الجمهورية الجزائرية الديمقراطية الشعبية

Republique Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université Saad Dahlab Blida 1**

**Faculty of Science**
**Department of Mathmatics**

MASTER'S THESIS
SPECIALITY: MATHEMATICS
OPTION: STOCHASTIC AND STATISTIC MODELING

THEME:

# Time Series Forecasting Using Hybrid AutoRegressive Integrated Moving Average(ARIMA) and Artificial Neural Network (ANN) Model

***Presented By :***

– OUADFEUL ADEL
– OLIDIO JOSE LUIS DOS SANTOS

***Jury Members :***

| | | | |
|---|---|---|---|
| President: | Mr. REDHOUANE BOUDJEMAA | M.C.A | U.S.D.BLIDA 1 |
| Examiner : | Mr. MOHAMED BOUKHARI | M.A.A | U.S.D.BLIDA 1 |
| Supervisor: | Mr. REDOUANE FRIHI | M.C.B | U.S.D.BLIDA 1 |

2023/2024

# Thanks

We thank GOD, the Almighty, for the will, patience, and courage that He has granted us to complete this work.
We would like to express our gratitude to our supervisor, **Mr. Redouane Frihi**, for his invaluable assistance and guidance throughout this lengthy journey. We appreciate the opportunity he provided to work on such a significant topic.
We also extend our thanks to our parents, friends, and classmates for their support and understanding.
Next, we would like to thank all the members of the jury who will evaluate our work and provide us with their knowledge and constructive feedback. We want to express our gratitude to all the teachers of the mathematics department, especially to the department head **Mr. Omar TAMI** .

# Acknowledgement

I dedicated this humble and honorable work to:

**My Father:** My dear father, my strength, my idol, my hero and my pride, my first support in the pursuit of my studies and in obtaining a university diploma, to the one who gave me everything in this life , to the greatest man of my life, thank you for everything.

**My Mother:** My dear Mother, my motivation and my joy of life, who is dear to me and dear to my heart, my world and my universe, my first love and my first support in life and in obtaining a University diploma and thanks for your tireless dedication and unconditional love over the years. Thank you for everything.

**My grandparents:** Grandfather and Grandmother Tereza, Guebe, Catarina, José Kayanda and to Grandfather and Grandmother Luis, Kayaya and eva who have already passed away.

**My Brothers and Sister:** Nolasco, Tadeu, Josefina, Meury, Paula, Hortência, Hermenegildo, Domingos(M.N), Yola and to all my brothers and sisters.

I dedicate to all my family and my coursemates, without exception, and to all those who supported me in my academic journey.

**Finally:** I offer the greatest dedication and thanks to myself, who faced many pitfalls to arrive at this stage.

*OLIDIO JOSE LUIS DOS SANTOS*

# Acknowledgement

This work is dedicated to those who have fueled my journey:

**My mother**, whose unwavering support and encouragement were the fuel that propelled me through my studies.

**Dad**, your belief in me from the very beginning empowered me to reach this point.

To my brothers, **Yassine** and **Abdesslam**and **yasser**, my rocks, I wouldn't be here without you.

And to my incredible **familly**, and my **freinds** who i will never forget.

This work serves as a token of my deepest gratitude to all of you.

Additionally, I extend my thanks to everyone who knows and supports **ouadfeul adel**. Your well wishes have not gone unnoticed.

*OUADFEUL ADEL*

# ملخص

يعد المتوسط المتحرك المتكامل للانحدار الذاتي (*ARIMA*) أحد النماذج الخطية الشائعة في التنبؤ بالسلاسل الزمنية خلال العقود الثلاثة الماضية. الأنشطة البحثية الأخيرة في التنبؤ مع تشير الشبكات العصبية الاصطناعية (*ANN*) إلى أن *ANN* يمكن أن تكون بديلاً واعداً للطرق الخطية التقليدية. غالبًا ما تتم مقارنة نماذج *ARIMA* و *ANN* باستنتاجات مختلطة من حيث التفوق في التنبؤ بالأداء. في هذه الأطروحة، تم اقتراح منهجية هجينة تجمع بين نموذجي *ARIMA* و *ANN* للاستفادة من القوة الفريدة لنماذج *ARIMA* و *ANN* في النمذجة الخطية وغير الخطية. تشير النتائج التجريبية مع مجموعات البيانات الحقيقية إلى أن النموذج المدمج يمكن أن يكون وسيلة فعالة لتحسين دقة التنبؤ التي يحققها أي من النماذج المستخدمة بشكل منفصل.

# Abstract

Autoregressive Integrated Moving Average (ARIMA) is one of the popular linear models in time series forecasting during the past three decades. Recent research activities in forecasting with Artificial Neural Networks (ANN) suggest that ANN can be a promising alternative to the traditional linear methods. ARIMA models and ANN are often compared with mixed conclusions in terms of the superiority in forecasting performance. In this thesis, a hybrid methodology that combines both ARIMA and ANN models is also proposed to take advantage of the unique strength of ARIMA and ANN models in linear and nonlinear modeling. Experimental results with real datasets indicate that the combined model can be an effective way to improve forecasting accuracy achieved by either of the models used separately.

# Resumé

Autoregressive Integrated Moving Average (ARIMA) est l'un des modèles linéaires les plus populaires dans la prévision de séries chronologiques au cours des trois dernières décennies. Activités de recherche récentes en prévision avec Artificial Neural Network (ANN) suggèrent que l'ANN peut être une alternative prometteuse aux méthodes linéaires traditionnelles. Les modèles ARIMA et ANN sont souvent comparés avec des conclusions mitigées en termes de supériorité des performances de prévision. Dans cette thèse, une méthodologie hybride combinant les modèles ARIMA et ANN est également proposée pour tirer parti de la force unique des modèles ARIMA et ANN en modélisation linéaire et non linéaire. Les résultats expérimentaux avec des ensembles de données réels indiquent que le modèle combiné peut être un moyen efficace d'améliorer la précision des prévisions obtenue par l'un ou l'autre des modèles utilisés séparément.

# Abbreviation and Notations

AR $\rightarrow$ Autoregressive

MA $\rightarrow$ Moving Average

p $\rightarrow$ The number of past values included in the AR model, or the size of the Autoregressive window.

q $\rightarrow$ The number of past forecast errors included in the MA model, or the size of the moving average window.

d $\rightarrow$ The number of times the time series is differenced.

$\phi$ $\rightarrow$ AR Parameters

$\theta$ $\rightarrow$ MA Parameters

$\hat{\phi}$ $\rightarrow$ AR Estimated Parameters

$\hat{\theta}$ $\rightarrow$ MA Estimated Parameters

$\sigma$ $\rightarrow$ Standard Deviation (variance)

$\hat{\sigma}$ $\rightarrow$ Estimated Deviation

$\varepsilon$ $\rightarrow$ Error

$\rho$ $\rightarrow$ Autocorrelation

$\gamma$ $\rightarrow$ Partitial Autocorrelation

$\hat{R}$ $\rightarrow$ The rescaled Residuals

|  |  |  |
|---|---|---|
| $\hat{W}$ | $\rightarrow$ | White Noise Standard Deviation. |
| e.g., | $\rightarrow$ | For Exemple (from latin) |
| i.e., | $\rightarrow$ | This is (from latin) |
| $\nabla X_t$ | $\rightarrow$ | The differenced time series |
| PACF | $\rightarrow$ | Partial Autocorrelation |
| ACF | $\rightarrow$ | Autocorrelation |
| AIC | $\rightarrow$ | Akaike Information Criterion |
| BIC | $\rightarrow$ | Bayesian Information Criterion |
| ANN | $\rightarrow$ | Artificial Neural Networks |
| $F$ | $\rightarrow$ | Activation Function |
| $L_t$ | $\rightarrow$ | Linear Component |
| $N_t$ | $\rightarrow$ | Nonlinear Component |
| $f$ | $\rightarrow$ | Nonlinear function of preceding Residuals |
| $\hat{N}_t$ | $\rightarrow$ | Nonlinear Component Estimated |
| $\hat{L}_t$ | $\rightarrow$ | Linear Component Estimated |
| $\bar{\hat{\phi}}$ | $\rightarrow$ | Mean of the $\hat{\phi}$ |
| $MSE$ | $\rightarrow$ | Mean Squared Error |
| $\overline{MSE}$ | $\rightarrow$ | Mean of the Mean Squared Error |
| $\bar{\hat{\theta}}$ | $\rightarrow$ | Mean of the $\hat{\theta}$ |
| $BTC$ | $\rightarrow$ | Bitcoin |

# Contents

# List of Figures

# List of Tables

# Introduction

Forecasting is indispensable across finance and numerous other fields, allowing organizations and individuals to make well-informed decisions based on anticipated future conditions. In the realm of finance, accurate forecasting is critical for managing risks, efficiently allocating resources, and maximizing returns. It enables investors to predict market trends, assess the feasibility of investments, and strategize to mitigate potential losses. For businesses, forecasting aids in budgeting, financial planning, and setting achievable sales targets. Beyond finance, forecasting plays equally vital roles in diverse sectors such as supply chain management, public health, and environmental science Abu-Mostafa and Atiya (1996).

Time series forecasting is a powerful technique that relies on historical data points ordered chronologically to predict future values. By analyzing patterns and trends within the data, practitioners can make informed predictions about future outcomes. This method is particularly valuable when there is limited knowledge about the underlying data generation process or when traditional explanatory models are inadequate Chatfield (2000).

Key methodologies in time series forecasting include the autoregressive integrated moving average (ARIMA) model and artificial neural networks (ANNs). The ARIMA model is favored for its statistical rigor and the structured approach of the Box-Jenkins methodology. However, its reliance on linear relationships limits its ability to capture complex, nonlinear patterns that often characterize real-world data. On the other hand, ANNs offer flexibility by adaptively forming models based on data features, making them suitable for scenarios where theoretical guidance is lacking.

Recognizing the limitations and strengths of individual models, hybrid approaches have emerged as a strategy to enhance forecasting accuracy. Hybrid models combine the strengths of ARIMA and ANN models to leverage their complementary capabilities. By integrating these approaches, hybrid models can effectively capture both linear and non-linear patterns in time series data, thereby improving forecasting accuracy and robustness against structural changes in the data.

Empirical studies and forecasting competitions have consistently demonstrated the effectiveness of combining multiple forecasting models. This approach mitigates the risk of relying on a single model that may not adequately capture the diverse patterns present in real-world data. Moreover, the combined model approach enhances forecasting performance across various applications, from financial markets to public health planning and environmental monitoring.

In this thesis, we propose an exploration of ARIMA, ANN, and hybrid models for time series forecasting. The thesis will delve into detailed reviews of ARIMA and ANN methodologies, introduce the hybrid modeling approach in chapter 3, and present empirical results from real datasets in chapter 4. The goal is to contribute to advancing forecasting techniques by integrating complementary modeling approaches, thereby enhancing the accuracy and applicability of time series forecasting in practical scenarios.

# Chapter 1

# AutoRegresive Integrated Moving Average (ARIMA) Model

## 1.1 Introduction

Time series have already played an important role in the early natural sciences. Babylonian astronomy used time series of the relative positions of stars and planets to predict astronomical events. Observations of the planets movements formed the basis of the laws JOHANNES KEPLER discovered.

The analysis of time series helps to detect regularities in the observations of a variable and derive 'laws' from them, and/or exploit all information included in this variable to better predict future developments.

The basic methodological idea behind these procedures, which were also valid for the Babylonians, is that it is possible to decompose time series into a finite number of independent but not directly observable components that develop regularly and can thus be calculated in advance.

For this procedure, it is necessary that there are different independent factors which have an impact on the variable.

In the middle of the 19th century, this methodological approach to astronomy was taken up by the economists CHARLES BABBAGE and WILLIAM STANLEY JEVONS. The decomposition into unobserved components that depend on different causal factors, as it is usually employed in the classical time series analysis, was developed by WARREN M. PERSONS (1919). He distinguished four different components:

- A long-run development, the **trend**;
- A cyclical component with periods of more than one year, the **business cycle**;
- A component that contains the ups and downs within a year, the **seasonal cycle**;
- A component that contains all movements which neither belong to the trend nor to the business cycle nor to the seasonal component, the **residual**;

Under the assumption that the different non-observable factors are independent, their additive overlaying generates the time series which we can, however, only observe as a whole.
In order to get information about the data generating process, we have to make assumptions about its unobserved components.
The classical time series analysis assumes that the systematic components, i.e. trend, business cycle and seasonal cycle, are not influenced by stochastic disturbances and can thus be represented by deterministic functions of time. Stochastic impact is restricted to the residuals, which, on the other hand, do not contain any systematic movements.
It is therefore modelled as a series of independent or uncorrelated random variables with expectation zero and constant variance, i.e. as a pure random process. Gebhard Kirchgassner (2007)

## 1.2 Stochastic Processes

- A Stochastic Process is a collection of random variables

$$\{X_1, X_0, X_1, X_2, \ldots, X_T, \ldots\}$$

- An observed series $\{x_1, x_2, \ldots, x_T\}$ - realizations of a stochastic process.

### 1.2.1 Nonlinear and Linear time series

#### 1.2.1.1 Linear Time Series

A scalar process $X_t$ is a linear time series if it can be written as

$$X_t = \mu + \sum_{i=-\infty}^{\infty} \psi_i \varepsilon_{t-i}, \tag{1.2.1}$$

Where $\mu$ and $\psi_i$ are a sequence of constants with

$$\sum_{i=-\infty}^{\infty} {\psi_i}^2 < \infty, \quad \{\varepsilon_t\} \sim i.i.d \quad N(0, \sigma_\varepsilon^2) \tag{1.2.2}$$

i.e., $\varepsilon_t$ is a sequence of independent and identically (i.i.d.) random variables with mean zero and finite variance $\sigma_\varepsilon^2$. Such a sequence is also referred to as **strict white noise** as opposed to **weak white noise**, which is a stationary sequence of uncorrelated random variables. Obviously the requirement that $\varepsilon_t$ is i.i.d. is more restrictive than that this sequence is serially uncorrelated. Independence implies that third and higher-order non-contemporaneous moments of $\varepsilon_t$ are zero, i.e., $\mathbb{E}(\varepsilon_{t-i}\varepsilon_{t-j}) = 0 \ \forall i, j \neq 0$, and similarly for fourth and higher-order moments. When $\varepsilon_t$ is assumed to be Gaussian distributed, the two concepts of white noise coincide .

