Cairo University

Faculty of economics & political science

Statistics Department

English Section

Forecasting EGX-30 Closing Index

Presented By

Mariam Amir 5200948

Shahd Hesham 5200384

Yahia Quolquela 5200901

Youssef Hatem 5200836

Under The Supervision of

Dr. Sara Osama

Abstract

Predicting stock market prices is challenging due to the dynamic and unpredictable nature of the market. This research paper investigates various methods for predicting the EGX 30 closing index, a key indicator of the Egyptian stock market. The study compares traditional techniques such as the ARIMA model, deep learning techniques like neural networks, and machine learning techniques including Random Forest classification.

The research employs an innovative approach by treating the stock market prediction task as a classification problem, which has shown promising results in machine learning applications. By transforming the prediction task into a classification problem, the paper aims to reduce forecasting errors and provide more accurate investor predictions.

The findings of the study reveal the strengths and limitations of each method. The ARIMA model, while providing short-term predictions for the next day's closing price, has limited ability to forecast multiple upcoming days. On the other hand, the LSTM (Long Short-Term Memory) neural network demonstrates the potential to predict multiple future closing prices. Additionally, the Random Forest classifier achieves an accuracy of 73.76% in predicting the direction of the EGX 30 closing index.

The implications of these findings are significant for investors and financial decision-makers. The predicted stock prices from the comparative study can assist investors in maximizing their investment income and making more informed financial decisions. By understanding the strengths and limitations of different prediction techniques, investors can employ a combination of methods to enhance their trading strategies and improve their overall performance in the stock market.

Contents

Tables

Figures

Equations

Abbreviations

Abbreviations	Meaning
Abs Diff	Absolute difference
ACF	Autocorrelation function
AR	Auto Regressive
AIC	Akaike Information Criterion
ARIMA	Auto-Regressive Integrated Moving Average
CBE	Central bank in Egypt
EGID	Egypt for Information Dissemination
EGX	Egyptian Exchange
EMA	Exponential moving average
EM	Emerging markets
FN	False-negative
FP	False positive
FDI	Foreign Direct Investment
IPO	Initial public work
LSTM	Long short-term memory
MA	Moving Average
MAPE	Mean absolute percentage error
PACF	Partial Autocorrelation function
RMSE	Residual mean square error
Rel Diff	Relative difference
RF	Random Forest
ROC	Receiver operating curve
TN	True negative
TP	True positive
US	United states

Chapter1: Introductory about EGX-30 Closing Index.

In this chapter, we will talk about the outline of our thesis in the introduction: the dataset used and what models will be used to perform the analysis. Also, we will examine previous literature to see which models are frequently used in Time series analysis, the results, and the limitations of their analysis. this helps in deciding the approach we will take in our analysis and determining the main objectives of this study

1.1 Introduction

The Egyptian Exchange (EGX), the Egyptian stock market, plays a crucial role in the country's financial system. It provides investors with a wide range of investment opportunities and is considered an indicator of the overall state of the economy. Understanding the Egyptian stock market, its dynamics, and forecasting stock prices is essential for analysts, investors, and policymakers, as it reflects the performance of listed companies. Accurate stock price forecasting is vital for managing risks, making well-informed investment decisions, and creating efficient economic policies, especially considering continuous inflation in Egypt.

Predicting stock market prices has always been challenging due to the market's unpredictable and dynamic nature. Traditional forecasting methods can be effective but often struggle to account for the stock market's volatility, posing significant risks to investment decisions. Errors in forecasting can lead to substantial market risks, making it crucial to reduce these errors for safer investments. Thus, this study aims to forecast the Egyptian stock market's closing index price using conventional and contemporary methods.

Initially, we will perform a time series analysis by constructing an ARIMA model using the dataset from 2016 to 2024. This traditional approach will help identify trends, seasonality, and irregularities in the dataset, providing insights into market changes over the past years and aiding in projecting future trends. Subsequently, we will employ modern techniques, recognizing the power of deep learning in predicting stock market prices. Specifically, we will utilize Long Short-Term Memory (LSTM) models and build a machine learning model to compare the performance of these approaches.

Furthermore, we propose an innovative approach to reduce forecasting errors by treating the prediction task as a classification problem, a common method in machine learning. Treating stock prediction as a classification problem often yields better results than regression models. We will use ensemble learning, specifically the Random Forest Classification Algorithm, which combines multiple decision trees to improve prediction accuracy. Various technical indicators will be computed and used as features for training our model.

The dataset used in this project will primarily focus on the EGX-30 index, denominated in the Egyptian pound, which includes the top 30 companies in terms of liquidity and activity. By comparing the performance of the ARIMA, LSTM, and Random Forest models, we aim to identify the most effective method for predicting stock prices. The intended contribution of this paper is to provide predicted stock prices to help investors maximize their investment income and make more informed financial decisions.

1.2 Literature Review

Time series forecasting in financial markets has garnered extensive research attention, particularly in predicting future stock price trends. Various prediction algorithms *(Khaidem et al., 2016; Weibang et al., 2017; Chniti et al., 2018; Essam et al., 2020; Tikki Wal et al., 2020; Wenjie Lu et al., 2020; Bhowmick et al., 2021; Jain et al., 2022; Zafar et al., 2023)* have been explored, often challenging the Efficient Market Hypothesis (EMH) *(Malkiel and Fama, 1970)*, which suggests that stock prices reflect all available information, leaving no room for consistent prediction gains from historical data analysis. Despite this theory, numerous studies have showcased algorithms that effectively model complex financial dynamics, thus questioning the EMH's implications *(Malkiel and Burton, 2003).*

The use of ensemble learning has shown promise. For example, ***Khaidem et al. (2016)*** *[15]* treated the work as a classification problem and used random forests to forecast stock prices. By combining technical indicators like the Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), and the stochastic oscillator, Williams %R, Price Rate of Change they produced better outcomes throughout a range of prediction periods. Their results highlighted the possibility of using ensemble approaches to capture complex market behaviors—but only in certain datasets such as the Apple Inc. dataset and GE Dataset Which are listed on (NASDAQ) and Samsung Electronics Co. Ltd. (Which is traded in the Korean Stock Exchange) and it is shown that 3 Months prediction model outperform from other model 1 Month, 2 Months prediction model for all datasets

Deep learning models, particularly Long Short-Term Memory (LSTM) networks, have also demonstrated remarkable success. ***Weibang et al. (2017)****[27]* compared LSTM with other models like CNN-LSTM and LSTM-attention-LSTM across diverse datasets including stock prices and gold prices and Electricity and air quality and PM2.5 dataset Their study highlighted LSTM-attention-LSTM as particularly effective as it has lowest RMSE and MAPE, leveraging attention mechanisms to enhance forecasting accuracy in volatile markets like electricity and gold and the poor prediction performance of the Encoder-decoder-LSTM model This is because the complexity of the model is higher compared to the other models.

Chniti et al. (2018) *[10]* paragraph discusses the development of a robust forecasting model for predicting phone prices in European markets. Two approaches, namely Long Short-Term Memory (LSTM) neural network and Support Vector Regression (SVR), are employed in both univariate and multivariate settings, in the univariate approach, the SVR model exhibits a more accurate performance compared to the LSTM model, in the multivariate approach, the SVR model with a linear kernel effectively forecasts peak changes when the variation amplitude is not significant. However, it tends to make incorrect predictions during significant changes, the LSTM model benefits from incorporating additional time series data leading to more accurate predictions Moreover, utilizing 200 memory blocks in the LSTM model allows for accurate detection of peaks in the forecasted prices. Overall, the findings suggest that the SVR model performs better in the univariate approach, while the LSTM model demonstrates superior performance in the multivariate approach

In another study, ***Essam et al. (2020)****[8]* used different neural network architectures to estimate Egyptian stock exchange indices which are (EGX-30, EGX-70 EGX-50-EWI, EGX-100, NILE) daily from 2008 to 2019 while Nile from 2014 to 2019 such as Nonlinear autoregressive neural network with exogenous (NARX), Levenberg–Marquardt (LM), proposed NARX model, scaled conjugate gradient (SCG), Bayesian regularization (BR) instead of traditional statistical methods (regression, autoregressive, integrated moving average, exponential average, etc.) because traditional methods often fail to produce accurate predictions when dealing with multiple nonlinear characteristics. They obtained 108 individual prediction results and 6 forecast time horizons 3 model specifications (BR, LM, and SCG). Each dataset was divided into training, validation, and testing sets with the ratio of 70:15:15 with the BR model demonstrating promising accuracy for EGX-30 and on the other hand the average accuracy produced by Levenberg–Marquardt remained 99.92%, 99.85%, 99.95% and 99.89% for EGX-50-EWI, EGX-70, EGX-100, and NILE, respectively. It can be inferred from the findings that the LM algorithm was more accurate compared to BR. Their findings demonstrated the adaptability of neural networks in financial forecasting, prompting more investigation into optimal model designs.

Comparative analyses have further enriched the field. ***Tikki Wal et al. (2020)*** *[18]* compared artificial neural networks (ANN) and random forests (RF) in stock price predictions in certain datasets such as historical datasets for the five companies collected from Yahoo Finance (JP Morgan, Nike, Johnson and Johnson, Goldman Sachs and Pfizer). The dataset includes stock prices for 10 years from 2009 to 2019, it was shown that ANN proved adept at capturing nonlinear patterns and providing better performance according to Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Bias Error (MBE)

Wenjie Lu et al. (2020) *[26]* conducted a thorough comparison of machine learning models for forecasting the Shanghai Composite Index dataset which composed of the daily trading dataset of 7083 trading days from July 1, 1991, to June 30, 2020 and claimed that CNN-BiLSTM-AM is the best model to predict the stock closing price of the next day so they compared it with other models which are MLP, CNN, RNN, LSTM, BiLSTM, CNN-LSTM, CNN-BiLSTM, BiLSTM-AM, so it was shown that the CNN-BiLSTM-AM hybrid model as exceptionally accurate and values of MAE and RMSE are the lowest among all models and R-squared closed to be 1. Their study highlighted the benefits of integrating convolutional and recurrent architectures with attention mechanisms for enhanced prediction.

Moreover, hybrid models combining traditional statistical methods and deep learning have also been explored. ***Bhowmick et al. (2021)*** *[1]* discussed a study on predicting future stock prices of three key stocks from the National Stock Exchange of India: Infosys, ICICI, and Sun Pharma, representing the IT, Banking, and Health sectors respectively. The dataset from January 2004 to December 2019 is collected and split for training and testing purposes. Six models are developed, including Holt-Winter Exponential Smoothing, ARIMA, Random Forest, MARS, RNN, and LSTM, with LSTM demonstrating improved predictive abilities. While all models achieved high accuracy, ARIMA was identified as the most accurate among time series and econometric models, MARS performed best among machine learning models, and LSTM excelled among deep learning models. ARIMA's proficiency in forecasting sequential datasets, MARS's feature selection ability and handling of nonlinearity, and LSTM's capability to handle complex sequential datasets contributed to their respective successes.

Similarly, ***Zafar et al. (2023)*** *[14]* found ARIMA models more effective than LSTM according to different criteria such as R Square Score, MAE, MAPE, RMSE, and Median Absolute Percentage Error in predicting real estate prices from Mulkia Gulf Real Estate from Saudi Exchange data, suggesting that traditional methods can still excel depending on the forecasting task.

According to ***Jain et al. (2022)*** *[13],* the performance of certain deep learning models will change according to fixed length window in the past as an explicit input and in using the function of different look-back window sizes and different amounts of time to predict into the future, so they used hourly Beijing Air Quality Dataset obtained from the UCI website their analysis. This dataset covers five years from January 1, 2010, to December 31, 2014, and the target variable in the dataset is the 'PM2.5' column, while all other variables, including time, were utilized as input features. They used the Long Short-term Model (LSTM), Recurrent Neural Networks (RNN), Gated Recurrent Unit (GRU), and Transformer Model, which they evaluated for single-step and multi-step predictions in time series forecasting. For single-step predictions, the transformer network is superior when the look-back window is large ranging from 96 hours or more, while LSTMs and GRUs perform better with smaller window sizes. In multi-step predictions, the transformer network consistently outperforms other methods, and the optimal look-back window for achieving the best results is a moderate size window which is either 48 or 96 hours.

In conclusion, recent advancements in time series forecasting highlight the dynamic evolution driven by machine learning and deep learning models, which consistently outperform traditional methods. Nevertheless, the optimal choice of model hinges on the unique characteristics of the dataset and the specific forecasting objectives.

In our research, we will prioritize enhancing model scalability and fine-tuning hyperparameters. We aim to compare three distinct approaches: neural networks, ARIMA Model, and random forest classification. This comparative analysis will enable us to discern the strengths and limitations of each method, guiding us toward more effective strategies for forecasting in complex financial environments.

1.3 Main Objectives

The main objective of this paper is to study the trend of the EGX30 index from 2016 to 2024 and examine the factors that affect its fluctuations. To achieve this, we will perform statistical, machine learning, and deep learning models to forecast the EGX30 index and to compare the performance of traditional time series forecasting methods (such as ARIMA), deep learning techniques (such as LSTM neural networks), and machine learning approaches (such as Random Forest) in predicting the closing price of the EGX 30 index.

In the ARIMA model (Chapter 3), the main objective was to investigate the ability of the ARIMA model to forecast the EGX-30 closing index in the short term. Also, analyze the performance of the ARIMA model in capturing the linear and stationary patterns, and identify the optimal ARIMA model parameters that best fit the EGX 30 index. Finally, evaluate the performance of the constructed model using suitable accuracy measures and examine its limitations in forecasting the upcoming stock prices.

