac{\text{Average gain}}{\text{Average loss}}}\right]$$

- MACD:

I. It calculates the difference between two moving averages and is often used to identify changes in momentum.

II. The moving average convergence/divergence (MACD, or MAC-D) line is calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA. The signal line is a nine-period EMA of the MACD line.

III. MACD triggers technical signals when the MACD line crosses above the signal line (to buy) or falls below it (to sell).

IV. MACD can help gauge whether a security is overbought or oversold, alerting traders to the strength of a directional move, and warning of a potential price reversal.

V. MACD can also alert investors to bullish/bearish divergences (e.g., when a new high in price is not confirmed by a new high in MACD, and vice versa), suggesting a potential failure and reversal.

VI. After a signal line crossover, it is recommended to wait for three or four days to confirm that it is not a false move.

MACD Formula:

MACD = (12-Period EMA) - (26-Period EMA)

- OBV:

I. It is a volume-based indicator that tracks the cumulative volume of an asset and is used to identify trends in volume.

II. To measure a security's OBV, you need to understand the relationship of closing prices between two successful trading days.

- When the second day's price closes above the prior day's close:
  OBV = Previous OBV + Current trading volume
- If prices closed lower on the second day:
  OBV = Previous OBV − Current trading volume

❖ Model Training:

Model training is the process of teaching a machine learning or deep learning model to make predictions or classifications by exposing it to a dataset. During training, the model learns from the data, adjusting its internal parameters to minimize the difference between its predictions and the actual target values. This process involves splitting data into training and testing sets, choosing an appropriate model, using a loss function to measure prediction accuracy, optimizing the model's parameters, and evaluating the model's performance. Once successfully trained, the model can be deployed for real-world applications.

Long Short-Term Memory (LSTM) deep learning model was utilized for predicting the closing price of the stock. For this purpose, we set Mean Squared Error (MSE) as the model loss, which helped to assess the degree of error in our predicted values.

After defining and training our deep learning model, we can visualize its performance through the training and validation loss evolution plot. This plot shows the changes in both the training and validation loss over time, providing insight into how the model is learning and improving.

❖ Evaluation of the Model:

After training our LSTM model, we calculated various metrics to evaluate its performance on the test data. Some important evaluation metrics are as follows:

1. Mean Squared Error (MSE): MSE is a common metric used to measure the average of the squared differences between predicted values and actual values. A lower MSE indicates that the model's predictions are closer to the actual values, implying better performance.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2.Root Mean Squared Error (RMSE): RMSE is derived from MSE by taking the square root of the MSE. Like MSE, a lower RMSE suggests better model performance, and it is particularly useful for understanding the typical size of errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3.Mean Absolute Error (MAE): MAE measures the average of the absolute differences between predicted and actual values. It provides an absolute measure of prediction accuracy and is less sensitive to outliers compared to MSE. A lower MAE indicates better model accuracy.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

4.R-squared Score (R2 Score): R2 score, also known as the coefficient of determination, evaluates how well the model's predictions fit the actual data. It ranges from 0 to 1, where 1 indicates a perfect fit, and 0 means that the model's predictions are no better than simply using the mean of the target variable. A higher R2 score indicates a better model fit.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

5.Explained Variance Score: Similar to the R2 score, the explained variance score measures the proportion of variance in the target variable that the model can explain. A higher explained variance score suggests that the model is effectively capturing the underlying patterns in the data.

$$EV = 1 - \frac{Var[Y - \widetilde{Y}]}{Var[Y]}$$

6.Mean Absolute Percentage Error (MAPE): MAPE calculates the average percentage difference between predicted and actual values. It's a relative error metric, useful for understanding the scale of errors relative to the actual values. MAPE is expressed as a percentage, and a lower value indicates better model performance.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

7.Mean Percentage Error (MPE): MPE quantifies the average difference between predicted values and actual values as a percentage of the actual values. A positive MPE indicates that, on average, the model's predictions are higher than the actual values, while a negative MPE implies that the predictions are lower. It offers insights into the direction of prediction errors.

$$MPE = \frac{100\%}{n}\Sigma\left(\frac{y - \widehat{y}}{y}\right)$$

❖ Conclusion:

In this project, we delved into various techniques for predicting stock prices. We began with the utilization of the ARIMA model, a time-series forecasting tool that leverages historical data to make future projections. Next, we explored Fourier Transforms, which enabled us to break down the time series into its underlying frequency components and integrate these components into our model. Lastly, we investigated the incorporation of technical indicators like the Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) to create additional features for our model.

To assess the performance of our models, we used several key metrics, including mean squared error (MSE), mean absolute error (MAE), R2 score, explained variance score, mean absolute percentage error (MAPE), and mean percentage error (MPE).

In summary, our findings highlight the effectiveness of combining diverse methods and techniques, resulting in a more precise and robust prediction model. By incorporating a variety of features and metrics, we gained deeper insights into the underlying trends and patterns within the data, thereby enabling more informed predictions regarding future stock prices.

- ## Understanding the Model Results:

I. Mean Squared Error (MSE): This metric indicates the average squared disparity between predicted and actual values. A lower MSE value of 29.9% signifies relatively low prediction error.