More generally, the above concepts of white noise are in increasing degree of "whiteness" part of the following classification system:

$\{\varepsilon_t\}$ is said:

**(i) Weak white noise:** If

$$\{\varepsilon_t\} \sim WN(0,\sigma_\varepsilon^2), \text{ i.e.,} \quad \mathbb{E}(\varepsilon_t)=0,$$
$$\gamma_\varepsilon(l) = \mathbb{E}(\varepsilon_t\varepsilon_{t+l}) = \begin{cases} \sigma_\varepsilon^2 & \text{if} \quad l=0, \\ 0 & \text{otherwise} \end{cases}$$

**(ii) Strict white noise:** If

$$\{\varepsilon_t\} \sim \quad i.i.d \quad N(0,\sigma_\varepsilon^2).$$

**(iii) Gaussian white noise:** If

$$\{\varepsilon_t\} \sim \quad i.i.d \quad \mathcal{N}(0,\sigma_\varepsilon^2).$$

The process $\{X_t, t \in \mathbb{Z}\}$ is said to be linear causal if $\psi_i = 0 \ \forall i < 0$, i.e., if

$$X_t = \sum_{i=1}^{\infty} \psi_i\varepsilon_{t-i}, \quad where \quad \sum_{i=1}^{\infty} {\psi_i}^2 < \infty, \quad \{\varepsilon_t\} \quad \sim \quad i.i.d \quad N(0,\sigma_\varepsilon^2) \tag{1.2.3}$$

Now suppose that $\{\varepsilon_t\} \sim WN(0,\sigma_\varepsilon^2)$ in (1.2.3). In that case the best mean square predictor may not coincide with the best linear predictor. Moreover, under this assumption, the complete probabilistic structure of $\{\varepsilon_t\}$ is not specified: Thus, nor is the full probabilistic structure of $X_t$. Also, by virtue of $\{\varepsilon_t\}$ being uncorrelated, there is still information left in it. A partial remedy is to impose the assumption that $\{X_t, t \in \mathbb{Z}\}$ is a Gaussian process, which implies that the process $\{\varepsilon_t\}$ is also Gaussian. Hence, (1.2.3) becomes de Gooijer (2017):

$$X_t = \varepsilon_t + \sum_{i=1}^{\infty} \psi_i\varepsilon_{t-i}, \quad where \quad \sum_{i=1}^{\infty} {\psi_i}^2 < \infty, \quad \{\varepsilon_t\} \quad \sim \quad i.i.d \quad \mathcal{N}(0,\sigma_\varepsilon^2) \tag{1.2.4}$$

#### 1.2.1.2 Nonlinear Time Series

**Definition 1** ***( Formal definition ).*** *A nonlinear process is any stochastic process that is not linear. To this aim, a linear process must be defined. Realizations of time-series* *processes are called time series but the word is also often applied to the generating processes.*

**Definition 2** ***( Intuitive definition )***. *Nonlinear time series are generated by nonlinear dynamic equations. They display features that cannot be modelled by linear processes: time-changing variance, asymmetric cycles, higher-moment structures, thresholds and breaks.*

**Remark 1** *It is impossible to distinguish perfectly between linear and nonlinear processes or, put it in another way, given a finite series, it is always possible to find a good description for it by means of a linear model, possibly of sufficiently high order.*

### 1.2.2 Stationarity

A stationary time series has properties that do not depend on the time at which the series is observed. In contrast, non-stationary time series have properties that vary over time (Peter J. Brockwell (2002)).

#### 1.2.2.1 Covariance - Stationary

**Definition 3** *A Stochastic process $\{X_t\}$ is covariance - stationary (weak stationary) if:*

**(i)** $E[X_t] = \mu, \quad \forall\ t$

**(ii)** $Cov(X_t, X_{t-j}) = E[(X_t - \mu)(X_{t-j} - \mu)] = \gamma(j),\ \forall\ t, j$

**Remark 2**

1. Mean is time-invariant
2. $Cov(X_t, X_{t-j}) = \gamma(j)$ (Covariance doesn't depend on t).
3. $Var(X_t) = \gamma(0)$ (variance is also constant).
4. It is weak stationarity because it only relates to the first two moments. Higher moments can be time-variant.

**Exemple 1** *Let $\{X_t\}$ with $E(X_t) = 0$, $Var(X_t) = \sigma^2$, and $E(X_t X_s) \implies$ white noise (WN)*

1. $X_t \sim i.i.d \quad N(0, \sigma^2) \implies$ *independent white noise.*
2. $X_t \sim i.i.d \quad N(0, \sigma^2) \implies$ *Gaussian white noise.* Suda (2013)

#### 1.2.2.2 Strict (Strong) Stationary

**Definition 4** *A Stochastic process $\{X_t\}$ is (strictly / strongly) stationary if for any value of $j_1, j_2, \ldots, j_n$ the joint distribution of $(X_t, X_{t+j_1}, X_{t+j_2}, \ldots, X_{t+j_n})$ depends only on the intervals separating the dates $(j_1, j_2, \ldots, j_n)$ and not on date itself (t).*

- $\forall$ T , $t_1, t_2, \ldots, t_n$: $F_X(X_{t_1}, X_{t_2}, \ldots, X_{t_n}) = F_X(X_{t_1+T}, X_{t_2+T}, \ldots, X_{t_n+T})$

  Where $F_x$ is the joint distribution function.

- If a process is strictly stationary with a finite second moment it is also covariance-stationary.

- Normality $\implies$ strong stationarity: whole distribution depends on the first two moments. Suda (2013)

### 1.2.3 Tests for Stationarity in Time Series

#### 1.2.3.1 Dickey Fuller Test

Dickey Fuller test is a statistical test that is used to check for stationarity in time series. This is a type of unit root test, through which we find if the time series is having any unit root. Santra (2023)

Unit root is a feature of time series that indicates if there is any stochastic trend in the time series that drives it away from its mean value. Presence of unit root makes a time series non-stationary and as a result it leads to difficulties in deriving statistical inferences from the time series and future predictions. Dickey Fuller test assumes a AR(1) type time series model and it is represented mathematically as,

$$X_t = \mu + \phi_1 X_{t-1} + \varepsilon_t$$

After we subtract $X_{t-1}$ from both side, we get:

$$\Delta X_t = \mu + \delta X_{t-1} + \varepsilon_t$$

where, $\phi$: Coefficient

$X_{t-1}$: Value in the time series at lag of 1

$\varepsilon_t$: is the Error component,

$SE$: is a Standard Error,

$t$: Represents a time trend (if included),

$\Delta$:The difference between the true (population) mean $\mu$ and the lower bound of your null hypothesis.

P-Value : Or probability value, is a number describing how likely it is that your data would have occurred under the null hypothesis of your statistical test. and $\tau$ Is the test statistic.

The beta ($\beta$) coefficient is the degree of change in the outcome variable for every 1-unit of change in the predictor variable. The t-test assesses whether the beta coefficient is significantly different from zero.

The test statistic formula is:

$$\tau_{\hat{\delta}} = \frac{\hat{\delta}}{SE(\hat{\delta})} \tag{1.2.5}$$

#### 1.2.3.2 Augmented Dickey Fuller(ADF) Test

Augmented Dickey Fuller(ADF) test is an extension of Dickey Fuller test for more complex models than AR(1). The primary difference between the two tests is that the ADF is utilized for a larger sized set of time series models which can be more complicated. Augmented Dickey Fuller test assumes a AR(p) type time series model and it is represented mathematically as, Santra (2023)

$$X_t = \mu + \sum_{i=1}^{p} \phi_i X_{t-1} + \varepsilon_t \tag{1.2.6}$$

After we subtract $X_{t-1}$ from both side, we get:

$$\Delta X_t = \mu + \delta X_{t-1} + \sum_{i=1}^{p} \beta_i \Delta Y_{t-1} + \varepsilon_t \tag{1.2.7}$$

ADF is the same equation as the DF with the only difference being the addition of differencing terms representing a larger time series.

The test statistic formula is:

$$\tau_{\hat{\beta}_i} = \frac{\hat{\beta}_i}{SE(\hat{\beta}_i)} \tag{1.2.8}$$

**Assumptions**

The test is conducted under following assumptions:

1. Null Hypothesis ($H_0$): There exists a unit root in the time series and it is non-stationary. Unit root = 1 or $\delta = 0$
2. Alternate Hypothesis ($H_1$): There exists no unit root in the time series and it is stationary. Unit root < 1 or $\delta < 0$

#### 1.2.3.3 Condition to reject $H_0$ and accept $H_1$

If the test statistic is less than the critical value or if the p-value is less than a pre-specified significance level (e.g., 0.05), then the null hypothesis is rejected and the time series is considered stationary.

## 1.3 ARIMA Models

### 1.3.1 Backshift and Forward operators

**Definition 5** *We define the **backshift operator** by*

$$BX_t = X_{t-1}$$

and extend it to powers $B^2X_t = B(B_{X_t}) = BX_{t-1} = X_{t-2}$, and so on. Thus,

$$B^kX_t = X_{t-k} \tag{1.3.1}$$

**Definition 6** ***Differences of order d** are defined as*

$$\nabla^d X_t = (1-B)^d X_t = X_t - X_{t-d}, \tag{1.3.2}$$

*where we may expand the operator $(1-B)^d$ algebraically to evaluate for higher integer values of d. When d = 1, we drop it from the notation (Douglas C. Montgomery (2015)).*

### 1.3.2 Autoregressive Models (AR)

**Definition 7** *$\{X_t\}$ is a **autoregressive process of order p** if :*

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t, \quad \varepsilon_t \sim WN(0,\sigma^2) \tag{1.3.3}$$

where $X_t$ is stationary, and $\phi_1$, $\phi_2$, ..., $\phi_p$ are constants ($\phi_p \neq 0$). Although it is not necessary yet, we assume that $\varepsilon_t$ is a Gaussian white noise series with mean zero and variance $\sigma^2_\varepsilon$, unless otherwise stated.

The mean of $\varepsilon_t$ (1.3.3) is zero.

If the mean, $\mu$, of $X_t$ is not zero, replace $X_t$ by $X_t$ - $\mu$ in (1.3.3),

$$X_t - \mu = \phi_1(X_{t-1} - \mu) + \phi_2(X_{t-2} - \mu) + \cdots + \phi_p(X_{t-p} - \mu) + \varepsilon_t$$

or write

$$X_t = \alpha + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t \tag{1.3.4}$$

where $\alpha = \mu(1 - \phi_1 - \cdots - \phi_p)$

Some technical difficulties, however, develop from applying that model because the regressors, $X_{t-1}, \ldots, X_{t-p}$, are random components, where as $z_t$ was assumed to be fixed. A useful form follows by using the backshift operator (1.3.1 ) to write the AR(p) model, (1.3.3), as

$$(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)X_t = \varepsilon_t \tag{1.3.5}$$

or even more concisely as

$$\phi(B)X_t = \varepsilon_t. \tag{1.3.6}$$

The properties of $\phi(B)$ are important in solving 1.3.6 for $X_t$. This leads to the following definition.

**Definition 8** *An **autoregressive operator** is defined to be*

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p. \tag{1.3.7}$$

We initiate the investigation of AR models by considering the first-order model, AR(1), given by $X_t = \phi X_{t-1} + \varepsilon_t$. Iterating backwards k times, we get

$$X_t = \phi X_{t-1} + \varepsilon_t = \phi(\phi X_{t-2} + \varepsilon_{t-1}) + \varepsilon_t$$

$$= \phi^2 X_{t-2} + \phi\varepsilon_{t-1} + \varepsilon_t$$

$$= \phi^k X_{t-k} + \sum_{j=0}^{k-1} \phi^j \varepsilon_{t-j}$$

This method suggests that, by continuing to iterate backward, and provided that $|\phi| < 1$ and $X_t$ is stationary, we can represent an AR(1) model as a linear process given by

$$X_t = \sum_{j=0}^{\infty} \phi^j \varepsilon_{t-j} \tag{1.3.8}$$

Note that $\lim_{k\to+\infty} E(X_t - \sum_{j=0}^{\infty} \phi^j \varepsilon_{t-j})^2 = \lim_{k\to+\infty} \phi^{2k} E(X_{t-k}^2) = 0$,

So the (1.3.8) exists in the mean square sense.

The AR(1) process defined by (1.3.8) is stationary with mean

$$E(X_t) = \sum_{j=0}^{\infty} \phi^j \varepsilon_{t-j}$$

and autocovariance function,

$$\gamma(h) = cov(X_{t+h}, X_t) = E[(\sum_{j=0}^{\infty} \phi^j \varepsilon_{t+h-j})(\sum_{k=0}^{\infty} \phi^k \varepsilon_{t-k})]$$

$$= E[(\varepsilon_{t+h} + \cdots + \phi^h \varepsilon_t + \phi^{h+1} \varepsilon_{t-1} + \ldots)(\varepsilon_t + \phi \varepsilon_t)] \tag{1.3.9}$$

$$= \sigma_\varepsilon^2 \sum_{j=0}^{\infty} \phi^{h+j} \phi^j = \sigma_\varepsilon^2 \phi^h \sum_{j=0}^{\infty} \phi^{2j} = \frac{\sigma_\varepsilon^2 \phi^h}{1-\phi^2}, \quad h \geq 0$$

Recall that $\gamma(h) = \gamma(-h)$, so we will only exhibit the autocovariance function for h ≥ 0. from 1.3.9, the ACF of an AR(1) is

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} = \phi^h, \quad h \geq 0, \tag{1.3.10}$$

and $\rho(h)$ satisfies the recursion

$\rho(h) = \phi\rho(h-1)$, h = 1, 2, ...,

### 1.3.3 Moving Average Models (MA)

**Definition 9** *$\{Y_t\}$ is a **moving average process of order q** if :*

$$Y_t = \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}, \quad \varepsilon_t \sim WN(0,\sigma^2) \tag{1.3.11}$$

where there are q lags in the moving average and $\theta_1, \theta_2, \ldots, \theta_q$ ($\theta_q \neq 0$) are parameters. Although it is not necessary yet, we assume that $\varepsilon_t$ is a Gaussian white noise series with mean zero and variance $\sigma^2_\varepsilon$, unless otherwise stated.

**Definition 10** *the **moving average operator** is*

$$\theta(B) = 1 + \theta_1 B + \theta_1 B^2 + \cdots + \theta_1 B^p \tag{1.3.12}$$

*Unlike the autoregressive process, the moving average process is stationary for any values of the parameters $\theta_1, \ldots, \theta_q$, details of this result are provided in 1.3.6*

## 1.4 Case Studies and Practical Examples

### 1.4.1 Real-world Applications of ARIMA

ARIMA models are widely used in various fields for time series forecasting. Examples include:

- **Stock Market Predictions:** Forecasting stock prices and indices.
- **Weather Forecasting:** Predicting temperature, precipitation, and other weather-related variables.
- **Sales Forecasting:** Predicting future sales for inventory management and planning.

### 1.4.2 Industry-specific Forecasting Examples

Different industries have specific requirements and use cases for forecasting. Some examples include:

- **Finance:** Risk management, portfolio optimization, and economic forecasting.
- **Retail:** Demand forecasting, inventory management, and supply chain optimization.
- **Healthcare:** Patient admission rates, resource allocation, and disease outbreak prediction.