Additionally, in constructing the machine learning model, we will transform the forecasting problem into a classification problem (chapter 5) by predicting the direction of the EGX 30 index movement (up or down). By constructing a random forest model, we'll be able to the future direction of the EGX-30 index based on the historical dataset. Identify the suitable hyperparameters for the dataset and evaluate the performance of the model and its predictive power using suitable techniques. Assess the robustness and generalization ability of the Random Forest classifier in making accurate predictions on out-of-sample dataset.

Moreover, in the deep learning section (chapter 4), our main objective was to investigate the capability of LSTM neural networks to capture the complex, nonlinear patterns and dependencies in the EGX 30 index by constructing an LSTM model to forecast the closing prices of our index. By then we'll be able to evaluate the accuracy of the LSTM models in predicting future closing prices and provide insights into the strengths and limitations of the LSTM approach in the context of stock market forecasting for the EGX-30 closing index.

We’ll apply a comparison between the Three models using accuracy measures, analyzing their ability to capture the nonlinear patterns in the stock market dataset, and deciding which model represents the dataset best. Examine the potential of the LSTM and Random Forest models to outperform traditional time series forecasting methods in predictions of the EGX 30 index.

Chapter2: Data Description and Descriptive Analysis of EGX-30

In this chapter, we provide detailed descriptive statistics for the daily dataset related to the Egyptian stock market indices (EGX-30) from 2016 to February 2024. The chapter starts with a short presentation of the EGX-30, defining its role as a significant performance indicator of the Egyptian stock exchange, followed by a descriptive analysis of the dataset using statistical measures and visualization to underline trends and variations and record significant events that impacted the index over the investigated period.

2.1 Dataset Description

This research utilizes a daily dataset on Egyptian stock market indices (EGX-30) from 2016 to February 2024. The dataset was obtained from Egypt for Information Dissemination (EGID), a governmental entity providing authorized datasets for the Egyptian Exchange (EGX). EGID's established role and extensive data offerings ensure the accuracy and reliability of the data used in this research. The dataset consists of 1989 observations with 4 variables: index open price, high, low & index close price. And we split our dataset into train and test. Our training dataset is around 70% percent starting from the year 2016 till the end of 2022 and the testing dataset starts from 2023 till the end of February 2024

2.2 Descriptive Analysis

Brief Introduction about EGX-30

The Egyptian Stock Market launched the EGX-30 index in 2003, previously named CASE30 (Cairo and Alexandria Stock Exchanges), representing the main stock market index. In 2009, EGX published EGX-30 in US dollars. The index reflects the performance of the top 30 firms with high liquidity and activity in various sectors and is usually used to assess the performance of the Egyptian stock market.

Moving forward, we'll be introducing the assumptions and selecting criteria of the 30 top companies. Starting with the assumptions, EGX-30 assumes that the 30 highly ranked selected companies are representative of the Egyptian stock market and that large companies affect the index more than small ones.

Shedding light on the criteria for selecting the companies, we'll start with the main and most important one which is liquidity. The index assures us that the thirty companies are liquid which means that they can be either bought or sold without affecting their price. Liquidity is highly connected to the total value traded, which reflects the activity of trading during the period. When companies are highly liquid, this results in a large total value traded. EGX-30 also specified a minimum number of trading days of 78 for companies to be able to join the index. Moreover, the index restricted a minimum free float trade ratio of 15%, which means that at least 15% of the company's shares are available for public trading not for shareholders or insiders. *[20]*

Table 2.1: Descriptive Statistics for the Closing Index

Mean	Median	Max	Min	Standard Deviation	Missing values
13182	12972	30347	5713	4029.612	0%

Firstly, the dataset doesn't include any missing values. Secondly, we found that according to standard deviation and min and max values, EGX-30 has a variation of equals **4029.612** and mean equals **13182.** Moreover, we found that the highest record of EGX-30 is **30347** which was recorded on 2024-01-03, and the lowest record of EGX-30 is **5713** which was recorded on 2018-01-18

A time series plot is a graph that displays data points collected through time that helps visualize and analyze data over time. On the x-axis, the time is represented, while on the y-axis the variable of interest is represented. As previously mentioned, we have the variable of interest (EGX-30 closing index), which we'll be representing time plots.

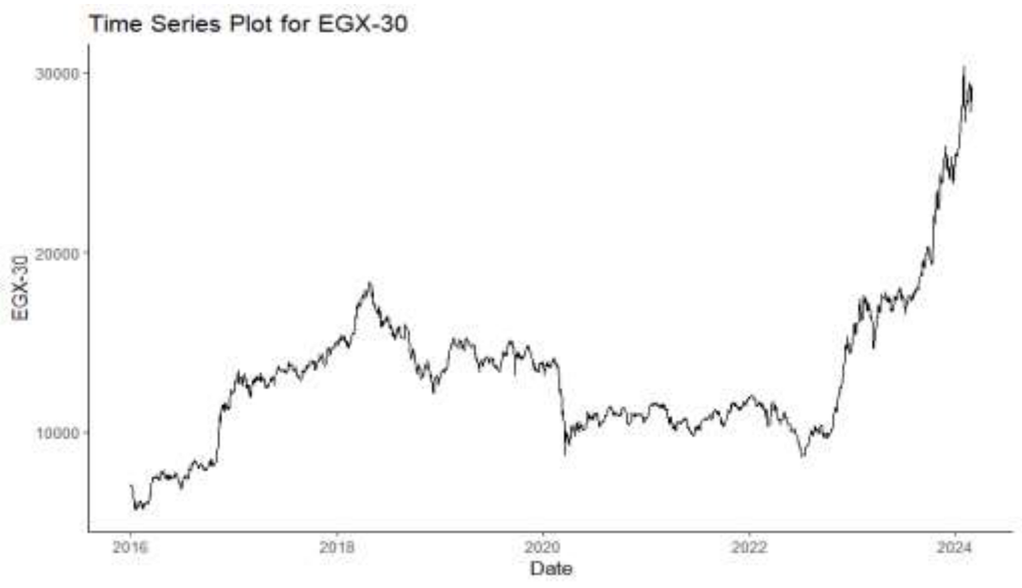

Figure 2.1: Time series plot for EGX-30 closing index during the period 2016 – now.

As shown in Figure (2.1), we have multiple turning points from the year 2016 up till 2018, we have a positive trend due to the following reasons *[9]*:

I. The first decision that was taken by the government in March 2016 that included:

- The lifting of restrictions on dollar deposits and withdrawals for companies in Egypt prevented the exit of global brands like General Motors and British Airways, providing reassurance to continue their operations.
- The Central Bank of Egypt (CBE) expanded its decision to include individuals' deposits and withdrawals, removing the monthly caps that previously restricted their ability to transact in dollars. This change has eliminated significant limitations for individuals, allowing them to operate businesses more effectively and make a greater contribution to the economy.
- Reducing the disparity between official and black-market exchange rates, the Central Bank of Egypt (CBE) devalued the currency by 14.30%, resulting in a strike price of $/EGP 8.95. This decision was supported by special dollar auctions and high-yield savings certificates offered by certain private banks. As a result, the official exchange rate has stabilized at $/EGP 8.88.

II. Despite challenges such as limited FDI and impacts on tourism, Egypt's real estate and development sector has shown signs of revival. Notable projects like Palm Hills' collaboration with SWA Group and Talaat Mostafa's Open-Air Mall faced obstacles due to restricted FDIs and tourism issues such as Russian and British travel bans and repetitive issues with Egypt Air, taking a toll on hotel revenues and occupancy rates. However, recent monetary measures, including currency devaluation and the lifting of caps on dollar transactions, are expected to have a positive impact and encourage FDI. The Egyptian Pound still faces pressure from the black market, leaving room for potential devaluation maneuvers in 2016. The real estate sector has a significant impact on various sectors.

- **Tourism Sector:**

Egypt's hospitality sector is rebounding after a challenging 2015. The currency devaluation has contributed to improved hotel occupancy rates (60%) and significant growth in average room rates (23%) and room yields (57.1%). These positive trends are particularly notable in Cairo and reflect the overall industry conditions, with 4- and 5-star hotels accounting for 66.22% of the market. *[9]*

- **Middle-Income Citizens Driving Growth in Apartment Sales:**

Egypt's residential real estate sector is poised to benefit from ongoing large-scale housing projects catering to low-to-middle-income individuals. Sales of residential apartments in New Cairo have risen by 7% year-on-year, while villa sales have declined by 12%. The government's affordable housing plan will further increase the supply of residential properties in 2016.*[9]*

Until the year 2018, we had a turning point, and we had a negative trend due to Factors like currency decline, delayed IPOs, the U.S.-China trade war, and the EM crisis affecting economic decisions. EGX's 2018 downtrend was attributed to expected interest rate hikes by the Federal Reserve, pressuring highly indebted emerging markets. The s-China trade war, the largest in Emerging Markets, led to a downturn in China's market and capital outflows from all emerging markets. Escalating trade protectionism, geopolitical tensions, and rising oil prices heightened concerns about meeting the budget deficit and inflation rate targets.*[11]*

Till we reached 2020 to 2022 and we have the lowest records of EGX-30 due to the COVID pandemic The Egyptian Stock Market (EGX) continued its decline due to the pandemic, prompting a half-hour trade suspension as 79 shares in 29 companies fell by over 5%. This marks the fourth suspension since March. EGX30 dropped by 8%, the biggest among Arab markets, losing over 800 points in early trading. *[6]*

We reached 2023 and we have a turning point that recorded the highest records of EGX-30 for the first time enabling it to break the 25,000 points threshold, The EGX30 main index soared by 70.53% throughout the year, with listed companies' market capitalization rising by 78.9% to reach LE 1.72 trillion. Treasury bills transactions were introduced on market screens for the first time during the outgoing year, totaling LE 2.68 trillion in government securities trade. Overall transactions amounted to LE 3.42 trillion.*[19]*

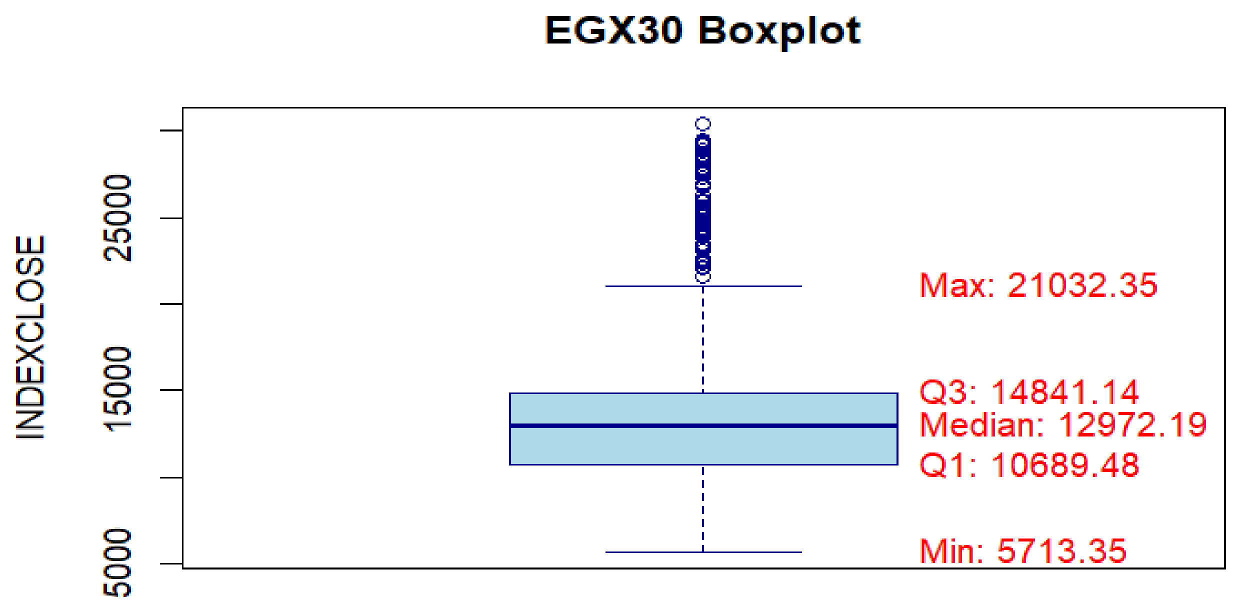

Figure 2.2: Boxplots for the EGX-30 closing price.

The boxplot of **EGX-30** showed that we have multiple outliers due to the sudden increase in recorded points that were shown and explained in the time series plot that EGX-30 breaks the 25,000 points thresholds but we will not remove or deal with those outliers as we want to study it and apply a model that able to capture it.

Chapter 3: Time Series Analysis

Classical time series approaches are statistical methods used to analyze and forecast datasets that exhibit a temporal structure. These approaches consider historical patterns and relationships within the datasets to make predictions about future values.

One of the popular time series models is ARIMA, which stands for Autoregressive Integrated Moving Average. ARIMA is a versatile and widely used model for time series analysis. It combines autoregressive (AR), differencing (I), and moving average (MA) components to capture different aspects of the time series data.

The autoregressive (AR) component of ARIMA models the relationship between an observation and a certain number of lagged observations. It assumes that the current value of the time series depends on its previous values.