II. Mean Absolute Error (MAE): This metric measures the absolute difference between predicted and actual values. An MAE value of 0.45 suggests that, on average, the model's predictions deviate by approximately $0.45 from the actual values.

III. R2 Score: A statistical measure indicating how well the model's predictions align with actual data. With a value of 0.965, the model is capable of explaining 96.5% of the variance in the target variable, indicating strong performance.

IV. Explained Variance Score: This metric quantifies the proportion of variance in the target variable that the model can account for. A value of 0.988 once again signifies the model's ability to explain 98.8% of the variance in the target variable, reflecting solid performance.

V. Mean Absolute Percentage Error (MAPE): It quantifies the average percentage difference between predicted and actual values. An MAPE value of 46.08 indicates that, on average, the model's predictions deviate by around 46.08% from the actual values. Despite our efforts, we could not make it more satisfactory.

VI. Mean Percentage Error (MPE): This metric calculates the average difference between predicted and actual values as a percentage of the actual values. The negative value of -19.42 implies that, on average, the model's predictions are slightly lower than the actual values.

❖ Relevance of this Project with Signal Processing:

Financial analysis and signal processing may seem like two distinct fields, but they are connected in several ways, especially in the context of analyzing financial time series data. Here are some keyways in which financial analysis is relevant to signal processing:

1. Time Series Analysis:
   - Time series analysis is fundamental to both financial analysis and signal processing. Financial data, such as stock prices, exchange rates, and economic indicators, are essentially time series data, where observations are collected over time.
   - Signal processing techniques, including time series analysis, are applied to study and understand the temporal patterns, trends, and fluctuations in financial data. Methods like Fourier transforms, auto-correlation, and moving averages are used to extract meaningful insights and patterns.

2. Filtering and Smoothing:
   - Both fields benefit from filtering and smoothing techniques. In financial analysis, digital filtering can be used to remove noise and smooth out the inherent volatility in financial markets.
   - Smoothing techniques help isolate important signals or trends from the random fluctuations present in financial data. These techniques aim to make the data more interpretable and reduce noise.

3. Frequency Analysis:
   - Frequency analysis is relevant in both financial analysis and signal processing. In signal processing, it involves studying the frequency components of a signal.
   - In financial analysis, it can include identifying periodic patterns or cycles in financial time series data. Techniques like spectral analysis and wavelet transforms can be applied to uncover hidden patterns or cycles in both financial and signal data.

4. Feature Extraction:
   - Feature extraction is a common task in both fields. In financial analysis, features represent specific attributes of financial data, such as technical indicators (e.g., moving averages, Relative Strength Index), trading volumes, or returns.
   - In signal processing, features can represent characteristics of a signal, such as amplitude, frequency, and phase. Feature extraction is employed to reduce the dimensionality of data and facilitate more effective modeling and prediction in both contexts.

5. Prediction and Forecasting:
   - Signal processing techniques, such as autoregressive models and moving averages, can be adapted to financial data for prediction and forecasting.
   - For example, predicting stock prices or future values of economic indicators shares similarities with predicting future values in a signal processing context, where signals may

represent anything from audio and image data to sensor readings.

In summary, the intersection of financial analysis and signal processing lies in their common use of time series data and various analytical tools. Signal processing techniques can enhance the accuracy and effectiveness of financial analysis, helping to extract valuable information from financial time series data and contributing to better decision-making in financial markets. The shared methodologies and approaches make these fields highly relevant to each other, particularly in the context of analyzing and making predictions based on time-dependent financial data.

❖ Resources used:

GENERAL
https://www.ee.iitb.ac.in/student/~vishrant/dsp-finance.pdf
https://www.math.utah.edu/~gustafso/s2017/2270/projects-2016/williamsBarrett/williamsBarrett-financial-securities-PowerPoint.pdf

ARIMA IMPLEMENTATION
https://youtube.com/playlist?list=PLvcbYUQ5t0UHOLnBzI46_Q6QKtFgfMGc3&feature=shared
https://caciitg.com/resources/tsa/
https://www.statisticshowto.com/adf-augmented-dickey-fuller-test/
https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322
https://www.statisticshowto.com/adf-augmented-dickey-fuller-test/
https://in.mathworks.com/help/curvefit/smoothing-data.html
DFT
https://brilliant.org/wiki/discrete-fourier-transform/
https://en.wikipedia.org/wiki/Discrete_Fourier_transform
https://www.tutorialspoint.com/digital_signal_processing/dsp_discrete_fourier_transform_introduction.htm
FFT
https://en.wikipedia.org/wiki/Fast_Fourier_transform
https://in.mathworks.com/help/matlab/ref/fft.html

Technical indicators
https://www.ig.com/en/trading-strategies/10-trading-indicators-every-trader-should-know-190604
https://www.investopedia.com/terms/m/macd.asp
https://www.investopedia.com/ask/answers/121714/what-onbalance-volume-obv-formula-and-how-it-calculated.asp

Evaluation of model
https://www.dataquest.io/blog/understanding-regression-error-metrics/
https://www.oreilly.com/library/view/machine-learning-algorithms/9781789347999/49c4b96f-a567-4513-8e91-9f41b0b4dab0.xhtml

THANK YOU

-