### 1.4.3 ARIMA Model Description

The ARIMA (AutoRegressive Integrated Moving Average) model is a widely used statistical method for analyzing and forecasting time series data. It integrates three key components: autoregression (AR), differencing (I for "integrated"), and moving average (MA). The AR component involves using the dependency between an observation and a number of lagged observations. Mathematically, it is represented as:

$$X_t = \alpha + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t, \tag{1.4.1}$$

where $X_t$ is the value at time $t$, $\alpha$ is a constant, $\phi_i$ are the coefficients for the lagged terms, and $\epsilon_t$ is the error term at time $t$.

The I component refers to differencing the series to achieve stationarity, which involves subtracting the current observation from the previous observation. For first-order differencing, it is given by:

$$\Delta^d X_t = X_t - X_{t-1}. \tag{1.4.2}$$

The MA component models the error term as a linear combination of past error terms, formulated as:

$$X_t = \mu + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}, \tag{1.4.3}$$

where $\mu$ is the mean of the series, $\theta_j$ are the coefficients for the past error terms, and $\epsilon_{t-j}$ are the past error terms.

Combining these components, the general form of an ARIMA model is:

$$\Delta^d X_t = \alpha + \sum_{i=1}^{p} \phi_i \Delta^d X_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}, \tag{1.4.4}$$

where $p$ is the number of lag observations (autoregressive part), $d$ is the degree of differencing needed to make the series stationary, and $q$ is the size of the moving average window. This model is effective in capturing various patterns in time series data, making it a powerful tool for forecasting future values in fields like finance, economics, and environmental science.

### 1.4.4 Auto - correlation Partial Auto - correlation

**Definition 11** ***Auto - correlation function (ACF)***

$$\rho_j \equiv \frac{\gamma(j)}{\gamma(0)} \equiv j^{th} \quad autocorrelation \equiv corr(X_t, Y_{t-j})$$

- *For AR(1),* $\rho_j = \phi \rho_{j-1}$
- *ACF* $\in < -1, 1 >$.

**Definition 12** *The* ***partial autocorrelation function (PACF)*** *of a stationary process,* $X_t$*, denoted* $\phi_{hh}$*, for* $h = 1, 2, \ldots,$ *is*

$$\phi_{11} = corr(X_{t+1}, X_t) = \rho(1) \tag{1.4.5}$$

and

$$\phi_{hh} = corr(X_{t+h} - \hat{X}_{t+h}, X_t - \hat{X}_t), \quad h \geq 2. \tag{1.4.6}$$

Both $(X_{t+h} - \hat{X}_{t+h})$ and $(X_t - \hat{X}_t)$ are uncorrelated with $\{X_{t+1}, \dots, X_{t+h-1}\}$. The PACF, $\phi_{hh}$, is the correlation between $X_{t+h}$ and $X_t$ with the linear dependence of $\{X_{t+1}, \dots, X_{t+h-1}\}$ on each, removed. If the process $X_t$ is Gaussian, then $\phi_{hh} = corr(X_{t+h}, X_t | X_{t+1}, \dots, X_{t+h-1})$; that is, $\phi_{hh}$ is the correlation coefficient between $X_{t+h}$ and $X_t$ in the bivariate distribution of $(X_{t+h}, X_t)$ conditional on $\{X_{t+1}, \dots, X_{t+h-1}\}$.

**Definition 13** *$k^{th}$ - order partial autocorrelation is regression coefficient (for the population) $\phi_{kk}$ in $k^{th}$ - order autoregression*

$$X_t = c + \phi_{k1} X_{t-1} + \phi_{k2} X_{t-2} + \cdots + \phi_{kk} X_{t-k} + \varepsilon_t$$

*It measures how important is the last lag in the process (Peter J. Brockwell (2006)).*

### 1.4.5 Sample autocorrelation function (ACF)

$$\bar{y} = \frac{1}{T} \sum_{t=1}^{T} y_t$$

$$\hat{\gamma}_j = \frac{1}{T} \sum_{t=j+1}^{T} (y_t - \bar{y})(y_{t-j} - \bar{y}) \quad \text{sample auto covariance estimate}$$

$$\bar{\rho}_j = \frac{\hat{\gamma}(j)}{\hat{\gamma}(0)} \quad autocorrelation$$

### 1.4.6 partial autocorrelation function (PACF)

The **partial autocorrelation function (PACF)** of an ARMA process $\{X_t\}$ function $\alpha(.)$ defined by the equations

$$\alpha(0) = 1$$

and

$$\alpha(h) = \phi_{hh}, \quad h \geq 1,$$

where $\phi_{hh}$ is the last component of

$$\phi_h = \Gamma_h^{-1}\gamma_h$$

$\Gamma_h = [\gamma(i-j)]_{i,j=1}^{h}$, and $\gamma_h = [\gamma(1),\gamma(2),\ldots,\gamma(h)]'$.

For any set of observations $\{x_1,\ldots,x_n\}$ with $x_i \neq x_j$ for some i and j , the **sample PACF** $\alpha(h)$ is given by

$$\hat{\alpha(0)} = 1$$

and

$$\hat{\alpha(0)} = \hat{\phi_{hh}}, \quad h \geq 1,$$

where $\hat{\phi_{hh}}$ is the last component of

$$\hat{\phi_{hh}} = \hat{\Gamma}_h^{-1}\gamma_h.$$

### 1.4.7 Estimation

In this section we shall consider some of mant techniques for preliminary estimation of the parameters $\phi_1,\ldots\phi_p,\theta_1,\ldots,\theta_q$, and $\sigma^2$.
From observations $x_1,\ldots,x_n$, of the causal and invertible Gaussian $ARMA(p,q)$ process defined by:

$$\phi(B)X_t = \theta(B)\varepsilon_t, \quad \{\varepsilon_t\} \sim WN(0,\sigma^2) \tag{1.4.7}$$

### 1.4.8 Maximum likelihood

maximum likelihood is a powerful and widely applicable method for estimating the parameters of statistical models based on observed data, leveraging the principle of maximizing the likelihood of observing the data under different parameter settings.

- **Likelihood Function:** Given a set of observed data $X = \{x_1,x_2,...,x_n\}$, the likelihood function $\mathscr{L}((\phi,\theta) \mid X)$ is a function of the parameters $\phi$ , $\theta$ of the statistical model. It represents the probability of observing the data $X$ given the parameters $\phi$ and $\theta$.

- **Maximum Likelihood Estimation (MLE):** MLE aims to find the parameters values $\hat{\phi}$, $\hat{\theta}$ that maximize the likelihood function $\mathscr{L}((\phi,\theta) \mid X)$. Symbolically, it can be expressed as:

$$(\hat{\phi},\hat{\theta})_{\text{MLE}} = \arg\max_{\phi,\theta} \mathscr{L}((\phi,\theta) \mid X)$$

- **Log-Likelihood Function:** Often, the logarithm of the likelihood function, called the log-likelihood function, is used for practical reasons. This is because the logarithm is a monotonic function, meaning it preserves the order of the likelihood values while simplifying calculations:

$$\ell((\phi,\theta) \mid X) = \log \mathscr{L}((\phi,\theta) \mid X)$$

- **Finding Maximum Likelihood:** To find $(\hat{\phi},\hat{\theta})_{\text{MLE}}$, one typically takes the derivative of the log-likelihood function with respect to $\phi$ and $\theta$, sets it to zero, and solves for $\phi$ and $\theta$. In some cases, numerical optimization techniques may be employed to maximize the likelihood function when analytic solutions are not feasible.

- **Properties:** MLE is consistent, meaning that as the sample size $n$ increases, $(\hat{\phi},\hat{\theta})_{\text{MLE}}$ converges in probability to the true parameters values $\phi^*$ and $\theta^*$. It is also asymptotically efficient, meaning it achieves the Cramér-Rao lower bound under regularity conditions, making it an optimal estimator in large samples.

### 1.4.9 Yule-Walker

The Yule-Walker method is a technique used in time series analysis to estimate the parameters of an autoregressive (AR) model. It is named after Udny Yule and George Udny Yule-Walker, who developed it in the early 20th century. Eshel (2020)

Steps in the Yule-Walker Method:

- **Compute Autocovariance or Autocorrelation:** - Calculate the autocovariance function $\gamma(k)$ or autocorrelation function $\rho(k)$ for the given time series data up to a certain lag $p$. These functions describe how a time series observation at one point in time relates to its values at earlier points.

- **Yule-Walker Formula:** - The Yule-Walker equations are a set of $p$ linear equations that relate the autocovariance (or autocorrelation) values to the parameters of the autoregressive model. For an AR(p) model, they can be written as:

$$\gamma(k) = \phi_1\gamma(k-1) + \phi_2\gamma(k-2) + \ldots + \phi_p\gamma(k-p)$$

  where $\gamma(k)$ is the autocovariance function and $\phi_1, \phi_2, \ldots, \phi_p$ are the parameters to be estimated.

- **Solve the Yule-Walker Equations:** Solve these equations to find the coefficients $\phi_1, \phi_2, \ldots, \phi_p$. This typically involves expressing the equations in matrix form and solving for the coefficients using techniques such as matrix inversion or least squares estimation.

- **Estimate Variance and Check Model Adequacy:** Once the coefficients $\phi_1, \phi_2, \ldots, \phi_p$ are estimated, compute the estimated variance of the error term $\sigma^2$. Check the adequacy of the model by examining residuals (the differences between observed and predicted values) to ensure that the model adequately captures the underlying dynamics of the time series. Hence the fitted model

$$X_t = \hat{\phi_1}X_{t-1} + \ldots + \hat{\phi_p}X_{t-p} + \varepsilon_t, \quad \{\varepsilon_t\} \sim WN(0, \sigma^2)$$

Advantages and Considerations:

- **Efficiency:** The Yule-Walker method provides efficient estimation of autoregressive parameters, particularly when the autocorrelation function is used directly.

- **Assumptions:** It assumes stationarity and can be sensitive to outliers and non-stationarity in the data.
- **Model Selection:** Choosing the appropriate lag $p$ for the autoregressive model is crucial and often requires model selection criteria (e.g., AIC, BIC) to determine the best-fitting model.
- **Application:** The Yule-Walker method is widely used in fields such as econometrics, finance, and signal processing for modeling and forecasting time series data. It provides a mathematical framework to understand the dependencies within a time series and make predictions based on its past values.

### 1.4.10 Diagnostic Checking

Typically, the goodness of fit of a statistical model to a set of data is judged by comparing the observed values with the corresponding predicted values obtained from the fitted model. If the fitted model is appropriate, then the residuals should behave in a manner that is consistent with the model (Jianqing Fan (2003)).

When we fit an ARMA(p, q) model to a given series we determine the maximum likelihood estimators $\hat{\phi},\hat{\theta}$ and $\hat{\sigma}^2$ of the parameters $\phi,\theta$ and. $\sigma^2$. In the course of this procedure the predicted values $\hat{X}_t(\hat{\phi},\hat{\theta})$ of $X_t$ based on $X_1,\ldots,X_{t-1}$ are computed for the fitted model. The residuals are then defined, by:

$$\hat{W}_t = (X_t - \hat{X}_t(\hat{\phi},\hat{\theta}))/(r_{t-1}(\hat{\phi},\hat{\theta}))^{1/2}, \quad t = 1,\ldots,n \tag{1.4.8}$$

If we were to assume that the maximum likelihood ARMA(p, q) model is the true process generating $\{X_t\}$, then we could say that $\{\hat{W}_t\} \sim WN(0,\hat{\alpha}^2)$. However, to check the appropriateness of an ARMA(p, q) model for the data we should assume only that $X_1,\ldots,X_n$ are generated by an ARMA(p, q) process with unknown parameters $\phi,\theta$ and. $\sigma^2$, whose maximum likelihood estimators are $\hat{\phi},\hat{\theta}$ and $\hat{\sigma}^2$, respectively. Then it is not true that $\{\hat{W}_t\}$ is white noise. Nonetheless $\hat{W}_t, t = 1,\ldots,n$ should have properties that are similar to those of the white noise sequence

$$W_t(\phi,\theta) = (X_t - \hat{X}_t(\phi,\theta))/(r_{t-1}(\phi,\theta))^{1/2}, \quad t = 1,\dots,n$$

Moreover, $W_t(\phi,\theta)$ approximates the white noise term in the sense that $E(W_t(\phi,\theta)Z_t)^2 \to 0 \quad as \quad t \to \infty$. Consequently, the properties of the residuals $\{\hat{W}_t\}$ should reflect those of the white noise sequence $\{Z_t\}$ generating the underlying ARMA(p,q) process. In particular, the sequence $\{\hat{W}_t\}$ should be approximately

**(i)** uncorrelated if $\{Z_t\} \sim WN(0,\alpha^2)$.

**(ii)** independent if $\{Z_t\} \sim iid \quad N(0,\alpha^2)$

**(iii)** normally distributed if $\{Z_t\} \sim N(0,\alpha^2)$

The **rescaled residuals** $\hat{R}_t, t = 1,\dots,n$ are:

$$\hat{R}_t = \hat{W}_t/\hat{\alpha} \tag{1.4.9}$$

Where $\hat{W}_t$ is white noise standard deviation.
If the fitted model is appropriate, the rescaled residuals should have properties similar to those of a WN(0, 1) sequence or of an iid(0,1) sequence if we make the stronger assumption that the white noise $\{Z_t\}$ driving the ARMA process is independent white noise.

### 1.4.11 Model selection

#### 1.4.11.1 Order Selection

Once the data have been transformed (e.g., by some combination of Box–Cox and differencing transformations or by removal of trend and seasonal components) to the point where the transformed series $\{X_t\}$ can potentially be fitted by a zero-mean ARMA model, we are faced with the problem of selecting appropriate values for the orders $p$ and $q$.

It is not advantageous from a forecasting point of view to choose p and q arbitrarily

large.
Fitting a very high order model will generally result in a small estimated white noise variance, but when the fitted model is used for forecasting, the mean squared error of the forecasts will depend not only on the white noise variance of the fitted model but also on errors arising from estimation of the parameters of the model.
These will be larger for higher-order models. For this reason we need to introduce a "penalty factor" to discourage the fitting of models with too many parameters.

Many criteria based on such penalty factors have been proposed in the literature, since the problem of model selection arises frequently in statistics, particularly in regression analysis. Robert H. Shumway (2011)

- Assume we can reject iid, i.e. PACF and ACF show some significant lags. How to determine which model is correct?
- But we have many models we can't really discriminate between just by looking.

#### 1.4.11.2 Box-Jenkins Approach

Matching model with actual data

- Transform data to "appear" covariance stationary
    - take logs (natural)
    - differences
    - detrend
- Examine the sample ACF and PACF
- Estimate ARMA models
- Perform diagnostic analysis to confirm that model is consistent with data

### 1.4.12 Information Criteria

Use information criteria to compare models with different $p$ and $q$ values. Lower values indicate better models (Jianqing Fan (2003)).

#### 1.4.12.1 Akaike Information Criterion (AIC)

$$AIC(p,q) = \ln(\sigma^2) + \frac{1}{T}2(p+q),$$

where the first term is responsible for the fit of the model and the second one is a penalty for number of parameters.