The differencing (I) component is used to address non-stationarity in the dataset. It involves taking the difference between consecutive observations to remove trends or seasonality.

The moving average (MA) component models the dependency between an observation and a residual error from a moving average model applied to lagged observations.

By combining these components, ARIMA models can capture the underlying patterns and dynamics of a time series, making it useful for forecasting future values. ARIMA models are typically identified and estimated based on the patterns observed in the autocorrelation and partial autocorrelation functions of the data.

ARIMA models have been widely applied in various fields such as finance, economics, and meteorology, among others, to analyze and predict time-dependent datasets. However, it's important to note that ARIMA assumes that the data is stationary and linear and may not be suitable for all types of time series.

Our objectives in this chapter are understanding the methodology of the ARIMA model and the different concepts used in the ARIMA model fitting the suitable model generating forecast to unseen time points not used to fit the model and evaluating these forecasts using different measures of accuracy.

3.1 ARIMA Methodology

Auto-Regressive Integrated Moving Average (ARIMA) is a linear non-stationary model used for forecasting time series data.

General formula for ARIMA(p,d,q):

$$\boldsymbol{\emptyset(B) = 1 - \sum_{1}^{p} \emptyset_p (B)^p} \qquad (\mathbf{1})$$

This is the autoregressive operator where p determines the number of autoregressive terms (it is assumed to be stationary), $\nabla^d = (1 - B)^d$ is a term determining the number of non-seasonal differences needed to transform the dataset to stationary one, θ_0 is a constant term that can be omitted & $\theta(B) = 1 - \sum_1^q \theta_q (B)^q$ is the moving average operator where q determines the number of lagged forecast errors (it is assumed to be invertible).

$$\boldsymbol{\emptyset(B)\nabla^d y_t = \theta_0 + \theta(B)\epsilon_t} \qquad (\mathbf{2})$$

The "AR" part means that the value of the time series at future points in time is a linear combination of past values. For instance, to forecast yt+1, add up the last P observed values of the series, each of which is multiplied by its corresponding AR coefficient. In other words, this approach assumes that past values have some direct influence on the future values of the time series in question. The MA part models the value of the series as a linear combination of the past error terms from the previous forecasts of the model. This captures "shock" effects from past periods, meaning that if the model made large errors in the previous time step, then those errors would be used for adjustments to the current forecast. It is a combination of both components to use the past values (AR) and past errors (MA) in forecasting future values. So, with this combination, the model can adapt for errors made before (i.e., corrective action from the MA part) and can continue or reverse the trend based on past values (i.e., direction from the AR part). This makes the ARIMA model very flexible and robust to handle the various types of time series data, especially when dealing with non-stationary datasets upon integration with differencing.*[4]*

3.2 Seasonality and Stationarity

Seasonality

Seasonality is a characteristic of a time series in which the data experiences regular and predictable changes that recur at every point in time. Any predictable fluctuation or pattern that recurs or repeats over one period is said to be seasonal as *Kenton defined (2020)*. We have to test the seasonality before choosing the type of models used. If there is seasonality, we have to use seasonal ARIMA models. If it is not, non-seasonal ARIMA models should be sufficient.

The Kruskal-Walli's test is a non-parametric test that is equivalent to the parametric test one-way analysis of variance (ANOVA). The null hypothesis states that all months (or quarters, respectively) have the same mean. Under this hypothesis, the test statistic follows a Chi-Square distribution. When this hypothesis is rejected, it is assumed that time series values differ significantly between periods.

Stationarity

Stationarity of a time series means that the series has a consistent behavior with no trends. Statistically speaking, stationarity refers to a series with constant mean and variance and covariance that's independent of time. We'll use the time series plot, the Augmented Dickey-Fuller Test, and the Autocorrelation Function Plot (ACF) to check whether the series is stationary or not.

3.3 Measures of Accuracy

Mean Absolute Percentage Error

The mean absolute percentage error (MAPE), also known as the mean absolute percentage deviation (MAPD), is a performance metric used to assess the accuracy of a forecasting system. It quantifies this accuracy as a percentage and is computed by taking the average of the absolute percentage errors between the predicted values and the corresponding actual values.

MAPE is widely employed as a standard measure for forecasting errors, primarily because it expresses the error in percentage units, making it more intuitive *[1]*. It is particularly effective when the dataset does not contain extreme values or zeros. Additionally, MAPE finds application as a loss function in regression analysis and model evaluation.

$$\boldsymbol{MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_{t+i} - \widehat{Y_{t+\iota}}}{Y_{t+i}}\right|\%} \qquad (3)$$

Residual Mean Square Error (RMSE)

The RMSE is the square root of the Mean Squared error. It measures the standard deviation of residuals and the Mean Squared Error represents the average of the squared difference between the original and predicted values in the dataset. It measures the variance of the residuals.

$$\boldsymbol{RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \widehat{Y_\iota}\right|^2}} \qquad (4)$$

Akaike Information Criterion (AIC)

Measure used in statistics to evaluate and compare the quality of statistical models. It aims to find the model that best explains the dataset with the smallest number of parameters, balancing the trade-off between model fit and complexity.

$$AIC=2k-2\ln(L) \qquad (5)$$

where:

- k is the number of parameters in the model.
- L is the maximum value of the likelihood function for the model.

Model Construction

To address the traditional approach, we'll be constructing an ARIMA model for the index: EGX 30 following these steps:

1. Test stationarity.
2. Test seasonality.
3. Fixing stationarity
4. Specify the initial model.
5. Specify the best model.
6. Forecast.

3.4 Time Series Results for EGX-30

Testing Stationarity for the EGX-30 Closing Index.

In this paper, we'll test stationarity using both graphical methods including time plots and ACF, and statistical methods like the Augmented Dickey-Fuller test.

Graphical methods:

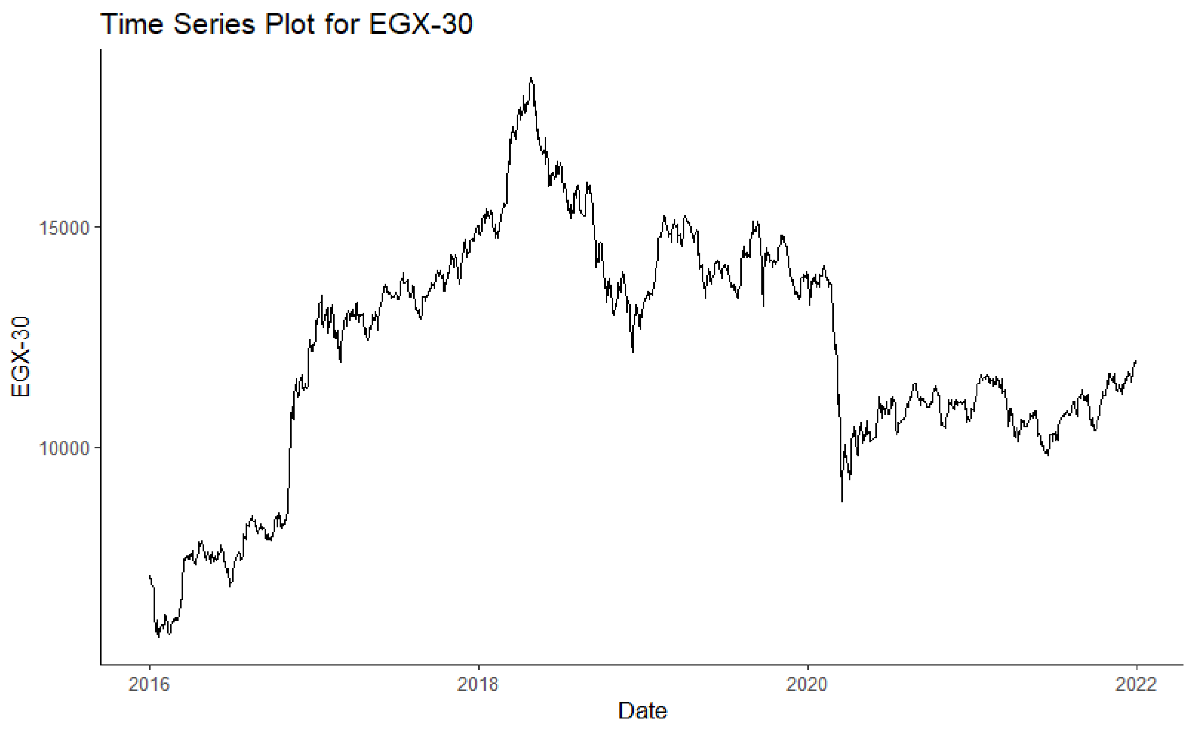

Figure 3.1: Time series plot for EGX-30 closing index during the period 2016 – 2022.

From Figure (3.1), we can notice that the series has a positive trend throughout the years until 2018 then had a negative trend till 2019, and turns to a positive trend, this is an indication that the series is not stationary in mean and from the plot it shows that we don't have very large fluctuations so we don't have seasonality pattern but we have stationarity in variance. We'll move forward to rechecking stationarity using the ACF.

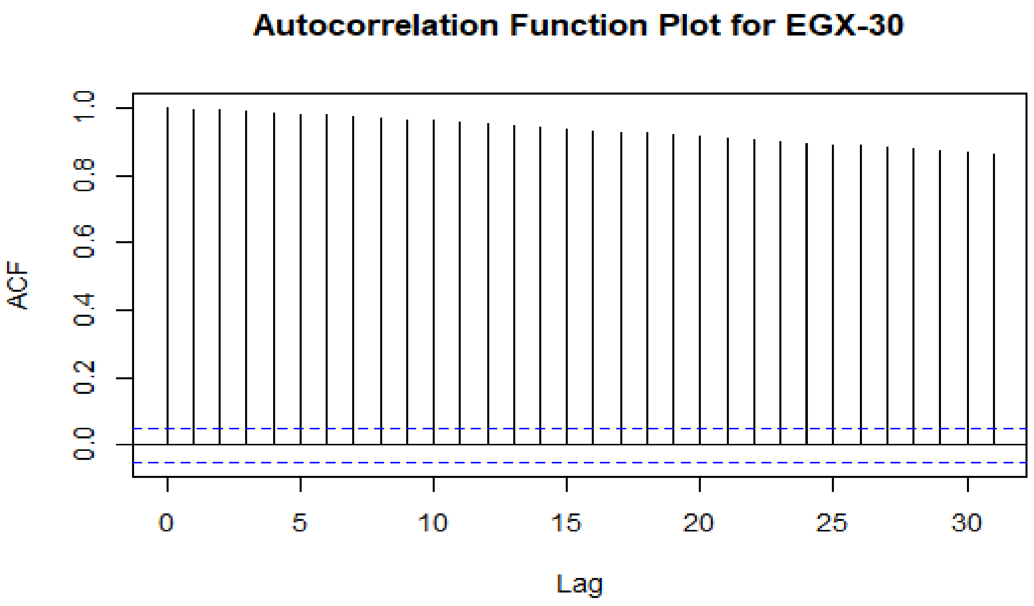

Figure 3.2: ACF for EGX-30 closing index during the period 2016 – 2022.

As it's clear from Figure (3.2), the ACF lags are decaying very slowly which means that the series is a non-stationary one.

Since both the time plots and ACF plots made us sure that the EGX-30 index series are not stationary, now we will apply a statistical test to strengthen our findings from the plots.

Applying the Dickey-Fuller Test of Stationarity

H_0: The series is non-stationary.

H_1: The series is stationary

Table 3.1: Augmented Dickey-Fuller Test

Closing Index	p-value
EGX-30	0.48>0.05

Our decision is not to reject the null hypothesis at the level of significance 0.05, therefore, we can conclude that the series of the EGX-30 closing index is non-stationary.

Testing Seasonality for the EGX-30 Closing Index

We must check seasonality to determine whether to apply the ARIMA or SARIMA model using the Kruskal Wallis test for seasonality.

H_0: There's an absence of seasonality in the series.

H_1: There's seasonality in the series.

Table 3.2: Kruskal Wallis Test for Seasonality

Closing Index	p-value
EGX-30	0.4951>0.05

Then since the corresponding p-value exceeds 0.05, we're able to say that there exists no seasonality in the series for the EGX-30 closing index. Consequently, we'll use the ARIMA model.

Fixing Stationarity

Taking both the first difference and the natural logarithm for the closing indices:

To stabilize the variance that changes over time in the data, we took the natural log of the closing index. Moreover, to make the series a stationary one, we'll take the first difference and re-check stationarity after applying both transformations.

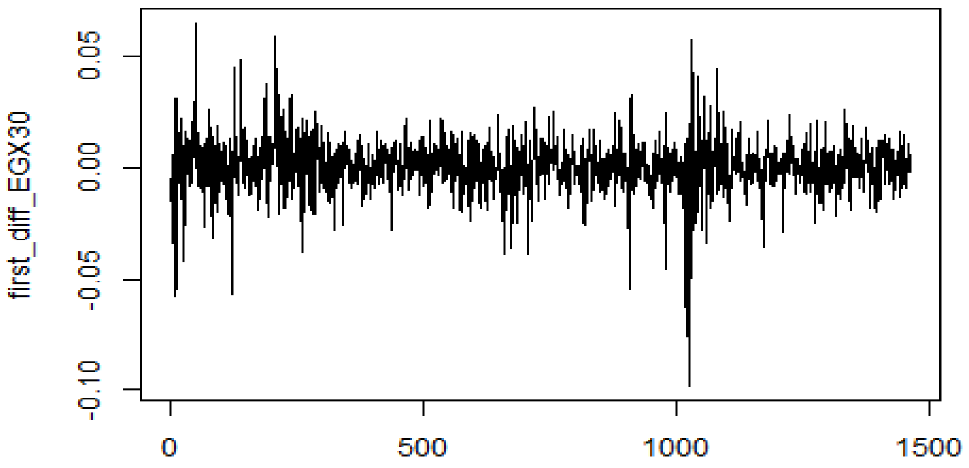

Figure 3.3: Time series plot for the first difference log (EGX-30) closing index during the period 2016 – 2022.