#### 1.4.12.2 Schwarz (Bayesian) Information Criterion(BIC)

$$BIC(p,q) = \ln(\sigma^2) + \frac{1}{T}\ln T2(p+q),$$

where now penalty is related to sample size.

### 1.4.13 Point Forecasts

Point forecasts involve predicting future values using the fitted ARIMA model. Given an ARIMA model, we can generate forecasts for future time steps.

$$\hat{X}_{t+h|t} = E(X_{t+h}|X_t, X_{t-1}, \ldots, X_1)$$

Here, $\hat{X}_{t+h|t}$ represents the forecasted value at time $t+h$ based on information available up to time $t$.

### 1.4.14 Forecast Intervals

Forecast intervals provide a range within which future observations are expected to fall with a certain probability. They account for the uncertainty in the forecasts.

The forecast interval is typically given by:

$$\hat{X}_{t+h|t} \pm z \times \sigma_{\hat{X}}$$

where $z$ is the critical value from the standard normal distribution corresponding to the desired confidence level, and $\sigma_{\hat{X}}$ is the standard error of the forecast.

# Chapter 2

# Arificial Neural Networks (ANN) Model

## 2.1 Introduction

Artificial neural networks (ANN's) are computer systems inspired by the way our brains work (2.1) , which were originally proposed by McCulloch and Pitts (1943) and Metropolis et al. (1953). They became popular in the 1980s because of advances in computing and learning techniques. One important development was the backpropagation learning method, introduced by Rumelhart et al. (1986).

This method adjusts the connections between neurons to make the network's output match what's expected. It led to ANN's being used widely in various fields to understand complex problems and make intelligent decisions. Neural Networks are defined by Hamid and Habib (2014) as a versatile and dependable field made up of primary and dominating elements-neurons.

While reducing the impact of the damaged neuron on the outcome, they permit the coupling of diverse brain inputs and outputs, benefits, and inhibition.

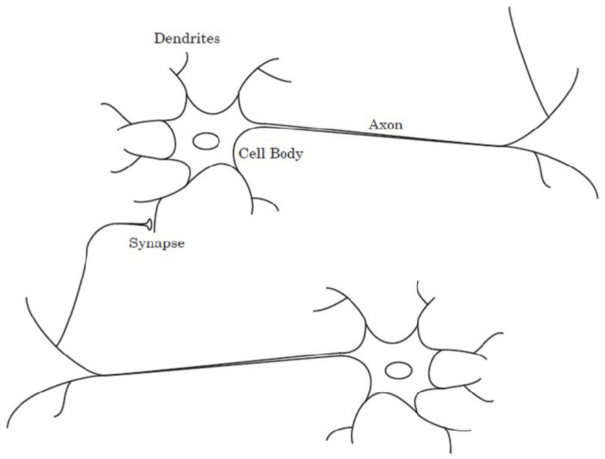


**Figure 2.1:** Diagrammatic Illustration of Biological Neurons

Neural networks , as stated by Mileris and Boguslauskas (2011) , are mostly useful for analyzing large amounts of data and establishing sets of criteria . The system operates on the same concept as an individual . However , humans aren't always able to comprehend and make sense of this volume of data.

As stated by Kliěstik (2013), artificial neural networks are among the most widely used methods available today, with applications in a variety of domains. Ashoori and Mohammadi (2011) mention the social and natural sciences, neurology, linguistics, technology, cognitive science, and business environment in addition to artificial intelligence.
These fields can be used, for example, to estimate expenses, firm value, or bankruptcy prediction. According to Altun et al. (2007), ANN is one of the most appealing approaches in operational research in the field of informatics; its real-world applications make significant use of contemporary technology.

## 2.2 ANN Model

The Artificial Neural Network (ANN) consists of several essential components that collectively enable them to process information and make predictions. Neurons, the basic computational units, receive inputs and apply transformations through weighted connections.
These weights, dynamically adjusted during training, govern the strength of connections

and shape the network's learning. Activation functions introduce non-linearity, enhancing the model's capacity to capture complex relationships in data. Organized into layers, including input, hidden, and output layers, ANN's exhibit hierarchical processing.
The architecture, determined by the arrangement and connectivity of neurons, plays a crucial role in defining the network's capabilities.
Additionally, biases, loss functions, optimization algorithms, learning rates, and back-propagation contribute to the robust learning and adaptation of ANN's, making them powerful tools in diverse domains of artificial intelligence.

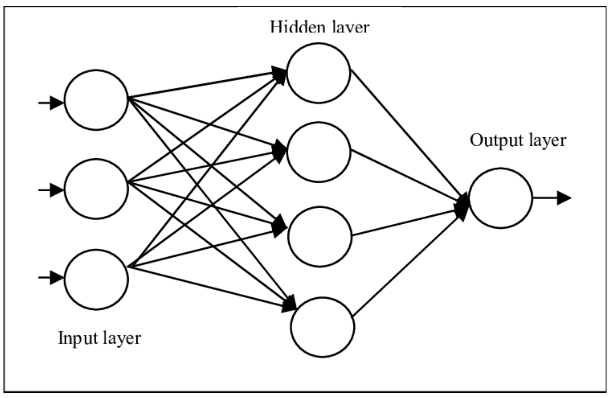


**Figure 2.2:** Architecture of Artificial Neural Networks model

The model is characterized by a network of three layers of simple processing units connected by acyclic links. The relationship between the output ($X_1$) and the inputs ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) has the following mathematical representation :

$$X_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j F\left(w_{0j} + \sum_{i=1}^{p} \beta_{ij} X_{t-i}\right) + \epsilon_t \tag{2.2.1}$$

where $\alpha_j (j = 0, 1, 2, \dots, q)$ and $w_{ij}$ $(i = 0, 1, 2, \dots, p, j = 1, 2, \dots, q)$ are the model parameters often called the connection weights, $p$ is the number of input nodes and $q$ is the number of hidden nodes, $F$ is the Activation Funcion.

Hence ,The ANN model of (2.2.1) in fact performs a nonlinear functional mapping from the past observations ($X_{t-1}$, $X_{t-2}$,..., $X_{t-p}$) to the future value $X_t$, i.e.,

$$X_t = f(X_{t-1}, X_{t-2}, X_{t-3}, \dots, X_{t-p}, w) + \epsilon_t \tag{2.2.2}$$

where w is a vector of all parameters and $f$ is a function determined by the network structure and connection weights. Thus, the ANN is equivalent to a nonlinear autoregressive model.

## 2.2.1 Artificial Neurons

A fundamental component of each artificial neural network is the artificial neuron. (2.3) represents an artificial neuron with its inputs, weights, Activation function, bias, and outputs.

Its structure and operations are based on the study of a biological neuron, which is the fundamental unit of biological neural networks, or systems, that comprise the brain, spinal cord, and peripherajl ganglia. (2.3) represents an artificial neuron with its inputs, weights, Activation function, bias, and outputs. These similarities in design and functionality can be seen.

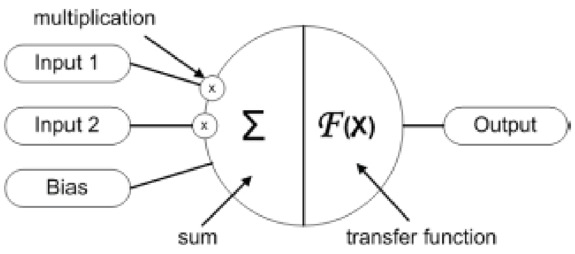


**Figure 2.3:** Artificial Neuron

When a biological neuron receives information through a dendrite, the soma analyzes it before sending it along via an axon. When it comes to artificial neurons, the information enters the system through weighted inputs (each input can be multiplied by a weight on its own). Next, the artificial neuron's body adds up the weighted inputs, adds bias, and uses a Activation function to "process" the total. Ultimately, the information digested is sent via output(s) by an artificial neuron. The mathematical explanation of the artificial neuron model below demonstrates the benefit of its simplicity:

### 2.2.2 Input layer

The input layer is the first step in the complex information processing process (Figure 2.2), The input layer consists of artificial neurons and is responsible for transforming raw input data into a format the network can understand, In this layer, every neuron takes on the crucial duty of representing a particular feature or input variable, which encodes the core of the properties of the data into the architecture of the neural network. Aligning the number of neurons with the dimensionality of the input data is the fundamental idea that drives the structure of the input layer and guarantees that all of the dataset's subtleties are accurately recorded and later handled.

### 2.2.3 Hidden layer

Hidden layers in neural networks sit between the input and output layers. They play a crucial role in helping the network understand complex patterns in data. Each neuron in a hidden layer takes inputs from the previous layer, applies weights and biases to these inputs, and then passes the result through an activation function. This process allows the network to learn and recognize patterns that are not straightforward or linear. By transforming data in this way, hidden layers enable neural networks to solve more complex problems and make accurate predictions.

### 2.2.4 Output layer

The output layer represents the final stage where the network's computations culminate in generating predictions or outcomes. This layer consolidates the information processed across previous layers into actionable insights. Its configuration, including the number of neurons, is tailored to match the complexity of the intended output. Through a combination of weighted summations and activation functions, the output layer transforms complex computations into interpretable outputs, such as class labels in classification tasks or continuous values in regression.

### 2.2.5 Activation Functions

In a neural network, the most important components are its net inputs. These units are transformed into an output result known as the activation of the unit using the application of a function known as the Activation function, threshold function, or transfer function, which is a scalar to scalar transformation.

The primary purpose of a Activation function is to determine the output of a neuron based on its input (Sharma et al. (2017)). Some of the commonly used activation functions in artificial neural networks are:

1. Linear Activation Function
2. Sigmoid Activation Function
3. Hyperbolic Tangent Activation Function
4. Rectified Linear Unit (ReLU) Activation Function

#### 2.2.5.1 Linear Activation Function

Sharma et al. (2017). The input and the linear activation function have a direct proportionality. Since the binary step function lacks an $x$ component, its primary flaw was that it had a zero gradient. A linear function can be utilized to take it out. It can be described as:

$$F(x) = ax$$

The value of variable $a$ can be any constant value chosen by the user.

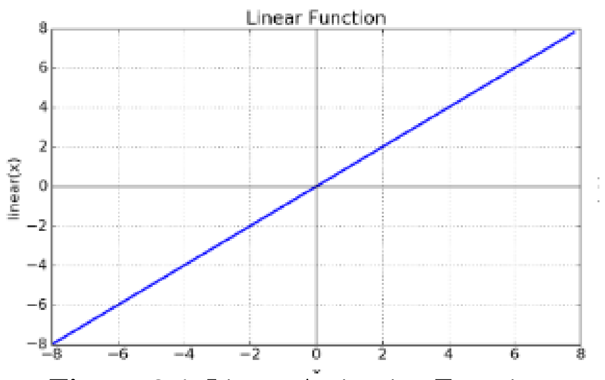


**Figure 2.4:** Linear Activation Function

Using a linear function has no advantage because the neural network would not reduce error because the gradient would remain constant during the iteration. Furthermore, the data will not allow the network to recognize intricate patterns. For this reason, linear functions are appropriate for straightforward tasks and situations requiring interpretability Sharma et al. (2017).

#### 2.2.5.2 Sigmoid Activation Function

The sigmoid function stands as one of the most commonly employed activation functions due to its non-linear nature. By transforming input values into a range between 0 and 1, it enables the network to capture complex relationships Sharma et al. (2017). Mathematically, the sigmoid function can be expressed as :

$$F(x) = \frac{1}{e^{-x}}$$

Furthermore, the sigmoid function lacks symmetry about zero, resulting in output values of neurons sharing the same sign. This asymmetry can be addressed by scaling the sigmoid function Sharma et al. (2017).

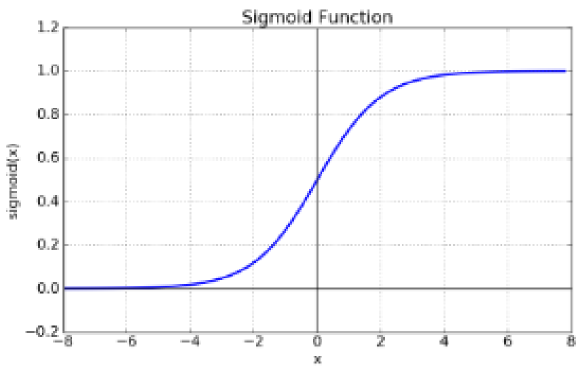


**Figure 2.5:** Sigmoid Activation Function

#### 2.2.5.3 Hyperbolic Tangent Activation Function

The hyperbolic tangent function, also known as the tanh function, serves as an alternative to the sigmoid function. Unlike the sigmoid function, the tanh function is symmetric around the origin. This symmetry results in varied signs of outputs from previous layers, providing diverse inputs for the subsequent layers in the neural network Sharma et al. (2017). Mathematically, the tanh function can be expressed as:

$$F(x) = \tanh(x)$$

The tanh function is both continuous and differentiable, producing values within the range of $-1$ to 1. In contrast to the sigmoid function, the tanh function exhibits a steeper gradient. This characteristic makes tanh preferable over the sigmoid function, as its gradients are not confined to a specific direction and it is centered around zero.

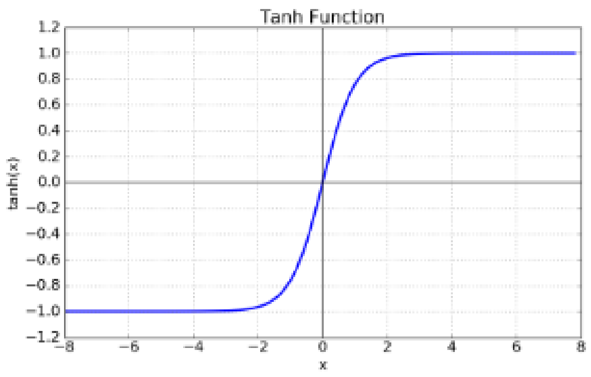


**Figure 2.6:** Hyperbolic Tangent Activation Function

#### 2.2.5.4 Rectified Linear Unit(ReLU) Activation Function

ReLU (Rectified Linear Unit) fig (2.7) is considered more efficient compared to other activation functions because it doesn't activate all neurons simultaneously. Instead, it selectively activates a certain number of neurons at a time Sharma et al. (2017).

In some cases, the gradient value can be zero, leading to non-updating of weights and biases during the backpropagation step in neural network training.

Mathematically, the ReLU function can be expressed as:

$$F(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

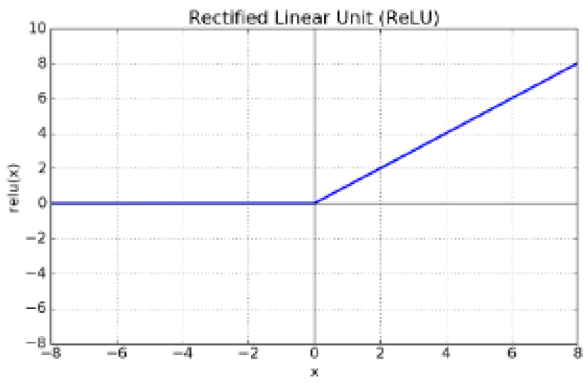


**Figure 2.7:** Rectified Linear Unit(ReLU) Activation Function

### 2.2.6 Weights

Weights within an artificial neural network (ANN) represent numerical values linked with the connections among neurons across various layers of the network Figure (2.8). Each connection between neurons carries an associated weight, indicating the magnitude and direction (positive or negative) of the impact one neuron exerts on another. As input signals traverse the network, they undergo multiplication by these weights, collectively influencing the ultimate output of the network.

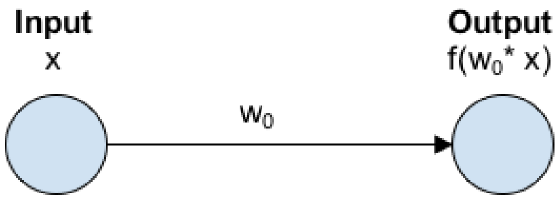


**Figure 2.8:** weights betwin nuerons

## 2.3 Unsupervised and Supervised learning models

Several different types of ANN's have been developed, but two main categories can be easily recognized, according to their learning process (Lek and Park (2008)) :

### 2.3.1 Unsupervised learning:

- The model is not provided with the correct results during the training.
- Can be used to cluster the input data in classes on the basis of their statistical properties only.
- Cluster significance and labeling.
- The labeling can be carried out even if the labels are only available for a small number of objects representative of the desired classes.

### 2.3.2 supervised learning

- Training data includes both the input and the desired results.
- For some examples the correct results (targets) are known and are given in input to the model during the learning process.
- These methods are usually fast and accurate.
- Have to be able to generalize: give the correct results when new data are given in input without knowing a priori the target.

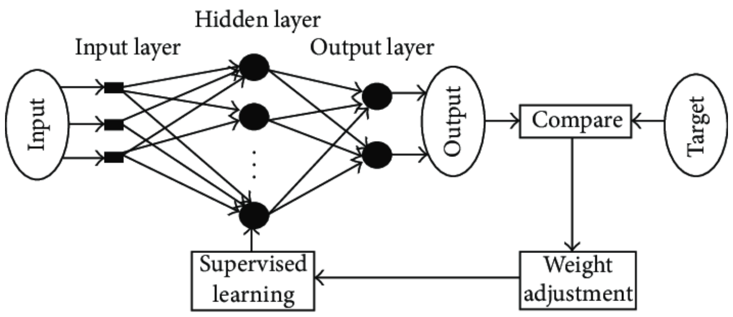


**Figure 2.9:** supervised architecture of ann

## 2.4 Types of Neural Networks

### 2.4.1 Multilayer Perceptron (MLP)

Multilayer Perceptrons (MLPs) are structured as a network of basic neurons organized into layers .Typically, a Multilayer Perceptron (MLP) is structured with an input layer comprising source neurons, at least one hidden layer consisting of computational neurons, and an output layer composed of computational neurons. Input signals are propagated forward through the network, layer by layer, in a sequential manner (Ali et al. (2011)). Mathematically, the MLP can be represented as:

$$y = F\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.4.1}$$

where $w$ stands for the weight vector, $x$ for the input vector, $b$ for the bias, and $F$ for the activation function. By acting as a squashing function, the activation function stops the network's expansion from increasing.

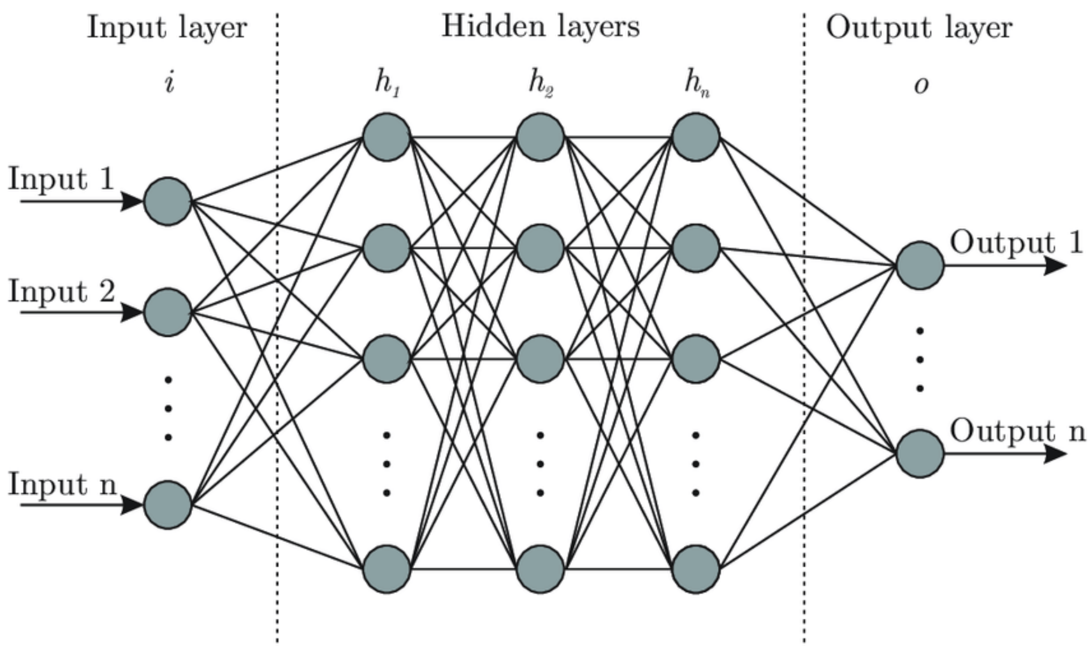


**Figure 2.10:** MLP Structure

### 2.4.2 Radial basis function

In RBF networks, the input layer outputs are computed by measuring the distance between the network inputs and the centers of the hidden layer. The subsequent layer, known as the linear hidden layer, produces outputs that are weighted combinations of the input layer outputs. Each neuron in the hidden layer is associated with a parameter vector referred to as the center. Consequently, a comprehensive representation of the network can be formulated as follows Howlett and Jain (2001):

$$\hat{y} = \sum_{i=1}^{n} w_{ij}\phi(\|x - c_i\|) + b_j \tag{2.4.2}$$

Typically, the Euclidean distance serves as the norm, while the radial basis function is commonly defined as a Gaussian function. Specifically, the Gaussian function is expressed as follows:

$$F(r) = \exp(-\alpha_i.\|x - c_i\|^2) \tag{2.4.3}$$

where

| | |
|---|---|
| $I$ | Number of neurons in the hidden layer |
| $J$ | Number of neurons in the output layer |

| | |
|---|---|
| $w_{ij}$ | Weight of the $i^{th}$ neuron and $j^{th} output$ |
| $F$ | Radial basis function |
| $\alpha_i$ | Spread parameter of the $i^{th}$ neuron |
| $x$ | Input data vector |
| $c_i$ | Center vector of the $i^{th}$ neuron |
| $b_j$ | Bias value of the output $j^{th}$ neuron |
| $\hat{y}_j$ | Network output of the $j^{th}$ neuron |

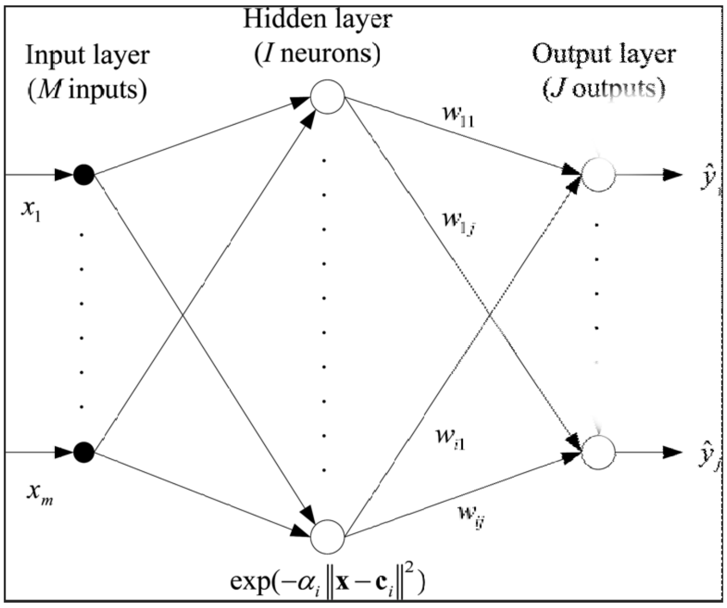


**Figure 2.11:** RBF Network architecture

### 2.4.3 Recurent Neural Netwwork

A Recurrent Artificial Neural Network (RNN) is a type of neural network architecture characterized by its recurrent topology, which allows information to flow not only in the forward direction but also backwards. This allows networks to analyze and learn sequences, such as recognizing and reproducing sequences or predicting temporal patterns (Bullinaria (2013)). (Figure 2.12) illustrates a small Fully Recurrent artificial neural network, showcasing the intricate interconnections among its artificial neurons. The fundamental topology of a recurrent artificial neural network is the fully recurrent artificial network, where every artificial neuron is directly connected to every other neuron in all directions. Other variants of recurrent artificial neural networks, such as Hopfield, Elman, Jordan, bi-directional networks, and others, are special cases derived from the basic fully recurrent architecture (Suzuki (2011)).

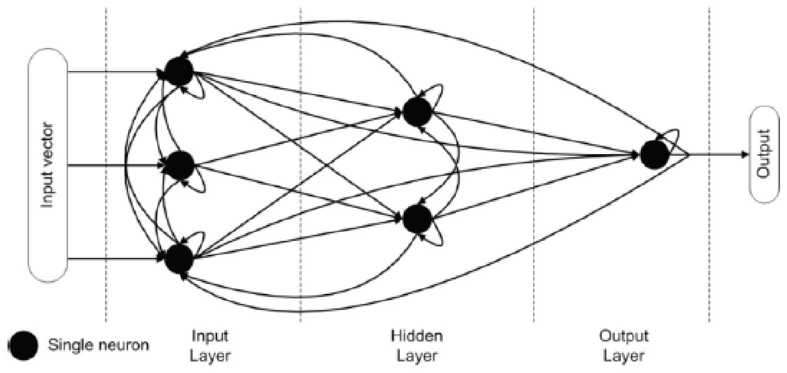


**Figure 2.12:** Recurrent artificial neural network

## 2.5 Identification of ANN model

### 2.5.1 The Backpropagation Algorithm

One of the most popular NN algorithms is back propagation algorithm. Rojas (2013) stated that there were four primary steps in the BP algorithm. The required adjustments are computed using the back propagation approach once the network's weights have been randomly selected. The algorithm can be decomposed in the following four steps:

#### 2.5.1.1 Feed-forward computation

Input to hidden layer:

$$z_j = \sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}$$

$$a_j = F(z_j)$$

Hidden layer to output:

$$z_k = \sum_{j=1}^{m} w_{jk}^{(2)} a_j + b_k^{(2)}$$

$$\hat{y}_k = F(z_k)$$

#### 2.5.1.2 Back propagation to the output layer

Output layer error:

$$\delta_k = (\hat{y}_k - y_k) \cdot F'(z_k)$$

#### 2.5.1.3 Back propagation to the hidden layer

Hidden layer error:

$$\delta_j = F'(z_j) \sum_k w_{jk}^{(2)} \delta_k$$

#### 2.5.1.4 Weight updates

Update rule for weights (and biases):

$$w_{ij}^{(1)} \leftarrow w_{ij}^{(1)} - \beta \cdot \delta_j \cdot x_i uu$$

$$b_j^{(1)} \leftarrow b_j^{(1)} - \beta \cdot \delta_j$$

$$w_{jk}^{(2)} \leftarrow w_{jk}^{(2)} - \beta \cdot \delta_k \cdot a_j$$

$$b_k^{(2)} \leftarrow b_k^{(2)} - \beta \cdot \delta_k$$

Where:

- $z_j$ and $z_k$ are the weighted sums before activation functions,
- $a_j$ and $\hat{y}_k$ are the activations (outputs),
- $F$ is the activation function,
- $F'$ is the derivative of the activation function,
- $w_{ij}^{(1)}, b_j^{(1)}, w_{jk}^{(2)}$, and $b_k^{(2)}$ are weights and biases,

- $\beta$ (beta) is the learning rate,
- $\delta_j$ and $\delta_k$ are the error terms for hidden and output layers, respectively.

## 2.6 Evaluating Forecast Accuracy

### 2.6.1 Metrics

Evaluating the accuracy of the forecasts is crucial. Common metrics include:

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |X_i - \hat{X}_i|$$

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (X_i - \hat{X}_i)^2$$

- **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\text{MSE}}$$

- **Mean Absolute Percentage Error (MAPE):**

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n} \left| \frac{X_i - \hat{X}_i}{X_i} \right|$$

- **Forecast Efficiency:**

$$\text{Forecast Efficiency} = 1 - \frac{\text{MSE of forecasts}}{\text{MSE of actual values}}$$

### 2.6.2 Visualizing Forecasts

Visualizing the forecasts alongside the actual values can provide insights into the model's performance. Common visualizations include:

- **Line Plots:** Plotting actual vs. forecasted values over time.
- **Prediction Intervals:** Including confidence intervals in the plots to show the uncertainty in forecasts.

# Chapter 3

# Hybrid Forecasting.

Hybrid models, also known as ensemble models or combined models, integrate multiple forecasting methods to achieve better prediction accuracy compared to individual models. They can combine statistical methods like ARIMA, multiple regression, and exponential smoothing with artificial intelligence (AI) techniques such as fuzzy inference systems, genetic algorithms, and neural networks. By merging diverse methodologies, hybrid models seek to capture the nuances of complex time series data and improve forecasting performance.

## 3.1 Hybrid Models

Time series data can contain nonlinear and linear component.So, hybrid methods using both nonlinear and linear models are better and more accurate than individual models for forecasting time series data. (Alsuwaylimi (2023)) Various hybrid methods which exist in the literature include the following approach: Given a time series data, ARIMA model is directly modeled on the data. The residuals obtained from ARIMA model is considered as a nonlinear component, and this nonlinear data is modeled using ANN in different methods. Some such hybrid models considered in this thesis are those of Additive, Khashei-Bijari, and multiplicative hybrid methods which are shown below (Alsuwaylimi (2023)).

## 3.2 Types Of Hybrid Models

Hybrid models combine different forecasting techniques to leverage the strengths of each approach and improve overall prediction accuracy. These models can be classified into several types based on how they integrate different methods. Some common types of hybrid models include:

### 3.2.1 Linear and Nonlinear Models

These hybrids combine both linear and nonlinear forecasting techniques. For example, ARIMA may be used as the linear component, while artificial neural networks (ANN's) serve as the nonlinear component.