As shown in Figure (3.3), after taking the log transformation and the first difference, we can notice that the series has no trend now and the series becomes stationary.

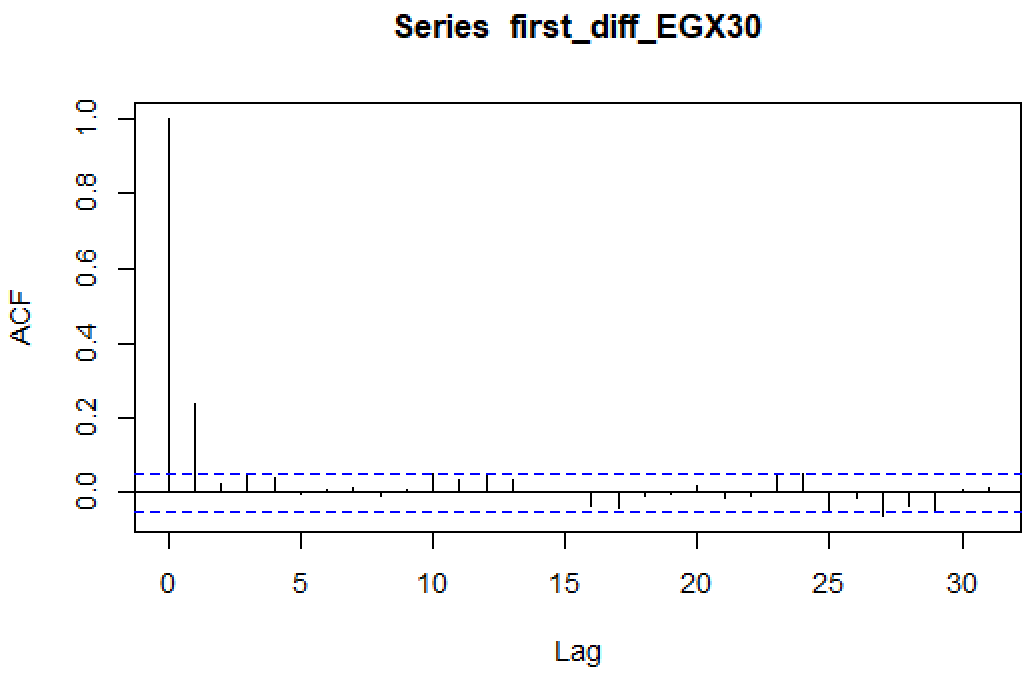

Figure 3.4: ACF for the first difference of log (EGX-30) closing index during the period 2016 – 2022.

shown in Figure (3.4), the decay of the lags is very quick, therefore we can conclude that the problem of non-stationarity is solved. The first lag is significant in addition to two more lags, therefore we'll need to assess our results with a test, so we'll conduct the Augmented Dickey-Fuller test to double-check stationarity.

Since the ACFs of the closing index have significant lags, we'll double-check the stationarity assumption using the statistically Augmented Dickey-Fuller test.

Table 3.3: Augmented Dickey-Fuller Test for the first difference of the log of the closing index

Closing Index	p-value
EGX-30	$0.01<0.05$

Since the p-value of the corresponding test is less than 0.05, then there is significant evidence that the series is stationary after taking the first difference of the natural logarithm of the closing index.

Determining the Initial Model

One way to determine the initial model is by looking at the ACF and PACF of the transformed series.

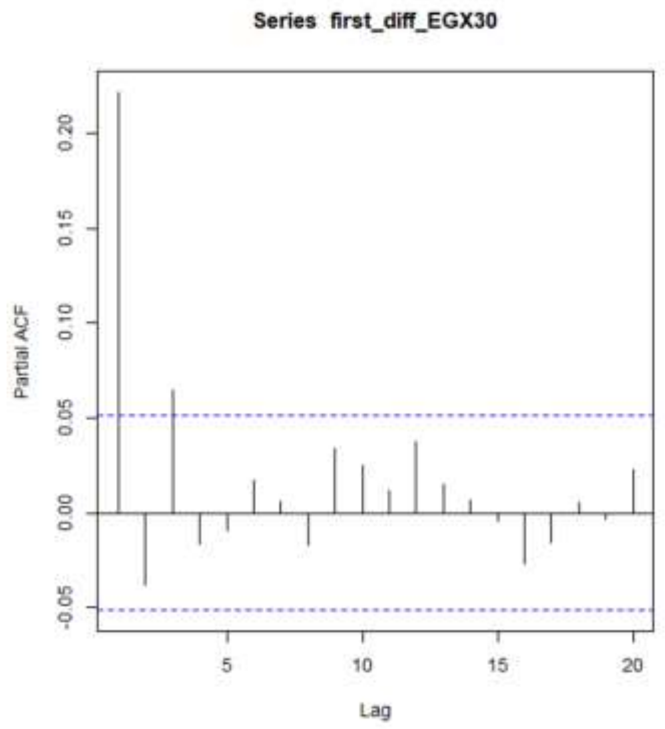

Figure 3.5: PACF for the first difference of log (EGX-30) closing index during the period 2016 – 2022.

The PACF lags in Figure (3.5) decay quickly which means that there is a possibility that the series follows an Autoregressive process (AR), but this assumption needs to be checked as the graph also shows that the third lag is significant. As for the ACF in Figure (3.4), it declines quickly after the first lag but the second and 24th lag are significant. Therefore, it's hard to determine the MA and AR order from the ACF and PACF.

Function Auto-Arima

The R function auto. Arima () utilizes a modified version of the Hyndman-Khandakar algorithm (Hyndman & Khandakar, 2008) to determine an appropriate ARIMA model. This algorithm combines unit root tests, AIC minimization, and maximum likelihood estimation (MLE) to obtain the desired model. The arguments are available for auto. Arima () offers flexibility in specifying different variations of the algorithm.

Hyndman & Khandakar, 2008 algorithm for automatic ARIMA modeling

1- The number of differences $0 < d < 2$ is determined using repeated KPSS tests.

2- The selection of values for p and q is achieved by minimizing the AIC after performing data differencing d times. Instead of exhaustively considering all possible combinations of p and q, the algorithm employs a stepwise search technique to navigate through the model space.

3- The best model (with the smallest AIC value) fitted in step (a) is set to be the "current model".

Determining the suitable ARIMA model

To mitigate the issue of not being able to determine an initial model, we will use the Auto-ARIMA function in R which is based on the AIC criteria. When using this function, the ARIMA (0,1,1) was displayed as the suitable model for the closing price of the EGX 30 Index and this will be shown in the following steps. To confirm that ARIMA (0,1,1) is the suitable model some conditions must be satisfied:

1. The coefficient must be significantly different from zero.
2. Errors are white noise.
3. Residuals follow a normal distribution.

ARIMA (0,1,1) formula and assumptions:

The general formula for ARIMA (0,1,1) is as follows:

$$(1-B)\, Y_t = (1-\theta B)\, \varepsilon_t \qquad (6)$$

B: backshift operator.

Θ: parameter of the moving average component.

This model is called "**The simple exponential smoothing model**" which uses an exponentially weighted moving average of past values. In other words, this model gives exponentially decreasing weights to past observations, and higher weights to more recent observations, it takes the average of the last few observations rather than the most recent one to forecast the next observation.

The three main assumptions of ARIMA (0,1,1) are:

1. Stationarity.
2. No autoregressive (AR) component (p=0).
3. A moving average component where (q=1).

To confirm that the model chosen is suitable, we'll check whether the residuals of the ACF and PACF are white noise or not and we'll make sure that the autocorrelation coefficients are insignificant.

- **Checking the significance of the coefficients in the ARIMA (0,1,1) model:**

$$H_0: \theta = 0$$

$$H_1: \theta \neq 0$$

When testing the coefficients of the ARIMA model, the following results were displayed:

Table 3.4: **Testing significance of the coefficient in EGX-30 closing index.**

Z-test coefficients:	Estimate	p-value
MA (1)	0.24991	2.2e^16<0.05

The p-value corresponding to the MA (1) coefficient is very small which gives us an indication that the coefficient is significantly different from zero, therefore, this indicates that the chosen model ARIMA (0,1,1) is the best model for the EGX-30 closing index.

- **Residuals:**

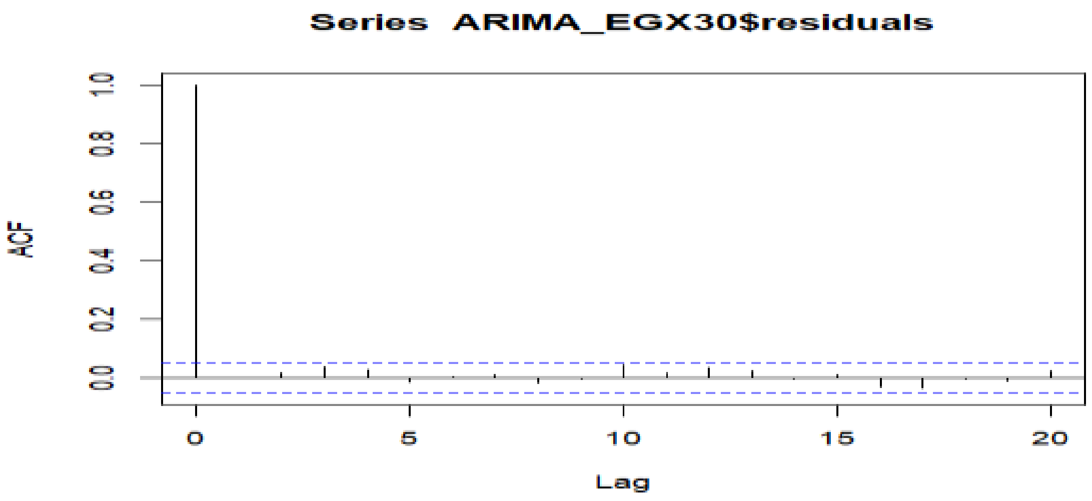

***Figure 3.6*: ACF for the residuals of the ARIMA (0,1,1) model for EGX-30.**

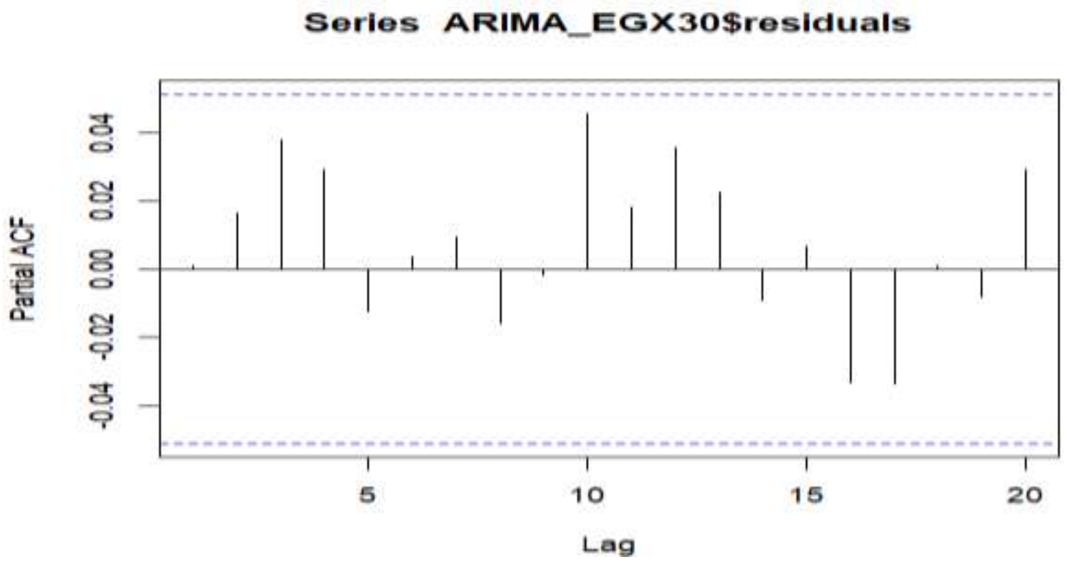

Figure 3.7: PACF for the residuals of the ARIMA (0,1,1) model for EGX-30.

From Figures (3.6) and (3.7), we can notice that all lags are insignificant except for the first lag of the ACF, but since graphs are subjective and don't always give an accurate indication, we'll use the Box-Ljung test to assess whether there exist significant autocorrelations in the residuals at the lags. To assess the significance of this test, we performed a loop in R software which iterated through 100 lag intervals, checking the significance.

- **Box-Ljung Test**

H_0: The errors are white noise (or the model is appropriate)

H_1: The errors are not white noise (or the model is not appropriate) at each one.

Table 3.5: Box-Ljung test for residuals in EGX-30 closing index.

Lag	p-value
1	0.972
3	0.475
5	0.556
28	0.1179
55	0.199
73	0.454

From the displayed results, it appears that we won't reject the hypothesis that the errors are white noise since the corresponding p-values exceed 0.05, which means that the errors are white noise.