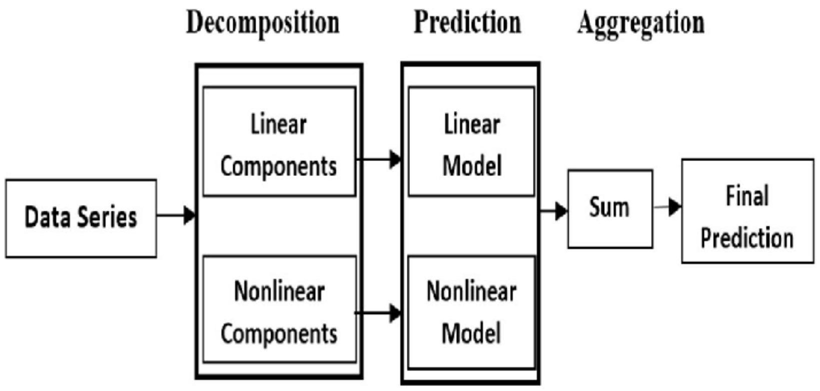


**Figure 3.1:** linear-nonlinear-series-hybrid-models

### 3.2.2 Weighted Averaging Models

In this approach, forecasts from individual models are combined using weighted averages. The weights assigned to each model can be based on historical performance or determined through optimization techniques.

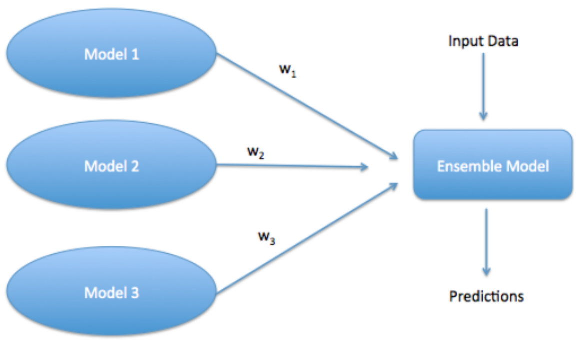


**Figure 3.2:** Weighted Averaging Models

### 3.2.3 Hybrid with various models

A set methods combine multiple individual models to generate a single prediction. Techniques such as bagging, boosting, and stacking are commonly used to combine forecasts from different models Sagi and Rokach (2018).

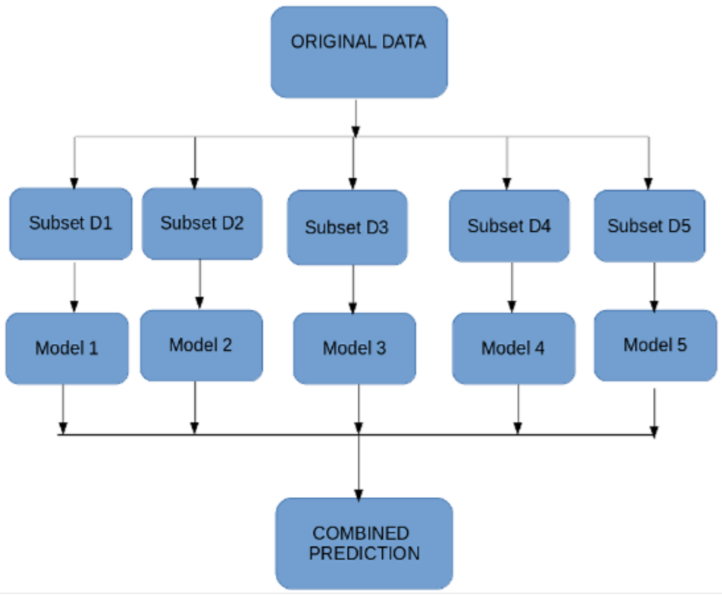


**Figure 3.3:** With various models

## 3.3 Hybridizing Linear and Nonlinear Patterns

Zhang (2003) Stated that both ARIMA and ANN models have found success within their respective linear and nonlinear domains. However, neither model is universally applicable to all forecasting scenarios. While ARIMA models excel at capturing linear trends and patterns, they may fall short when dealing with complex nonlinear relationships.

Conversely, ANN's are adept at modeling nonlinear data but may struggle with linear problems, leading to mixed results. Blindly applying ANN's to any dataset without considering their limitations can result in suboptimal performance.
Given the challenge of fully understanding the characteristics of real-world data, a hybrid approach that combines the strengths of both linear and nonlinear modeling capabilities emerges as a promising strategy for time series forecasting.
By leveraging the complementary aspects of ARIMA and ANN models, hybrid methodologies can capture a broader range of underlying patterns, thereby enhancing forecasting accuracy and robustness.

It may be reasonable to consider a time series to be composed of a linear 'autocorrelation structure and a nonlinear component. That is,

$$Y_t = L_t + N_t \tag{3.3.1}$$

where
$L_t =$ : The linear component.
$N_t$ : The nonlinear component.
These two components have to be estimated from the data. First, we let ARIMA to model the linear component, then the residuals from the linear model will contain only the nonlinear relationship.

### 3.3.1 Additive Hybrid Model

(Zhang (2003)) proposed a hybrid model of ARIMA and ANN for the additive model. which assumes the time-series $y_t$ summing the linear and nonlinear components as:

$$X_t = L_t + N_t \tag{3.3.2}$$

where Lt represents the linear component and Nt the nonlinear component.
During the first phase of the proposed hybrid approach, an ARIMA model is applied to

the linear component of time series, which is assumed to be $\{y_t, t = 1, 2, \ldots\}$, and a series of forecasts are generated, namely $\hat{L}_t$. By comparing the actual value $y_t$ with the forecast value $\hat{L}_t$ of the linear component, we can obtain a series of nonlinear components, which are defined to be $\{e_t\}$.

$$e_t = X_t - \hat{L}_t \tag{3.3.3}$$

Where

$$\begin{aligned} \hat{L}_t =& \mu + \hat{\hat{\phi}}_1 X_{t-1} + \hat{\phi}_2 X_{t-2} + \cdots + \hat{\phi}_p X_{t-p} \\ & - \hat{\theta}_1 \varepsilon_{t-1} - \hat{\theta}_2 \varepsilon_{t-2} - \cdots - \hat{\theta}_q \varepsilon_{t-q} \end{aligned} \tag{3.3.4}$$

Thus, a nonlinear time series is obtained.

The second phase is concerned with modeling the nonlinear component of the specified time series in an ANN model.
The trained ANN's model is responsible for making a series of forecasts of nonlinear components, denoted by $\hat{N}_t$, which are based on the previously deduced nonlinear time series $\{e_t\}$ values as the inputs. That is, the ANN's time-series forecasting model is a nonlinear mapping function, as shown below:

$$e_t = f(e_{t-1}, e_{t-2}, e_{t-n}) + \varepsilon_t \tag{3.3.5}$$

Where $f$ is a nonlinear function of preceding residuals, while $\varepsilon_t$ is the component white-noise in the ANN modeling.

The second phase can be seen as a process of error correction of time series prediction in the ANN's model on the basis of $ARIMA$ model. the combined forecast is given by Equation below:

$$X_t = \hat{L}_t + \hat{N}_t \tag{3.3.6}$$

### 3.3.2 Khashei and Bijari Hybrid Model

Khashei and Bijari (2010) proposed a hybrid prediction model for time series, suggest-

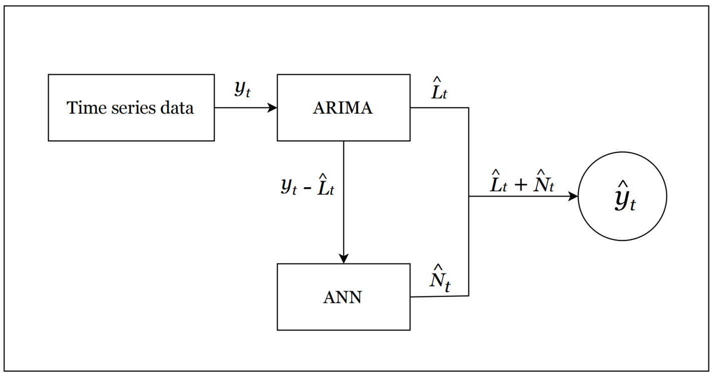


**Figure 3.4:** Additive Hybrid Model.

ing that y can be represented as a combination of linear and nonlinear components, as described in their equation.(3.3.7):

$$y_t = F(L_t + N_t) \tag{3.3.7}$$

This model relates to the Additive model that assumes the composition of linear and nonlinear components in any time-series data. The linear components are modelled by using ARIMA while the residuals assumed to contain nonlinear component cannot be modelled by a linearity approach. In the next stage in which linear forecast and the computed nonlinear residuals are combined, this model is differentiated from the Additive model. This model claims an additive relationship between linear and nonlinear parts. The relation might be underestimated and degrade the performance. Valid nonlinear patterns in the residuals of ARIMA model may further not be guaranteed and therefore assumptions under this approach lowers the performances in another situation. The above reasons permit consideration of linear and nonlinear components as functional time-series shown in the Equation 3.3.8:

$$\hat{y}_t = F(\hat{L}_t + \hat{N}_t) \tag{3.3.8}$$

Where $\hat{L}_t$ is the linear component which is modelled by ARIMA, $\hat{N}_t$ is the nonlinear one modelled by ANN. In modelling nonlinear component, multi-layer component is used by introducing past original values, present forecast of linear component, and past error data with residuals as inputs to ANN shown in the Figure ?? which simply summaries the Khashei-Bijari's model. ANN does not only get residuals as input but also takes past linear forecasts and actual values. Therefore, it can undermine the strict assumption that is additive association between linear and nonlinear components. The performed experiments support this approach at outperforms the forecasting accuracy of Additive model in a number of applications.

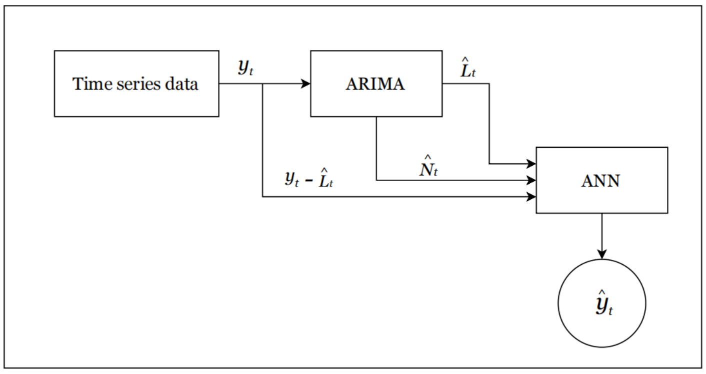


**Figure 3.5:** Khashei and Bijari's Hybrid Model.

### 3.3.3 Multiplicative Model :

Wang, introduced a multiplicative hybrid-model for predicting time-series data that was contrasted with additive hybrid models. In this model, a given time series data is the product of a linear and a non-linear time series unlike the additive nature assumed by Zhang's model and the Khashei-Bijari's hybrid models. $L_t$ and $N_t$ are the linear and nonlinear parts of Equation 3.3.9:

$$y_t = L_t N_t \tag{3.3.9}$$

The time series $y_t$ in Equation 3.3.9 is modeled using ARIMA additively. The quotient of $y_t$ by the forecasts $\hat{L}_t$ results to the nonlinear component given by Equation 3.3.10.

$$e_t = y_t / \hat{L}_t \tag{3.3.10}$$

The nonlinear component $N_t$ is modeled and forecasted using ANN and the final model forecasts obtained using the product of the nonlinear forecasts $\hat{N}_t$ and linear forecasts $\hat{L}_t$ as illustrated in Equation 3.3.11:

$$y_t = \hat{L}_t * \hat{N}_t \tag{3.3.11}$$

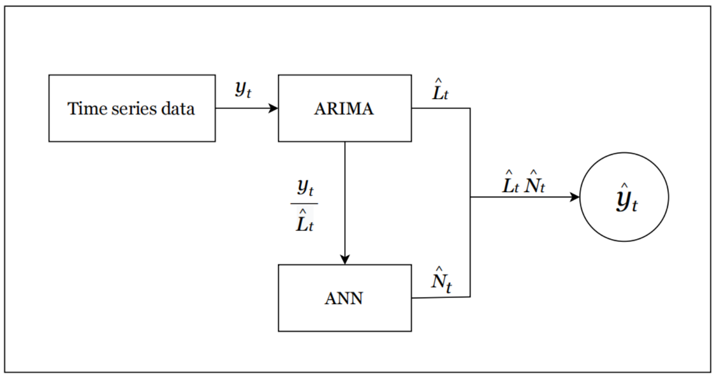


**Figure 3.6:** Multiplicative Hybrid Model.

In summary, the proposed hybrid modeling is conducted in two phases. Initially, an ARIMA model is identified, and the corresponding parameters are evaluated, that is, an ARIMA model is constructed. As a result, the nonlinear components are computed based on this model. In the second phase, a neural network model deals with the nonlinear components.

# Chapter 4

# Application

## 4.1 Introduction

The application part is divided into two steps. The first is to use the simulated data to compare the three prediction models (AutoRegresive Moving Average, Artificial Nueral Network and ARMA-ANN). The second step is to predict real data using all three models, namely the Bitcoin exchange rate.
All codes used are in Python.

## 4.2 Model of Simulation

We have simulated 100 samples of size 1000 and 2000 of the AR(1) and ARMA(1,1) models.