- **Checking Normality**

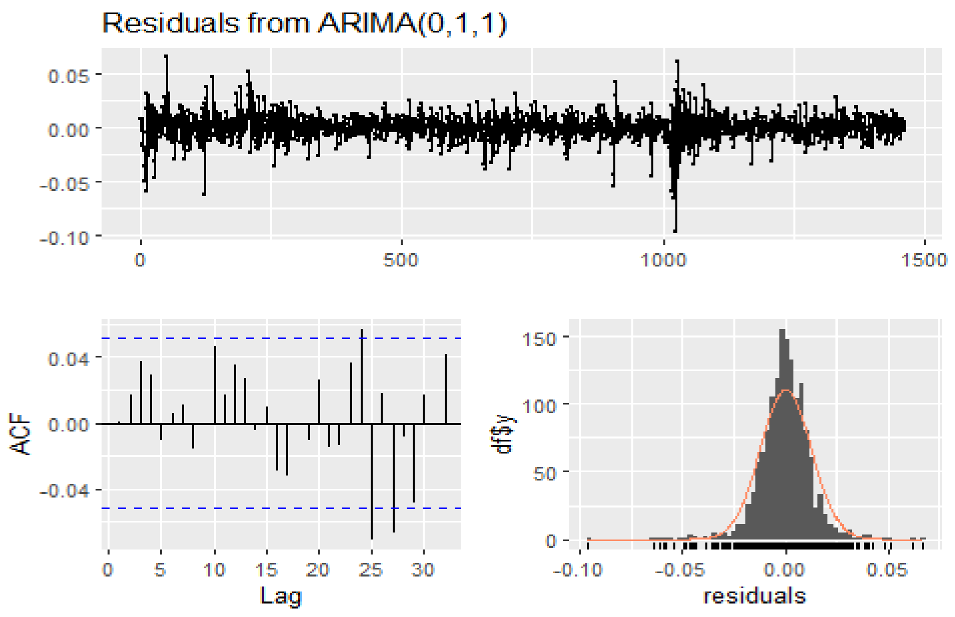

Figure 3.8: Residuals of the ARIMA (0,1,1) model for EGX-30.

From the above graph, we found that residuals are normally distributed which implies that the model is the most suitable one.

Therefore, the ARIMA (0,1,1) formula for the EGX-30 index is as follows:

$$\mathbf{(1\text{-}B)}\, Y_t = \mathbf{(1\text{-}\ 0.24991B)}\ \varepsilon_t \qquad \mathbf{(7)}$$

3.5 Forecasting

Forecasting is crucial for predicting future trends and behaviors based on historical data. It helps in making decisions related to investing in the stock market.

Since the Moving Average (MA) models forecast with only one point, we will obtain this point for every closing index and compare it with the actual one.

Table 3.6: The one-point forecast for the EGX-30 closing index.

Closing Index	Forecast	Actual
EGX-30	9.389653	9.385110

Measures of Accuracy

The absolute difference measure is a specific instance of the residual mean squared error (RMSE) metric. Similarly, the relative difference measure is a particular case of the mean absolute percentage error (MAPE) metric, but only when making a forecast for a single data point.

$$Absolute\ Difference = |\, Y_t - \widehat{Y_t}\, | \qquad (8)$$

$$Relative\ Difference = \left|\frac{Y_t - \widehat{Y_t}}{Y_t}\right| * 100 \qquad (9)$$

Table 3.7: Measures of accuracy for the closing index

Closing Index	Absolute difference	Absolute relative difference
EGX-30	0.004543	0.0484%

Chapter4: Machine Learning Approach; Deep Learning

Neural networks are a class of models inspired by the structure and function of the human brain. They consist of interconnected nodes, called neurons, organized in layers. Each neuron takes input, performs a computation, and produces an output that is passed to the next layer.

Neural networks have gained significant popularity for their ability to learn complex patterns and relationships from data. They excel in tasks such as image and speech recognition, natural language processing, and time series analysis.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that addresses the limitations of traditional RNNs in capturing long-term dependencies in sequential data. LSTM networks are designed to remember and utilize information over extended time intervals, making them particularly effective for time series analysis.

The distinguishing feature of LSTM is the presence of memory cells within each neuron. These memory cells can store information over time and selectively forget or update it based on the input. LSTM networks utilize gates, such as input, forget, and output gates, to control the flow of information through the memory cells. This gating mechanism enables LSTMs to learn and retain important contextual information over long sequences.

LSTMs have shown exceptional performance in various applications, including speech recognition, machine translation, sentiment analysis, and stock market prediction. Their ability to make long-term dependencies makes them well-suited for modeling and forecasting time series data with complex temporal patterns.

In summary, neural networks are computational models inspired by the structure of the human brain, and LSTM is a specialized type of recurrent neural network designed to capture long-term dependencies in sequential data. Both neural networks and LSTMs have revolutionized the field of machine learning and have proven to be powerful tools for analyzing and predicting complex patterns in various domains.

In this part we will investigate the details of the architecture of the LSTM model and how the model works to make a forecast then we will forecast future data using the LSTM model

4.1 Deep learning Approach methodology

The LSTM Model

Long Short-Term Memory (LSTM) networks represent a sophisticated class of recurrent neural network (RNN) architectures specifically designed to address the vanishing gradient problem that hampers traditional RNNs. LSTM networks are engineered to retain information over extended sequences, thereby capturing long-range dependencies that are critical in various sequential data tasks. *[25]*

The logic of the LSTM model

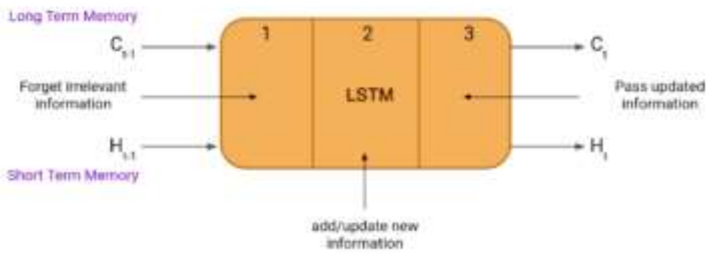

Figure 4.1: Schematic Diagram of a Long Short-Term Memory (LSTM) Cell *[25]*

The initial component determines whether the information from the previous timestamp should be retained or discarded as irrelevant. The second component allows the cell to assimilate new information from the current input. Finally, in the third component, the cell transmits the updated information from the current timestamp to the next. This cycle constitutes a single-time step of an LSTM.

These three components of an LSTM unit are referred to as gates, which regulate the flow of information in and out of the memory cell. The first gate is the **Forget gate**, the second is the **Input gate**, and the third is the **Output gate**. An LSTM unit comprising these three gates and a memory cell functions similarly to a layer of neurons in a traditional feedforward neural network, where each neuron has a hidden state and a current state. Additionally, an LSTM has a hidden state, with **H(t-1)** representing the hidden state from the previous timestamp and **H(t)** representing the hidden state of the current timestamp. Furthermore, LSTMs have a cell state, denoted as **C(t-1)** for the previous timestamp and **C(t)** for the current timestamp.*[25]*

Architecture of LSTMs

LSTM’s architecture deals with both Long-Term Memory (LTM) and Short-Term Memory (STM) and for making the calculations simple and effective it uses the concept of gates. By discovering the architecture and its mathematical formula *[23]*

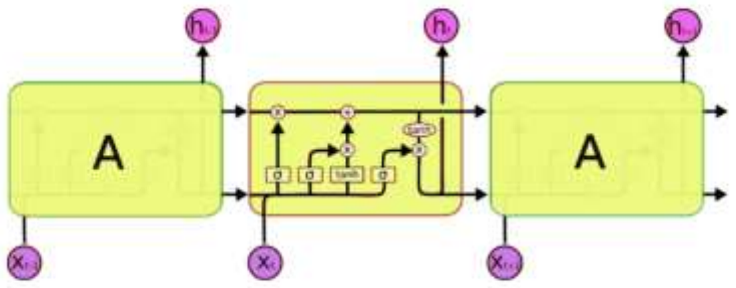

Figure 4.2: Detailed Architecture of an LSTM Cell with Input and Output Flow (LSTM) Cell *[25]*

We will discuss this architecture in detail how each gate works and the importance of each in the architecture.

- **<u>The Forget Gate</u>**

<u>Its structure:</u>

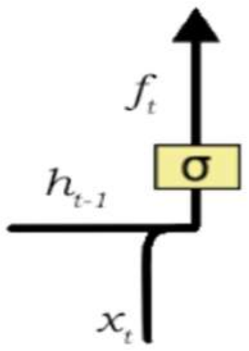

Figure 4.3: Diagram of the Forget Gate in an LSTM cell *[25]*

A forget gate is responsible for removing unnecessary information from the cell state. It discards information that is no longer required or deemed less important for the LSTM's functionality by applying a filter through multiplication. This process is crucial for optimizing the performance of the LSTM network *[25]*

$$\boldsymbol{f_t = \sigma\,(x_t * u_f + H_{t-1} * W_f)} \qquad \textbf{(10)}$$

$$\boldsymbol{\sigma(x) = \frac{1}{1+ e^{-x}}} \qquad \textbf{(11)}$$

Xt: input to the current timestamp.

Uf: weight associated with the input

Ht-1: The hidden state of the previous timestamp

Wf: It is the weight matrix associated with the hidden state

The sigmoid σ(x) is applied to it. That will make it a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp, such that

$C_{t-1} * f_t = 0$ **if** $f_t = 0$ **then (forget everything)**

$C_{t-1} * f_t = C_{t-1}$ **if** $f_t = 1$ **then (forget nothing)**

- **<u>Input Gate</u>**

  <u>Its structure:</u>

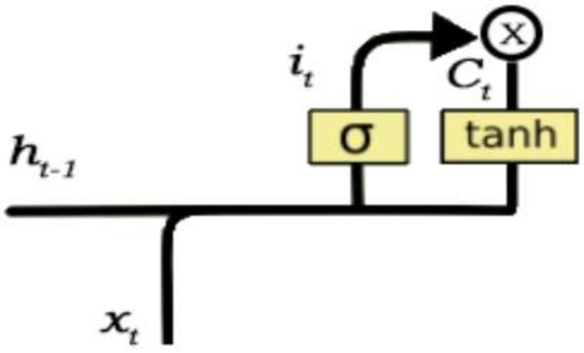

Figure 4.4: **Diagram of the Input Gate in an LSTM Cell** *[25]*

Same requirements as the output gate and the same use of sigmoid function the value of I at timestamp t will be between 0 and 1. But the difference here is that the input gate has **three important processes** *[25]*:

1- Regulating which values should be added to the cell state involves using a sigmoid function. This function, similar to the forget gate, acts as a filter for all the information from $h_{t-1}$and X_t

$$I_t = \sigma (x_t * u_i + H_{t-1} * W_i) \qquad \textbf{(12)}$$

2- Creating a vector containing all possible values that can be added (as perceived from ht-1 and xt) to the cell state. This is done using the tanh function, which outputs values from -1 to +1.

$$N_t = \tanh (x_t * u_c + H_{t-1} * W_c) \qquad \textbf{(13)}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad \textbf{(14)}$$

3- The value from the regulatory filter (the sigmoid gate) is multiplied by the vector created by the tanh function. This resulting useful information is then added to the cell state through an addition operation

$$C_t = f_t * C_{t-1} + I_t * N_t \qquad \textbf{(15)}$$

➢ **<u>Output Gate</u>**

<u>Its structure:</u>

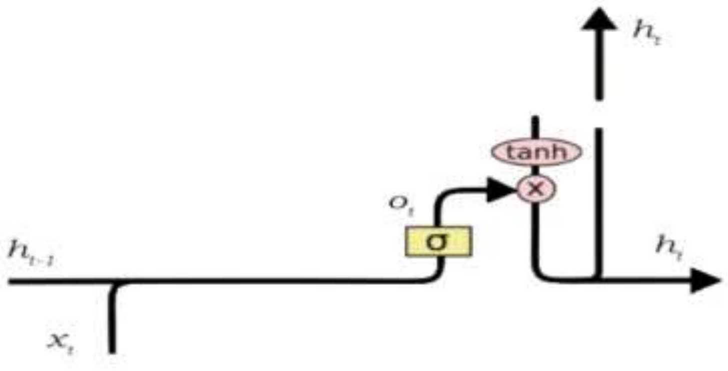

Figure 4.5: Diagram of the Output Gate in an LSTM Cell *[25]*

The functioning of an output gate can be broken down into three steps *[25]*:

1- Regulating and filtering which values should be added to the cell state involves using a sigmoid function. This function, like the forget and input gate, acts as a filter for all the information from $h_{t-1}$and X_t

$$\boldsymbol{O_t = \sigma\,(x_t * u_o + H_{t-1} * W_o)} \qquad \textbf{(16)}$$

2- Create a vector after applying the tanh function to the cell state, thereby scaling the values to the range -1 to +1 and calculating the current hidden state

$$\boldsymbol{H_t = O_t * \tanh\,(C_t)} \qquad \textbf{(17)}$$

3- If you need to take the output of the current timestamp, The output layer applies a linear transformation to $\boldsymbol{H_t}$ followed by the linear activation function also known as the identity

function, simply passes the pre-activation values through without any further transformation. Therefore, the final prediction for the continuous value, denoted as **Yt**

$$\boldsymbol{Z_t = W * H_t + b} \qquad \textbf{(18)}$$

$$\mathbf{Yt} = \boldsymbol{Z_t}$$

W: is the weight matrix of the dense layer

b: is the bias vector

$\boldsymbol{Z_t}$ **:** represents the pre-activation values of the dense layer.

4.2 Model Construction and Evaluation

To apply LSTM to predict the close price of EGX30 we used Keras & tensorflow libraries. Both libraries are available in Python. We used a time window approach for the LSTM. In this study, we chose the window size to be 30 days.

We use previous values of the close price to predict future values.

$$\boldsymbol{\hat{y}_t = f(y_{t-1}, y_{t-2} y_{t-3}, \ldots\ldots, y_{t-n})} \qquad \textbf{(19)}$$

Where $\hat{y}_t$ is the forecasted value, $y_{t-1}, y_{t-2} y_{t-3}, \ldots$ are the trained values, & n is the window size. From Figure (4.6), we can see the result of the LSTM model, the LSTM forecasts are so close to their corresponding actual test data. This graph indicates that the LSTM model is effective in capturing the overall trend and general patterns. This suggests that the model can be a useful tool for forecasting future index values, provided that similar patterns continue

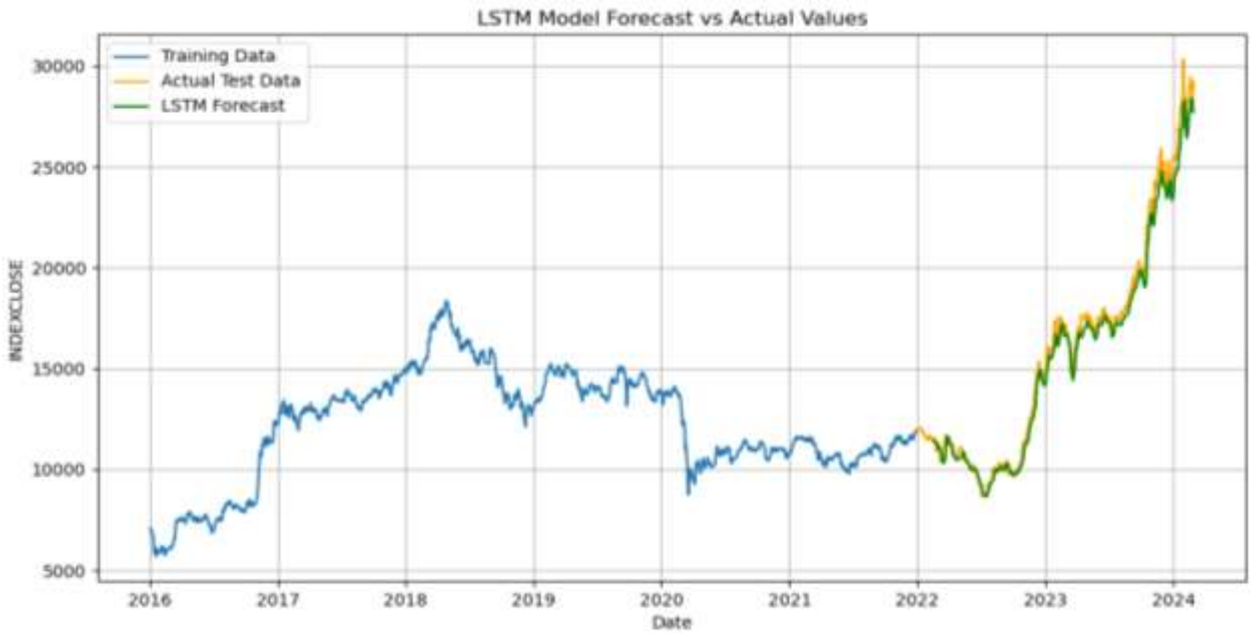

Figure 4.6: LSTM model

LSTM has a high prediction power, as Figure (4.6) shows how close the forecast is to the actual points We will use Root Mean Square Error (RMSE) & Mean Absolute Percentage Error (MAPE) to determine the accuracy measure for the model.

Table 4.1: Performance Metrics for LSTM Model

Metric	LSTM
MAPE	2.1291510615731095
RMSE	486.97410989711244

From both RMSE & MAPE, LSTM has good predictive power and can capture several trends in the close index price of EGX-30, and the LSTM model takes into consideration turning points in our data. As a specialized type of recurrent neural network, LSTM is designed to capture long-term dependencies and patterns in sequential data. This includes the ability to identify and emphasize turning points, significant changes, or transitions in our data. By leveraging their memory cells and processing the sequential data, LSTM models can effectively capture the temporal patterns and turning points, making them well-suited for forecasting our series as shown in the results and from the graph (4.6)

Chapter 5: Machine Learning Approach; Ensemble Learning Algorithm

Predicting stock market prices has always been challenging due to the market's unpredictable and dynamic nature. Traditional forecasting methods can be effective but often struggle to account for the volatility of the stock market, posing significant risks to investment decisions. Thus, errors in forecasting can lead to significant market risks, so reducing these errors is crucial for safer investments. In this Section, we propose a new approach to reduce forecasting errors by treating the prediction task as a classification problem, which is a common method in machine learning. Treating stock prediction as a classification problem, rather than a regression problem, often yields better results. The goal is to create an intelligent model that learns from market data using machine learning techniques to forecast future trends in stock prices. The predictions from this model can help investors make better decisions.

Various algorithms have been used by researchers, including Support Vector Machines (SVM), Neural Networks, and Naive Bayesian Classifiers. However, a promising advancement in machine learning is the use of ensemble learning techniques. Ensemble learning combines multiple models to improve prediction accuracy by leveraging the strengths of each model and mitigating their weaknesses. One powerful ensemble learning method is the Random Forest Classification Algorithm, which aggregates the predictions of multiple decision trees to enhance the overall model performance.

In this study, we will utilize the Random Forest Classification Algorithm to predict stock market trends. This approach not only improves accuracy but also provides robustness against overfitting, a common issue in predictive modeling. To train our model, various technical indicators are computed and used as features for the model. The study aims to demonstrate the effectiveness of the Random Forest algorithm, an ensemble learning technique, in financial market prediction.

In this part, we will investigate to understand the methodology of the random forest model and determine the variables we will include in the model construct model and evaluate its accuracy and evaluate using other measures such as sensitivity, specificity, precision, negative predictive score, and F1-score

5.1 Machine Learning Approach Methodology

Random Forests is considered a supervised machine learning algorithm that uses multiple decision trees in the aggregate to help make more stable and accurate predictions. Decision Trees are one of the building blocks of Random Forest to understand Random Forest.

Decision Trees are considered as a hierarchical model, for a simpler understanding of its structure, they could be imagined as a flowchart where each level represents a step in the decision-making process, with each level prompting a new question.

The basic principle of decision trees is **recursive partitioning** of the feature space using a tree structure which means that they learn by progressively splitting the data into smaller and more homogenous groups. The decision tree starts with the entire dataset at the **root node** and then starts splitting by asking a question on an attribute. To search for the **best feature** to split this data into two groups (**child nodes**), the tree determines the best feature split. It maximizes the purity of the child node. Purity refers to how similar the data points within a group are. They could be measured using impurity measures such as Shannon Entropy or Gini impurity.

Gini impurity is used as the function to measure the quality of the split in each node. Gini impurity at node N is given by

$$\boldsymbol{g(N) = 1 - \sum_{i=1}^{d} P(w_i)^2} \qquad \textbf{(20)}$$

Where P(ωi) represents the proportion of instances with class label ωi node N and d denotes the number of classes. This measure helps in evaluating how mixed the classes are within the node. A Gini impurity of 0 indicates perfect purity (all instances belong to a single class), while higher values indicate greater impurity and diversity of class labels within the node.

After Calculating the Gini index and the split that will be selected the data is portioned into two child nodes and each of these nodes is treated in the same way as the original node. This recursive process continues until a stopping criterion is met. For instance, the process may stop when each unsplit node has fewer than a specified number of cases. When the stopping criteria are met, nodes that are no longer split are called “**terminal nodes**.” To predict a new point using the decision tree, predictor values are used to pass down the tree from the top of the tree and follow the branches until reaching the **Terminal Node,** and the prediction for the terminal node is used as the prediction for the new point.

However, one of the disadvantages of decision trees is that they are a high-variance algorithm where slight variation in the training data can result in large changes in the learned model.

On the other hand, Random forests tackle this problem by combining the way the decision tree works with the concept of bagging a technique that involves training multiple models on bootstrapped subsets of the data and aggregating their predictions.

To elaborate, suppose that given a set of N independent observations $Z_1, Z_2 \dots \dots \dots .. Z_n$ each with variance σ^2 the variance of the mean Z bar of the observations is represented by $\frac{\sigma^2}{n}$. This mathematical expression illustrates that as the sample size increases, the dispersion in the mean diminishes proportionally. Consequently, the practice of averaging observations serves as an effective means of variance reduction. Therefore, a practical methodology for reducing variance and enhancing prediction accuracy in statistical learning involves the generation of multiple training sets from the population, creating individual prediction models based on each set, and subsequently averaging the resulting predictions.

Since having access to more than one dataset is not applicable in real life, Random forests bootstraps, by repeatedly drawing samples from the same dataset, create a decision tree and train each one on these bootstrapped datasets individually, and the final prediction is determined by aggregating the individual predictions using **Majority Voting Scheme** in classification problem *[24]*

5.2 Constructing the model

Target Variable

To transform our problem from a regression task into a classification task, we define the target for the i^{th} day as follows:

$$\boldsymbol{target_i = sign(close_{i+d} - close_i)} \qquad \textbf{(21)}$$

Here, **d** represents the number of days after which the prediction is to be made & **sign** represents the sign of the difference between close_{i+d} & close_i. A $\boldsymbol{target_i}$ value of +1 indicates a positive shift in the price after d days, while a $\boldsymbol{target_i}$value of -1 indicates a negative shift after d days. This formulation allows us to classify the direction of stock price movement. The $\boldsymbol{target_i}$values are then assigned as labels to the i^{th}row in the feature matrix, facilitating the construction of a classification model.

Technical Indicators

Technical indicators are mathematical calculations calculated from time series stock data to better understand the trading patterns of financial assets. This project will use five quantitative indicators as independent variables to predict future stock market movements.

1. *Relative Strength Index*

The Relative Strength Index (RSI) is a widely used indicator that measures the momentum of a stock determining whether it's oversold or overbought. The method of calculating the RSI is mainly based on comparing the average gains to the average losses within a timeframe of 14 days. Moreover, RSI ranges from 0 to 100, where values below 30 indicate that the stock is oversold and this usually happens when the stock's price goes down sharply below its fair, usually resulting from panic selling. On the other hand, values above 70 indicate that the stock is overbought, and this occurs when the demand unjustifiably pushes the price upwards above its fair and by then we may be ready for a trend reversal to the downside. *[15]*

$$\boldsymbol{RS = \frac{Average\ gain\ over\ the\ past\ 14\ days}{Average\ loss\ over\ the\ past\ 14\ days}} \qquad \textbf{(22)}$$

$$\boldsymbol{RSI = 100 - \frac{100}{1+RS}} \qquad \textbf{(23)}$$

2. *Stochastic Oscillator*

A stochastic oscillator is a measure of the momentum of a stock by putting the latest closing price to previous price ranges during a specific timeframe, usually 14 days.

Also, it ranges from 0 to 100, where values above 80 indicate that the stock is overly bought, and values below 20 indicate that the stock is overly sold. *[15]*

$$\%\boldsymbol{K} = \frac{\boldsymbol{Close - Low_{14}}}{\boldsymbol{High_{14} - Low_{14}}} * \mathbf{100\%} \qquad \textbf{(24)}$$

Close: the current closing price.

$\mathbf{Low_{14}}$: the lowest low over the past 14 days.

$\mathbf{High_{14}}$: the highest high over the past 14 days.

3. Williams %R

Williams %R is identical to the stochastic oscillator index except that it ranges from -100 to 0, where a value above -20 indicates that the stock is overbought and a value below -80 indicates that the stock is oversold. *[15]*

$$\%\boldsymbol{R} = \frac{\boldsymbol{High_{14} - Close}}{\boldsymbol{High_{14} - Low_{14}}} * \mathbf{-100\%} \qquad \textbf{(25)}$$

4. Moving Average Convergence Divergence

It measures changes in a stock’s momentum, strength, and trend. It mainly tracks the difference between two exponential moving averages of a stock's price. *[15]*

$$\boldsymbol{MACD = EMA_{12}(C) - EMA_{26}(C)} \qquad \textbf{(26)}$$

$$\boldsymbol{Signal = EMA_{9}(MACD)} \qquad \textbf{(27)}$$

$\mathbf{EMA_n}$: the exponential moving average of the stock price in the past (n) days.

C: the closing price series

The signal line is used along with the MACD index to represent the ups and downs movements of the market. We have two cases, first, if the MACD line is above the signal line then this indicates that the stock prices might increase. The other case is if the MACD line is below the signal line, this will indicate that the stock prices might decrease. It is calculated using the MACD values of the past 9 time points to calculate EMA_9. *[15]*

5. ***Price Rate of Change***

The price rate of change calculates the percentage change of a stock price in (n) days ago. In our case, we used **n=9.** It generates both positive and negative values where positive values indicate an upward momentum and a negative value indicates a downward momentum. *[15]*

$$\boldsymbol{PROC(t)} = \frac{\boldsymbol{Close(t)-Close(t-n)}}{\boldsymbol{Close(t-n)}} \mathbf{*100\%} \qquad \mathbf{(28)}$$

Close(t): closing price at time (t).