$(i)$ $AR(1) : X_t = \varphi X_{t-1} + \varepsilon_t.$

$(ii)$ $ARIMA(1,1) : X_t = \varphi X_{t-1} + \varepsilon_t - \varepsilon_{t-1}$

### 4.2.1 Order Identification by AIC and BIC

Uising AIC and BIC Percentage to select the best order of the Model.

| simulation model | model | AIC Percentage | BIC Percentage |
|---|---|---|---|
| | AR(1) | 70 | 100 |
| AR(1) | ARIMA(1,1) | 30 | 0 |
| | AR(1) | 10 | 0 |
| ARIMA(1,1) | ARIMA(1,1) | 90 | 100 |

**Table 4.1:** comparing tow models using AIC and BIC percentage of 100 sample

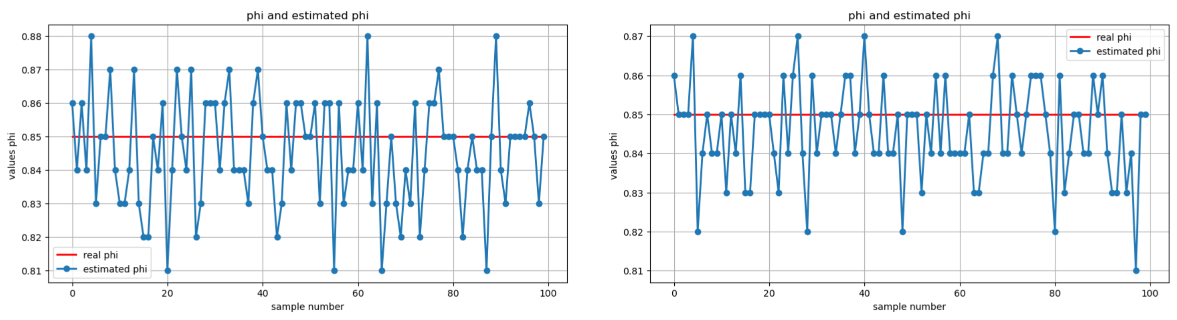
### 4.2.2 Parameters Estimation

The estimated parameter using 100 sample of size 1000 and 2000 of the simulated data.

| Observations | Sample Size | $\varphi$ | $\bar{\hat{\phi}}$ |
|---|---|---|---|
| 100 | n=1000 | 0.85 | 0.845 |
| 100 | n=2000 | 0.85 | 0.848 |

**Table 4.2:** Comparing parameters and estimated parameters of 100 samples of size 1000 and 2000

A graph of the estimated parameters using 100 Observations of 1000 and 2000 sample size of the simulated data



**(a)** 100 Observations of samples of size 1000 **(b)** 100 Observations of samples of size 2000

**Figure 4.1:** 100 samples of size 1000 and 2000

### 4.2.3 Mesure of Performance of the AR Parameters

In the table below we compare the mean square error of AR parameter of 100 Observations of 1000 and 2000 sample of the simulated data

| Observations | Sample Size | $\overline{MSE}$ |
|---|---|---|
| 100 | n=1000 | 0.00026 |
| 100 | n=2000 | 0.00014 |

**Table 4.3:** Compare the Mean Square Error of 100 samples of size 1000 and 2000

### 4.2.4 AR(1) Mesure Performance for Predictions

In the table below we compare the mean square error of AR(1) parameter of 100 Observations of 1000 and 2000 sample of the simulated data

| Observations | Sample Size | $\overline{MSE}$ |
|---|---|---|
| 100 | n=1000 | 1.0741 |
| 100 | n=2000 | 1.0599 |

**Table 4.4:** Compare the Mean Square Error of 100 samples of size 1000 and 2000

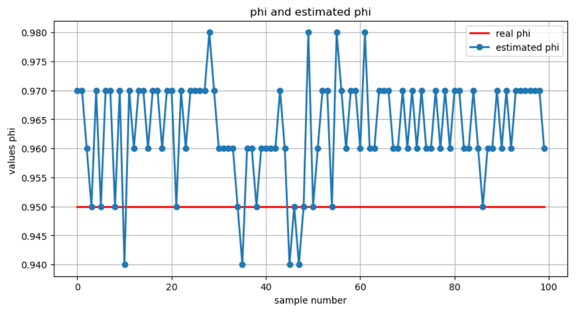
### 4.2.5 ARIMA Parameters Estimation

The estimated values of the parameters of the ARIMA model using 100 Observations of the Sample of size 1000 and 2000 of the simulated data.

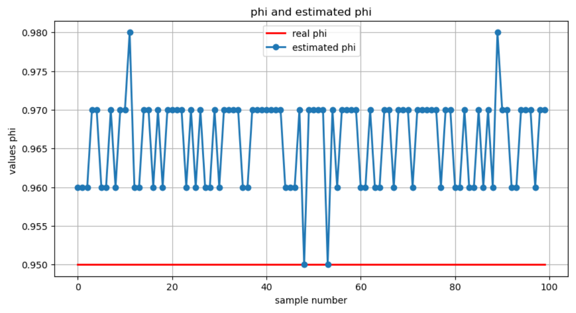
| Observations | Sample Size | $\varphi$ | $\bar{\hat{\phi}}$ | $\theta$ | $\bar{\hat{\theta}}$ |
|---|---|---|---|---|---|
| 100 | n=1000 | 0.95 | 0.96329 | -0.3 | -0.2517269 |
| 100 | n=2000 | 0.95 | 0.9661 | -0.3 | -0.2513155 |

**Table 4.5:** 100 samples of size 1000 and 2000

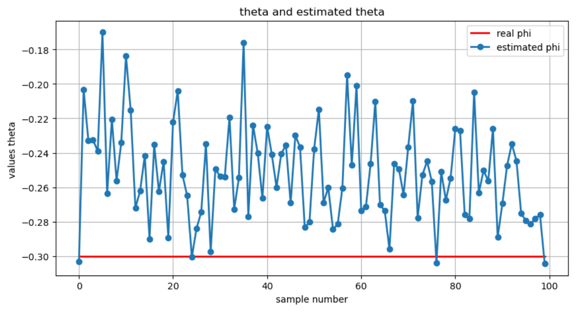
In this graph, we consider the analysis of ARIMA parameters from 1000 to 2000, of 100 Observations.The data are simulated



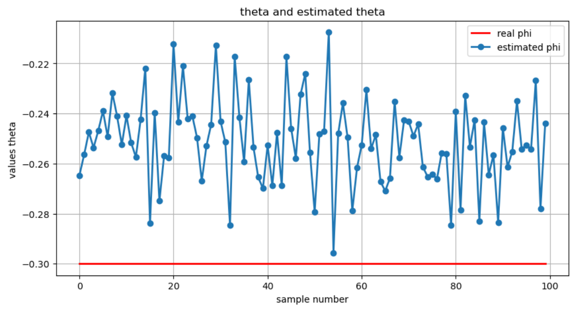
**(a)** Estimated $\phi$ of 100 Observations of samples of size 1000



**(b)** Estimated $\phi$ of 100 Observations of samples of size 2000



**(c)** Estimated $\theta$ of 100 samples of size 1000



**(d)** stimated $\theta$ of 100 samples of size 2000

**Figure 4.2:** Graphs showing e stimated parameters of 100 Observations of samples of size 1000 and 2000

### 4.2.6 Mesure Performance of the ARIMA parameters

The comparison of the mean square error of ARIMA parameters of 100 Observations of 1000 and 2000 sample of the simulated data

| parameters | Observations | sample size | $\overline{MSE}$ |
|---|---|---|---|
| $\phi$ | 100 | 1000 | 0.0031478 |
| | 100 | 2000 | 0.00028 |
| $\theta$ | 100 | 1000 | 0.0031478 |
| | 100 | 2000 | 0.002667 |

**Table 4.6:** 100 samples of size 1000 and 2000

### 4.2.7 Mesure Performance of the ARIMA(1, 0, 1) Model

The comparison of the mean square error of ARIMA Models of 100 Observations of 1000 and 2000 sample of the simulated data

| Models/Order | Observations | sample size | $\overline{MSE}$ |
|---|---|---|---|
| ARIMA(1, 0, 1) | 100 | n=1000 | 1.05746 |
| ARIMA(1, 0, 1) | 100 | n=2000 | 1.09362 |

**Table 4.7:** Comparison of 2 ARIMA(1, 0, 1) models

### 4.2.8 Mesure Performance of the ANN Model

The comparison of the mean square error of ANN Models of 100 Observations of 1000 and 2000 sample of the simulated data

| Models | Observations | sample size | Activation Function | $\overline{MSE}$ |
|---|---|---|---|---|
| ANN | 100 | n=1000 | linear | 1.81789 |
| ANN | 100 | n=1000 | tanh | 1.63226 |
| ANN | 100 | n=2000 | linear | 1.06435 |
| ANN | 100 | n=2000 | tanh | 1.07068 |

**Table 4.8:** Comparison of 4 ANN models

### 4.2.9 Mesure Performance of the Hybrid Model

The comparison of the mean square error of Hybrid Models of 100 Observations of 1000 and 2000 sample of the simulated data

| Models | Observations | sample size | Activation Function | $\overline{MSE}$ |
|---|---|---|---|---|
| Hybrid | 100 | n=1000 | linear | 1.06111 |
| Hybrid | 100 | n=2000 | linear | 1.09269 |
| Hybrid | 100 | n=1000 | tanh | 1.06639 |
| Hybrid | 100 | n=2000 | tanh | 1.08902 |

**Table 4.9:** Comparison of 4 hybrid models

## 4.3 Datasets

A cryptocurrency (crypto-currency) is a digital currency designed to work as a medium of exchange through a decentralized computer network using cryptography to secure transactions and control the creation of new units.
Unlike traditional currencies, cryptocurrencies operate independently of any central authority, such as a government or bank, making them immune to government interference or manipulation. Cryptocurrencies typically utilize blockchain technology, a distributed ledger enforced by a network of computers, to maintain transparency, security, and the integrity of transactions.

## 4.4 DATA Description

This study focuses on Bitcoin, a widely-known digital currency celebrated for its decentralized structure. Data was gathered from Yahoo Finance spanning January 1, 2020, to June 13, 2024. The dataset includes daily opening, high, low, and closing prices, adjusted for corporate actions, as well as trading volume. We ensured data accuracy through rigorous cleaning, addressing errors and outliers.
Using Python with Pandas and Matplotlib, we organized and visualized the data, enabling a detailed analysis of Bitcoin's market trends and fluctuations over the specified period.

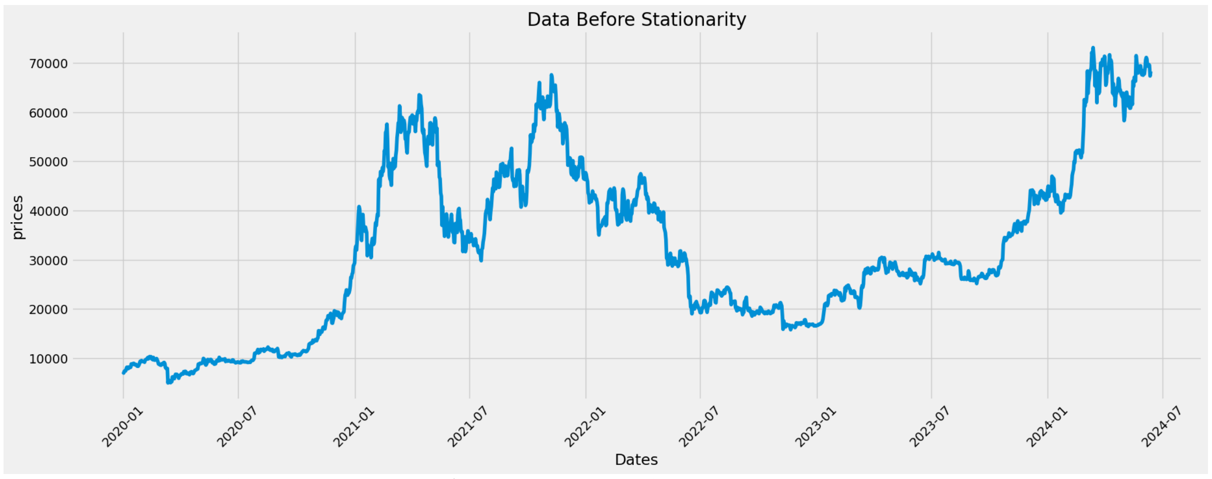


**Figure 4.3:** Bitcoin's price curve: illustrating its ups and downs

### 4.4.1 ADF test

To prepare the data for analysis, we first checked its stationarity using the Augmented Dickey Fuller (ADF) test. Stationarity refers to the data's properties remaining constant over time. The test revealed that the data exhibited non-stationarity in two ways: its variance was unstable, and its average value fluctuated over time. To address these issues, we applied suitable transformations and differencing techniques. These methods aim to stabilize the variance and achieve stationarity in terms of the data's mean, making it more appropriate for subsequent analysis.

| Test | 0 Step diffrencing | 1 step diffrencing |
|---|---|---|
| p-value | 0.781 | 0 |

**Table 4.10:** Table Shows the Results of ADF Test

From the result in Table 4.10, it can be seen that the ADF test and get p-value = 0.781, so the data is not stationary in the mean. To make the data stationary, we use once differencing. Next we check stationary again, it can be seen that the ADF test result get p-value = 0.0, that means it's stationary in mean fig (**??**).

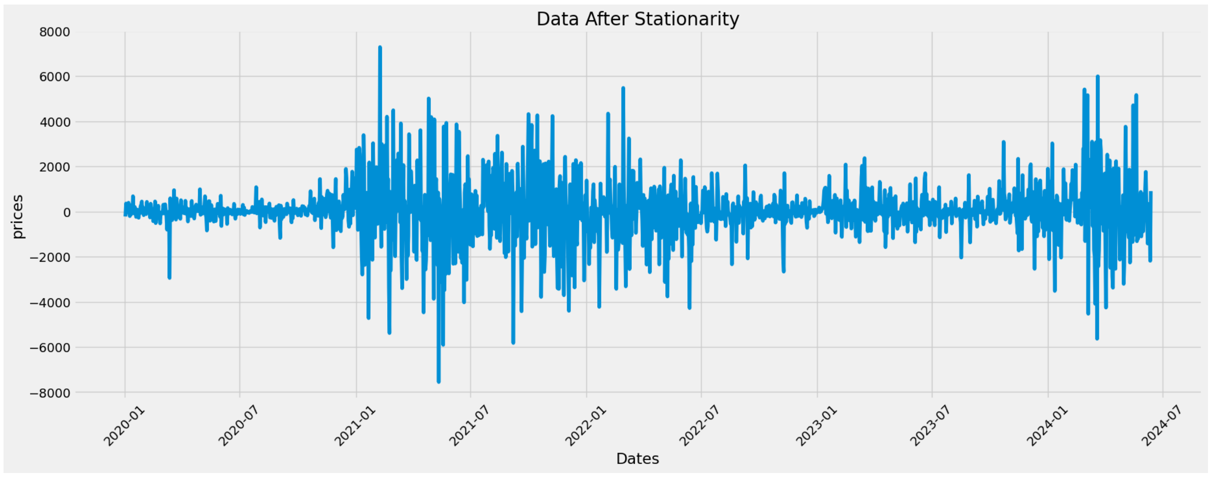


**Figure 4.4:** Stationary Bitcoin price curve after differencing.

### 4.4.2 Split DATA

To train the ARIMA model, we divided the data into training and test sets. Specifically, 97% of the data was allocated for training, while 3 % was reserved for testing. This division allows us to train the model on a majority of the data and then evaluate its performance on unseen data to assess its predictive capabilities..

| Data | Data Shape | Train Set(size) | Test Set (size) |
|---|---|---|---|
| BTC | 1625 | 1575 | 50 |