Note that technical indicators in financial analysis rely on some past data points (n) to calculate their current value. For example, the Relative Strength Index, the Stochastic Oscillator, and Williams %R use the last 14 closing prices to calculate their current values, therefore, the first 13 days are considered as lagged values. While the Price Rate of Change uses the last 9 days to calculate its current value, the first 8 days are lagged values. In other words, the technical indicators take time into consideration.

Model Configuration & Training Process:

A Random Forest Classifier was employed to predict the direction of stock price movement. This ensemble learning method utilizes multiple decision trees, offering advantages like improved prediction accuracy and reduced overfitting compared to single decision trees *[Breiman, 2001]*. The model was configured with the following parameters:

The training process involved the following steps:

1. **Data Splitting:** The dataset was divided into two sets: train_data for model training and test_data for evaluating its performance. The split was based on the INDEXDATE column. As described in the data description part.
2. **Feature Selection:** Specific features from the DataFrame were chosen for training: ' RIS', 'Stochastic Oscillator ', 'Williams R', 'Price_Rate_Of_Change', and 'MACD'. These features presumably represent technical indicators commonly used in stock price prediction.

3. **Target Variable:** The target variable is the binary classification label, indicating either an "Up Day" (1) or "Down Day" (0) based on the closing price change.
4. **Model Fitting:** The Random Forest Classifier was created with the defined parameters and subsequently fitted on the training data
5. **Prediction:** The trained model was used to make predictions on the unseen test data

Optimal Model

Since the Random Forest results depend on certain parameters which are the number of trees in the forest (n_estimators), the number of features considered for splitting at each node (max_features), the maximum depth of each tree (max_depth), the minimum number of samples required to split an internal node (min_samples_split), the minimum number of samples required at each leaf node (min_samples_leaf), and whether bootstrap sampling was used (bootstrap), then determining the optimal parameters is crucial to determine the best model. The process of optimizing hyperparameters for our Random Forest model involved employing the RandomizedSearchCV method from sklearn.model_selection. This approach emphasizes exploring a diverse range of hyperparameter values while leveraging cross-validation to assess their impact on model performance. The strategy aims to pinpoint the optimal combination of hyperparameters that maximizes predictive accuracy, tailored specifically to our dataset's characteristics.

The Randomized Search approach contrasts with Grid Search, which is an exhaustive search of all potential combinations. Instead, Randomized Search selects a predetermined number of combinations from the hyperparameter space. This not only lowers computing costs but also raises the likelihood of discovering a near-optimal solution in a realistic timeframe. The trade-off between thoroughness and performance makes Randomized Search ideal for models with a large number of hyperparameters or when computational resources are restricted. In our implementation, we defined a range of potential values for each hyperparameter of the Random Forest model.
We investigated settings for the number of trees ranging from **100** to **2000**, with increments of **200**. This range was selected to strike a balance between computational efficiency and model performance since performance is often improved up to a certain number of trees. By varying the

number of features considered for splitting between **'Auto', 'Sqrt',** $\boldsymbol{'log_2'}$**,** and **'None'**, we could test several approaches to taking features into account during splits. The **'Auto'** and **'Sqrt'** settings use the square root of the total number of features as the maximum number of features to consider at each split. While **'**$\boldsymbol{log_2}$**'** setting uses the logarithm base 2 of the total number of features as the maximum number of features for each split. The **'None'** setting uses the total number of features at each split.

To regulate the complexity of the model and prevent overfitting, the maximum depth of each tree parameter was evaluated in increments of **10** from **10** to **100** which means lower values reduce model complexity and help prevent overfitting, while higher values potentially capture more complex patterns but may overfit if not regularized properly. There was also a choice for no restriction, or **'None'** which means nodes are expanded until all leaves are pure (contain only instances of the same class) or until all leaves contain fewer samples than the minimum number of samples required to split an internal node. This can lead to deeper trees that fully exploit the training data but may increase the risk of overfitting.

To enhance how our Random Forest model develops, we tested different setups for the minimum number of samples required to split an internal node and the minimum number of samples required at each leaf node. Where, the min_samples_split parameter ranging from **2** to **40**, decides how many samples are needed to split a node inside the tree. This controls how finely the tree can divide the data, helping to avoid overfitting by making sure splits are based on enough samples. Meanwhile, min_samples_leaf was checked with values like **1, 2, 7, 12, 14, 16**, and **20** to ensure each leaf in the tree contains a minimum number of samples. This helps the model generalize well and prevents it from learning overly specific details from the training dataset. We also looked at two sampling methods: bootstrap sampling, where we randomly sample with replacement from the training dataset, adding randomness to the models, and using the entire dataset for each tree, which ensures all data is considered equally. These adjustments aimed to find a good balance between model complexity, how well it predicts, and its ability to work well across different datasets and real-world situations.

To assess the performance of the model, the Randomized Search was carried out across 100 distinct hyperparameter combinations using 3-fold cross-validation. The purpose of this configuration was to preserve computing efficiency while guaranteeing a thorough exploration of the hyperparameter space. We assured the reproducibility of our results by providing a random seed (random_state=42). By using this approach, we were able to effectively and systematically explore the hyperparameter space and find the configuration that offers the optimal trade-off between predictive capability and model complexity. Through the utilization of Randomized Search's advantages, we customized our Random Forest model to the distinct features of our dataset, hence enhancing its efficiency and efficiently allocating computational resources.

In evaluating the performance of our Random Forest model optimized through Randomized Search, we identified the optimal configuration of hyperparameters. The model was trained using **900** trees, which were found to provide the best balance between complexity and predictive power. For the tree structure, a maximum depth of **70** was chosen, ensuring that the model could capture intricate relationships in the data without overfitting. The decision nodes were split based on the minimum number of samples required, set to **2**, while each leaf node was constrained to contain at least **20** samples, promoting generalization and robustness in predictions. The choice of not restricting the number of features allowed the model to consider all available features at each split, optimizing its ability to capture relevant patterns. Bootstrap sampling was employed during training to enhance the model's variability and overall performance.

5.3 Model Evaluation:

To evaluate how well the model is performing and to provide robustness to the model, some measures were calculated, namely, accuracy, precision, sensitivity, and specificity.

$$Accuracy = \frac{tp+tn}{tp+tn+fp+fn} \quad (29) \qquad Sensitivity = \frac{tp}{tp+fn} \quad (31)$$

$$Precision = \frac{tp}{tp+fp} \quad (30) \qquad Specificity = \frac{tn}{tn+fp} \quad (32)$$

$$Negative\ Predictive\ Value\ (NPV) = \frac{tn}{tn+fn} \ (33) \qquad F1 - Score = 2 * \frac{percision*Recall}{Percision+Recall} \ (34$$

tp: number of true positive values

tn: number of true negative values

fp: number of false positive value

fn: number of false negative values.

Following parameter optimization, the model achieved an accuracy of **73.76%** on the test dataset, indicating its ability to correctly classify instances.

Classification Report

The classification report further illustrates the model's performance across different classes. Precision, recall, and F1-score metrics were computed for classes labeled as "Down Day" and "Up Day".

Precision measures the proportion of correctly identified samples in a population of samples that are classified as Up Days. The precision of the model was 78.72%, meaning that 78.72% of the instances the model classified as Up days were actually Up days. Negative Predictive Value NPV measures the proportion of correctly identified samples in a population of samples that are classified as down Days. NPV Was 68.4% meaning that 78.72% of the instances the model classified as Down Day were true Down Days.

Sensitivity (Recall for Up Day) is the percentage that the classifier correctly identified Up Days, which is equal to 66.5% which indicates the model's effectiveness in correctly identifying Up Days. On the other hand, specificity is the percentage that the classifier correctly identified Down Days. Specificity (Recall for Down Day) is equal to 79.84% and this indicates the models' effectiveness in correctly identifying negative labels. F1-score is the It is the harmonic mean of precision and recall (up days) or the mean of NPV and recall (down days). Since both values of the f1 score are greater than 70% (high value), then the classifier is effective at correctly identifying Up days & down days while minimizing false Up days & false Down days Respectively.

Table 5.1: Model Performance Measures

Day	Precision\ Negative Predictive Value (NPV)	Recall	F1-score
Up Day	78.72%,	66.55%	72.12%
Down Day	68.4%	79.84%	73.47%

Receiver Operating Characteristic

ROC curve is a plot of sensitivity as a function in (1-specificity) for the possible cutoff points.The closer the ROC curve is to the upper left corner, the (0,1) point, the better the predictive power of the model. The area under the curve represents an important pillar of evaluating the performance of the classifier, where an area of 1 represents the perfect classifier.

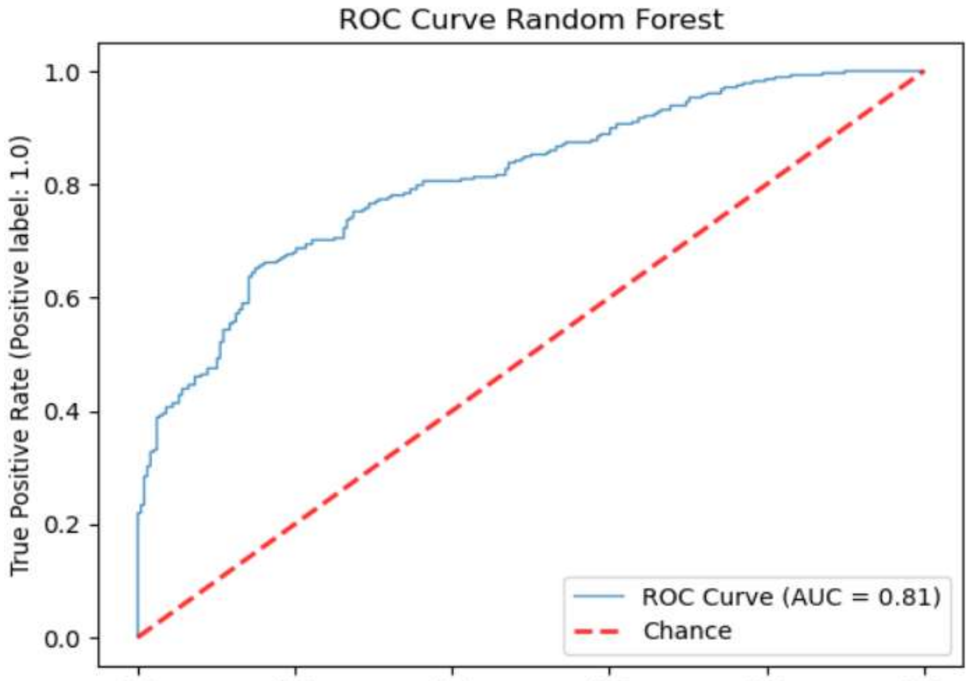

Figure 5.3 : ROC Curve for the Optimal Model.

From Figure (5.3), the AUC is equal to 0.81 which is close to 1, therefore, this highlights that the classifier has very good predictive power. These metrics provide insights into the model's ability to correctly identify instances of each class, demonstrating its strengths and potential areas for improvement.

Feature Importance

The feature importance scores for the Random Forest classifier reveal the relative significance of each feature in predicting the target variable, calculated using the Gini impurity criterion. The Stochastic Oscillator (k_percent) feature is the most influential, contributing 23.6622% to the model's predictive power. This indicates that the Stochastic Oscillator plays a crucial role in the model's decision-making process. Similarly, Williams R (r_percent) is nearly as influential, with a contribution of 23.1814%, highlighting its significant impact on the model. The Relative Strength Index (RSI) follows with a contribution of 19.8391%, demonstrating a moderate level of importance. While not as critical as Stochastic Oscillator or Williams R, RSI remains an important factor in the model's predictions. The Price Rate of Change feature accounts for 17.0354% of the model's importance, indicating a moderate impact that is less than RSI but still relevant. Lastly, the Moving Average Convergence Divergence (MACD) feature, with a contribution of 16.2819%, is the least influential among the features considered, yet it still holds relevance in the model's overall predictive capability. These important scores provide valuable insights into which features most affect the model's predictions, guiding further feature selection and model refinement efforts.

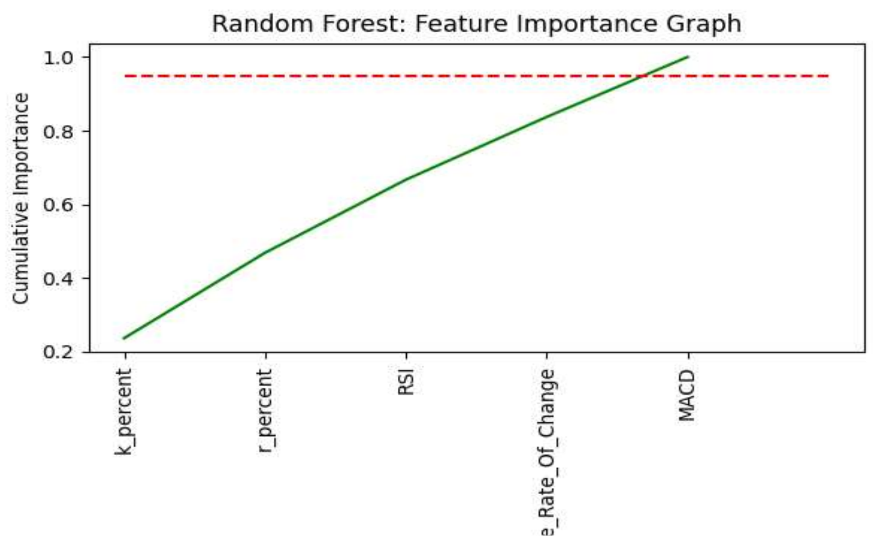

Figure 5.4 : Feature Importance for RF.

In conclusion, the optimized Random Forest model, configured with the identified set of hyperparameters, represents a robust approach for predictive modeling in the context of our research. The chosen parameters not only enhance the model's accuracy but also contribute to its interpretability and generalization capabilities, making it suitable for real-world applications where predictive accuracy and insight are paramount.

Chapter6: Conclusion and Recommendations

6.1 Conclusion

6.2 Limitation

6.3 Future recommendations

6.1 Conclusion

This research aimed to forecast the closing index price of the Egyptian stock market (EGX-30) using both conventional and contemporary methods. By employing ARIMA, LSTM, and Random Forest models, we conducted a comparative analysis to identify the most effective approach for predicting stock prices.

Our findings reveal that each model has distinct strengths and limitations. The ARIMA model, rooted in traditional statistical methods, effectively captures trends, seasonality, and irregularities in historical data, providing a reliable baseline for forecasting. However, it falls short in handling the non-linear and dynamic nature of financial markets, and in our model of ARIMA (0,1,1) it forecasts only one point despite its very close to the real point but if we need to make a new point forecast, we need to refit the model includes the training data and last point forecast so it is computationally complex and time-consuming.

The LSTM model, leveraging deep learning techniques, demonstrated superior performance in modeling complex and volatile market behaviors. Its ability to retain long-term dependencies and handle sequential data makes it particularly suitable for financial forecasting. The model showed a remarkable ability to capture variability and provide very accurate forecasts. Despite its strengths, the LSTM model requires substantial computational resources and fine-tuning of hyperparameters to achieve optimal performance.

By approaching the forecasting task as a classification problem, the Random Forest model utilized an ensemble learning method that significantly improved prediction accuracy. Incorporating various technical indicators (e.g., Moving Average Convergence Divergence, Relative Strength Index, and stochastic oscillator) as features proved beneficial. By fitting it showed promising accuracy precision and specificity and after improving our model by using hyperparameters optimization. It improved accuracy and contributed to its interpretability and generalization capabilities, making it suitable for real-world applications where predictive accuracy and insight are paramount.

Our study underscores the importance of selecting the appropriate model based on the specific characteristics of the dataset and forecasting objectives. While deep learning models like LSTM offer significant advantages, traditional methods like ARIMA and machine learning techniques like Random Forest remain valuable tools in the forecasting arsenal.

6.2 limitations

Using the LSTM model in general:

1- They exhibit greater complexity compared to conventional RNNs and necessitate a larger amount of training data to achieve effective learning.
2- They are not ideally suited for online learning tasks, such as prediction or classification tasks that do not involve sequential input data. Additionally, LSTMs can be sluggish to train when dealing with extensive datasets because they need to learn the parameters of the LSTM cells, which can be computationally demanding
3- LSTMs might not be suitable for all types of data, particularly when dealing with highly nonlinear data or data that contains a significant amount of noise, as their performance may be compromised in such cases

In our research:

1- The LSTM model takes a lot of time in the learning process (large computational effort)
2- It has many hyperparameters, we optimized only one parameter which is look back window, and unable to optimize other parameters, and optimization of all parameters may lead to overfitting

Using Random Forest

1- They face challenges when dealing with high-cardinality categorical variables, unbalanced data, time series forecasting, and variable interpretation, and are susceptible to hyperparameter sensitivity
2- The reduction in classification accuracy when redundant variables are present.
3- In research papers on Random Forest (RF), selecting appropriate performance measures, estimation methods, and a methodology for comparing multiple algorithms across multiple datasets can pose challenges.

In our research:

The hyperparameter optimization process can be time-consuming and hard. Moreover, finding the optimal set of hyperparameters can be very challenging as it requires extensive validation techniques to avoid overfitting. Overfitting happens when the model gets the training data well but can't generalize the test data, which eventually will lead to poor performance on the test dataset.

In addition, RF classification can give insights into the direction of stock prices whether it's up or down, but it may not offer the investor an accurate decision on whether to buy or not due to the additional factors that influence the stock prices including economic conditions, political issues, new events, and other factors.

Using ARIMA model

1- One of the primary drawbacks is the manual specification of parameters (p, d, q), which necessitates a time-consuming trial-and-error process to identify the optimal fit.
2- ARIMA models heavily rely on the quality and consistency of historical data, as well as the appropriate differencing of the data. Accurate and extensive data collection over a significant period is crucial to ensure the model produces accurate results and reliable forecasts.
3- This may lead to overfitting due to using high orders of (p,q,d) and if the values are too low, the model may underfit and fail to capture important patterns

In our research:

1- Our data has many fluctuations, so it takes effort to transform data into a stationary one
2- Unable to forecast all points we want. It forecasts only one point
3- Unable to capture turning points from assumptions of ARIMA to deal with stationary data we took first difference and ln transformation, so it didn't take into consideration the turning points

6.3 Future recommendations

Though the limitations encountered in our research using LSTM, Random Forest, and ARIMA models, several avenues for future work can address these limitations and further improve the predictive capabilities of the models. Some potential directions for future research include:

1**. Model Optimization**: Further exploration and optimization of hyperparameters for the LSTM, Random Forest, and ARIMA models can be undertaken. This can involve using advanced optimization techniques such as Bayesian optimization to find the optimal set of hyperparameters while avoiding overfitting. Additionally, investigating techniques for automated hyperparameter tuning, such as using auto ML frameworks, can help streamline the process.

2. **Feature Engineering:** Exploring additional features or other indicators in the data can enhance the performance of the models. This can involve incorporating external data sources such as economic indicators, news sentiment analysis, or social media data to capture relevant market trends and sentiments. Domain-specific feature engineering techniques can also be explored to extract meaningful patterns and relationships from the data.

3. **Deep Learning Architectures:** Beyond LSTM, exploring other advanced deep learning architectures such as Transformer models or hybrid models can be beneficial. Transformer models have shown significant success in various sequence-to-sequence tasks and may provide improved performance in capturing long-term dependencies in stock price data.

4. **Incorporating Domain Knowledge:** Integrating domain knowledge and expert insights into the modeling process can contribute to more accurate predictions. Collaborating with financial experts or market analysts to identify relevant features, define meaningful target variables, and incorporate market-specific constraints can lead to more informed and reliable predictions.

5**. Evaluation Metrics and Methodology:** Developing robust evaluation metrics and methodologies for comparing the performance of different models across multiple datasets is crucial. Future work can focus on identifying appropriate performance measures that align with the specific objectives of stock price prediction, considering metrics such as profitability and risk-adjusted returns. Furthermore, developing standardized methodologies for comparing models and establishing benchmarks can facilitate fair and meaningful comparisons.

6. **Hybrid Approaches:** Exploring hybrid approaches that combine traditional time series analysis techniques with machine learning algorithms can be fruitful. For example, combining ARIMA models with LSTM or Random Forest models can leverage the strengths of both approaches and potentially improve prediction accuracy.

7. **Real-Time Prediction:** Extending the research to develop real-time prediction models that can provide timely and accurate forecasts can be valuable for traders and investors. Investigating techniques for online learning and adapting models to handle sequential data in real time can be a focus of future work.

In conclusion, future research can focus on addressing the limitations and challenges encountered in our research by exploring model optimization, ensemble approaches, feature engineering, advanced deep learning architectures, incorporating domain knowledge, refining evaluation metrics and methodologies, hybrid approaches, and real-time prediction techniques. These efforts can contribute to more accurate and reliable predictions of stock prices, enabling better decision-making in the financial markets.

References

[1] A. Chatterjee, H. Bhowmick, and J. Sen, "Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models," in Proceedings of MysuruCon 2021, pp. 289-296, 2021, doi: 10.1109/MysuruCon52639.2021.9641610.

[2] CFI Team, "Autoregressive Integrated Moving Average (ARIMA)," Corporate Finance Institute, 2023. [Online]. Available: https://corporatefinanceinstitute.com/resources/datascience/autoregressive-integrated-moving-average-arima/.

[3] A. Sugandhi, "A Guide to Long Short Term Memory (LSTM) Networks," KnowledgeHut, 2024. [Online]. Available: https://www.knowledgehut.com/blog/web-development/long-short-term-memory.

[4] P. J. Davis and R. A. Davis, Introduction to Time Series and Forecasting, 3rd ed. Springer, 2016.

[5] P. S. P. Cowpertwait and A. V. Metcalfe, Introductory Time Series with R. Springer, 2009.

[6] A. H. Elalfy, "Market Watch-Was It a Good Year?" AmCham Egypt, 2021. [Online]. Available: https://www.amcham.org.eg/publications/business-monthly/issues/301/January-2021/4037/was-it-a-good-year.

[7] "What are the limitations of using Random Forest?" SciSpace. [Online]. Available: https://typeset.io/questions/what-are-the-limitations-of-using-random-forest-1s7o62pie7.

[8] E. H. Houssein, M. Dirar, K. Hussain, et al., "Assess deep learning models for Egyptian exchange prediction using nonlinear artificial neural networks," Neural Comput & Applic, vol. 33, pp. 5965-5987, 2021, doi: 10.1007/s00521-020-05374-9.

[9] "Egypt Real Estate Market: The Blominvest Report," BLOMInvest Bank SAL, 2016. [Online]. Available: https://blog.blominvestbank.com/17461/egypt-real-estate-market-blominvest-report/.

[10] H. Bakir, G. Chniti, and H. Zaher, "E-Commerce Price Forecasting Using LSTM Neural Networks," International Journal of Machine Learning and Computing, vol. 8, no. 2, pp. 169-174, 2018, doi: 10.18178/ijmlc.2018.8.2.682.

[11] H. Mohamed, "EGX downtrend in 2018," EgyptToday, 2019. [Online]. Available: https://www.egypttoday.com/Article/3/64097/EGX-downtrend-in-2018.

[12] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. OTexts, 2021.

[13] J. Shi, G. Narasimhan, and M. Jain, "Time Series Forecasting Using Various Deep Learning Models," 2022. [Online]. Available: https://www.researchgate.net/publication/361265989_Time_Series_Forecasting_Using_Various_Deep_Learning_Models.

[14] K. Albeladi, B. Zafar, and A. Mueen, "Time Series Forecasting using LSTM and ARIMA," International Journal of Advanced Computer Science and Applications, vol. 14, no. 1, pp. 313-316, 2023. [Online]. Available: https://thesai.org/Downloads/Volume14No1/Paper_33-Time_Series_Forecasting_using_LSTM_and_ARIMA.pdf.

[15] S. Basak, S. Kar, S. Saha, L. Khaidem, and S. R. Dey, "Predicting the direction of stock market prices using tree-based classifiers," The North American Journal of Economics and Finance, vol. 47, pp. 552-567, 2019, doi: 10.1016/j.najef.2018.06.013.

[16] D. A. Moneim, "Egypt's stock market suspends trade on 79 shares as market dips below 5% decline benchmark," Ahram Online, 2020. [Online]. Available: https://english.ahram.org.eg/NewsContent/3/12/365371/Business/Economy/Egypt%E2%80%99s-stock-market-suspends-trade-on--shares-as-.aspx.

[17] M. Khalid, "EGX market cap jumps 2.8 percent in first week of 2024," Ahram Online, 2024. [Online]. Available: https://english.ahram.org.eg/NewsContent/3/12/515133/Business/Economy/EGX-market-cap-jumps--percent-in-first-week-of-.aspx.

[18] M. Vijh, D. Chandola, V. A. Tikkiwal, and A. Kumar, "Stock closing price prediction using machine learning techniques," Procedia Computer Science, vol. 167, pp. 599-606, 2020, doi: 10.1016/j.procs.2020.03.326.

[19] S. Abdel-Razek, "A promising year on the Stock Exchange," Ahram Online, 2024. [Online]. Available: https://english.ahram.org.eg/News/515017.aspx.

[20] EGX30 Index Methodology, Scribd, 2013. [Online]. Available: https://www.scribd.com/document/257869386/EGX-30-Methodology-en-11-02-2013.

[21] G. Singh, "Understanding Architecture of LSTM," Analytics Vidhya, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/01/understanding-architecture-of-lstm.

[22] R. H. Shumway and D. S. Stoffer, Time Series Analysis and Its Applications with R Examples, 4th ed. Springer, 2017.

[23] P. Srivastava, "Essentials of Deep Learning: Introduction to Long Short Term Memory," Analytics Vidhya, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm.

[24] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using random forest," arXiv, 2016. [Online]. Available: https://arxiv.org/pdf/1605.00003.

[25] S. Saxena, "What is LSTM? Introduction to Long Short-Term Memory," Analytics Vidhya, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory.

[26] X. Zheng, "Stock price prediction based on CNN-BiLSTM utilizing sentiment analysis and a two-layer attention mechanism," Advances in Economics, Management and Political Sciences, vol. 47, pp. 40-49, 2023, doi: 10.54254/2754-1169/47/20230369.

[27] X. Wen and W. Li, "Time series prediction based on LSTM-attention-LSTM model," IEEE Access, pp. 1-1, 2023. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10124729.