**Table 4.11:** Shapes of Data, Train and Test

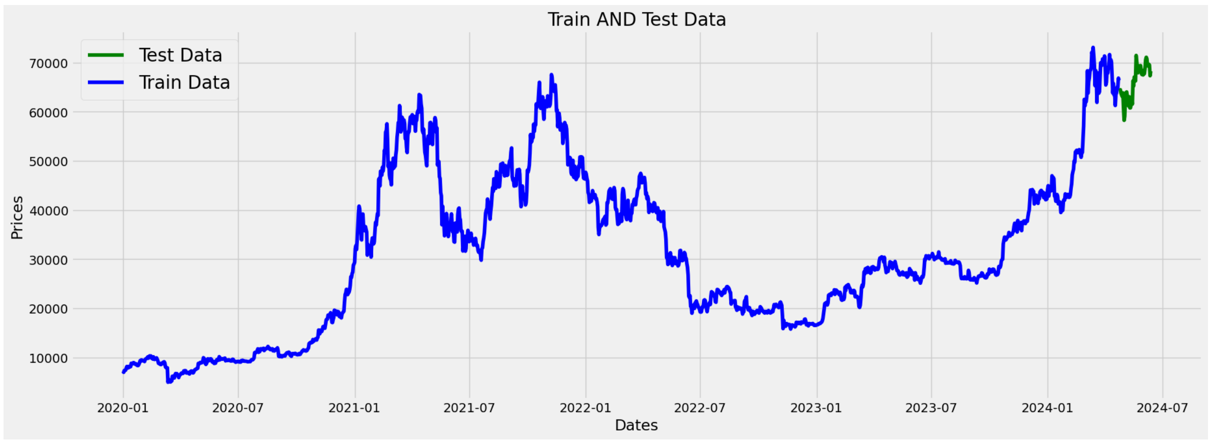
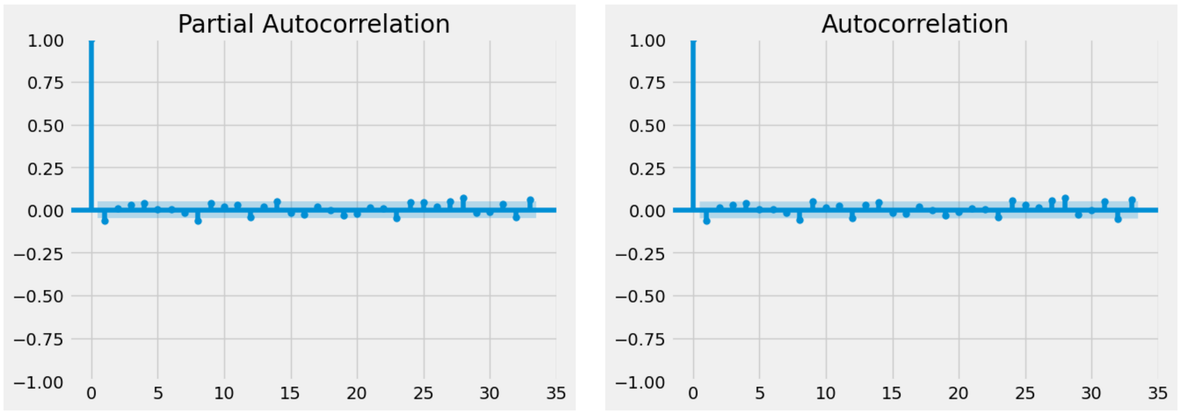


**Figure 4.5:** Bitcoin's price curve:illustrating Train and Test Parts

## 4.5 ARIMA Model Order Selection

### 4.5.1 ACF and PACF plots

From the training data, we determined the ARIMA model's order by analyzing the Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF) plots.



**(a)** pacf plot **(b)** acf plot

From the ACF and PACF plot, we get MA(0) and AR(1).

### 4.5.2 AIC and BIC Criteria

The best model that has smaller AIC and BIC because of the number of parameters is the smallest. Different models associated with accuracy criteria are listed in the following Table.

| ORDER | AIC | BIC |
|---|---|---|
| **(1,1,0)** | **27695.6859** | **27706.4700** |
| (1,1,1) | 27697.5693 | 27713.7454 |
| (2,1,0) | 27697.4522 | 27713.6283 |
| (2,1,1) | 27697.3584 | 27718.9265 |

**Table 4.12:** Comparing 4 models using AIC and BIC

Table (4.12) indicates that the suitable model for daily Bitcoin data is ARIMA (1,1,0).

## 4.6 Comparison of ARIMA Models Using Performance Metrics

In this section, we compare different ARIMA models using metrics like RMSE, and MAPE to assess their accuracy in forecasting daily Bitcoin data. By evaluating these metrics across various ARIMA configurations, we aim to identify the model that best captures the patterns and fluctuations in the dataset, ensuring reliable predictions for future observations.

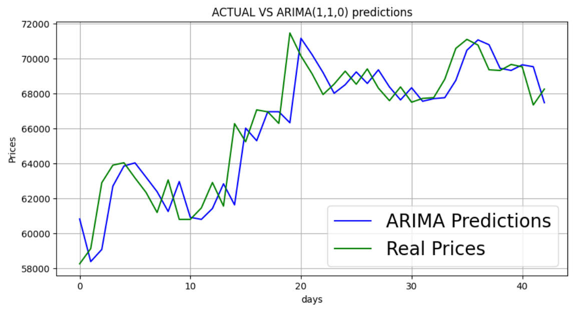
| ARIMA | RMSE | MAPE |
|---|---|---|
| **(1,1,0)** | **1611.48890** | **0.01812232** |
| (1,1,1) | 1612.25493 | 0.01814332 |
| (2,1,0) | 1613.45829 | 0.01817957 |
| (2,1,1) | 1621.01979 | 0.01839802 |

**Table 4.13:** comparing 4 models of arima

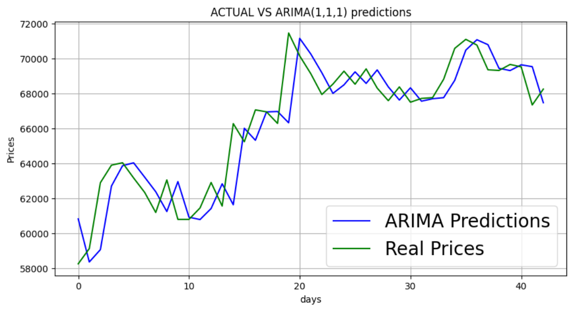
Table (4.13) indicates that the suitable model for daily Bitcoin data is ARIMA (1,1,0).

### 4.6.1 ARIMA Forecasts

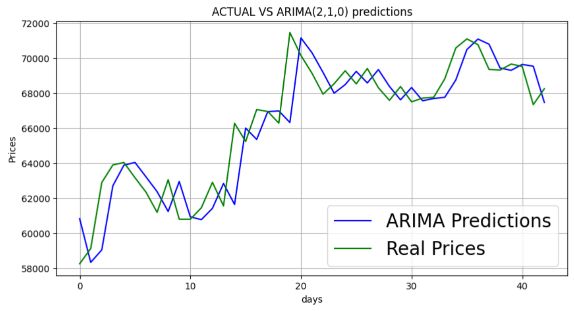
The forecasts from four distinct ARIMA models were plotted against actual daily Bitcoin prices (4.9). Each plot illustrates the models' predictions and their alignment with observed price movements over time. These visual comparisons allow for an evaluation of how effectively each ARIMA model captures Bitcoin's price volatility and overall trend.



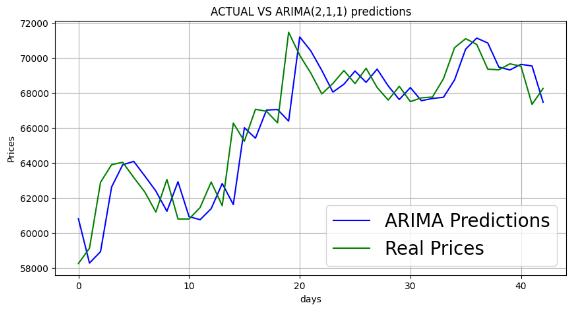
**(a)** ARIMA(1,1,0) predictions vs actual



**(b)** ARIMA (1,1,1) predictions vs actual



**(c)** ARIMA (2,1,0) predictions vs actual



**(d)** ARIMA (2,1,1) predictions vs actual

**Figure 4.7:** Daily Prices of BTC Forecasts with different ARIMA

## 4.7 ANN Model

### 4.7.1 Compare of tow ANN models using Performance Metrics

In the table (4.14), we compare two ANN models with linear and tanh activation functions, using metrics like RMSE and MAPE to assess their accuracy in forecasting daily Bitcoin data.

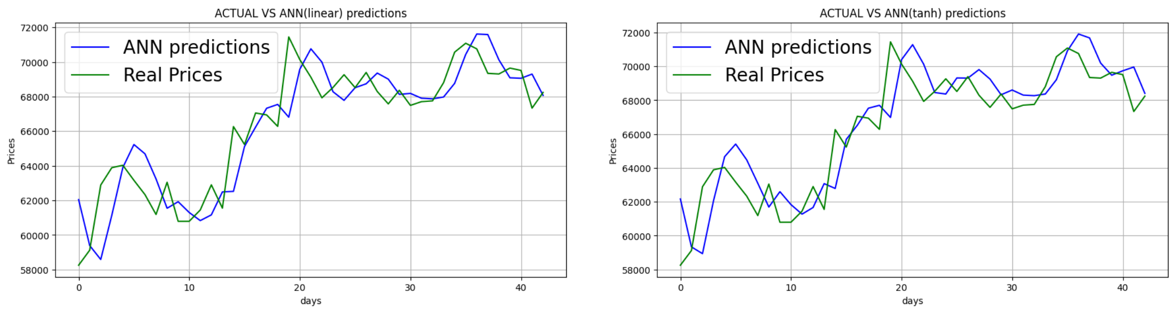
| ANN model | activation function | RMSE | MAE | MAPE |
|---|---|---|---|---|
| ANN | tanh | 2071.27812 | 0.02439363 | |
| **ANN** | **linear** | **1735.94799** | **0.01994308** | |

**Table 4.14:** The Result of Performance Metrics Of Tow ANN Models

We observed differing performance outcomes.The model with the linear activation function exhibited better results in terms of accuracy and precision metrics compared to the tanh activation model.This outcome highlights the significance of activation function selection in optimizing model performance.

### 4.7.2 ANN Forecasts

The forecasts of two ANN models with the linear activation and tanh activation function of the actual daily Bitcoin were plotted. Each plot illustrates the models' predictions and their alignment with observed price movements over time.These visual comparisons allow for an evaluation of how effectively each ANN model captures Bitcoin's price volatility and overall trend.



**(a)** linear ANN **(b)** tanh ANN

**Figure 4.8:** Daily Prices of BTC Estimated with ANN

## 4.8 Hybrid Models

In the table below we did compare four Hybrid models with linear and four Hybrid models with tanh activation functions, using metrics like RMSE and MAPE to assess their accuracy in forecasting daily Bitcoin data.

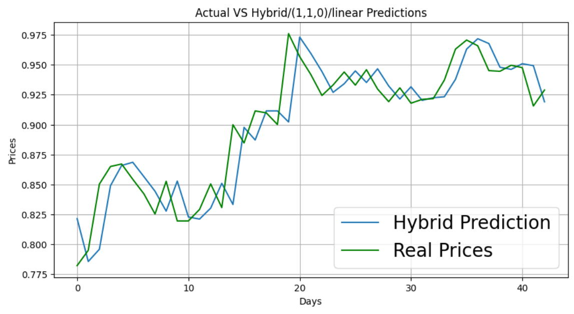
| HYBRID /ARIMA order | activation function | RMSE | MAPE |
|---|---|---|---|
| **Hybrid/(1,1,0)** | tanh | **1611.48676** | **0.018122319** |
| Hybrid/(1,1,1) | tanh | 1612.25293 | 0.018143312 |
| Hybrid/(2,1,0) | tanh | 1613.45520 | 0.018179551 |
| Hybrid/(2,1,1) | tanh | 1621.01900 | 0.018398023 |
| Hybrid/(1,1,0) | linear | 1611.48740 | 0.018122320 |
| Hybrid/(1,1,1) | linear | 1612.25288 | 0.018143312 |
| Hybrid/(2,1,0) | linear | 1613.45256 | 0.01817953 |
| Hybrid/(2,1,1) | linear | 1623.89193 | 0.01843089 |

**Table 4.15:** Comparing Different hybrid models

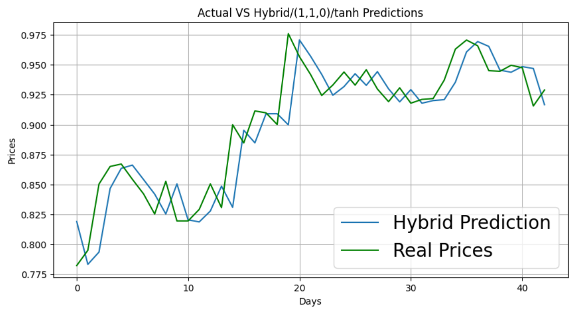
Observing different performance outcomes.We can see that the hybrid/(1,1,0) model with the linear and hybrid/(1, 1, 0) model with the tanh activation function exhibited better results in terms of accuracy and precision metrics compared to the others models.

## 4.9 Hybrid Forecast

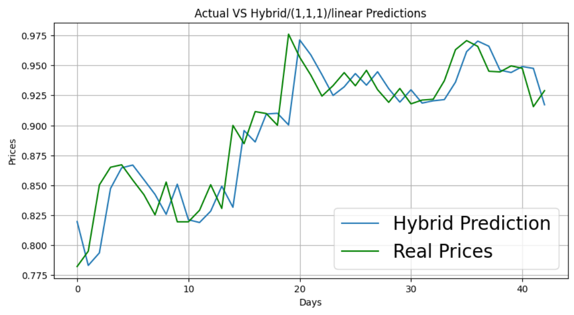
The forecasts of the Four Hybrid models with the linear activation and the Four Hybrid models with tanh activation function of the actual daily Bitcoin were plotted.
Each plot illustrates the models' predictions and their alignment with observed price movements over time. These visual comparisons allow for an evaluation of how effectively each Hybrid model captures Bitcoin's price volatility and overall trend.



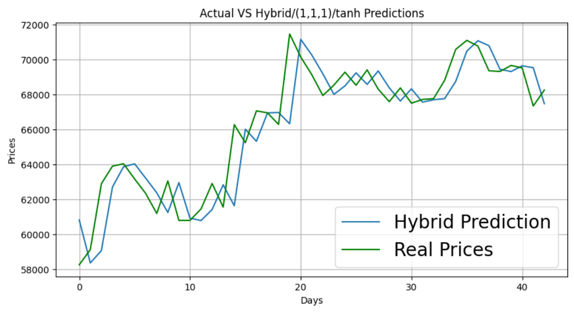
**(a)** Hybrid/(1,1,0) with linear Activation Function predictions vs actual



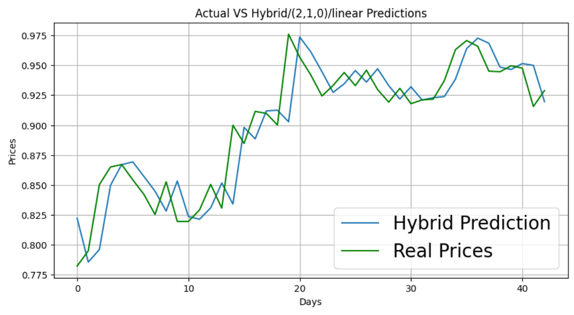
**(b)** Hybrid /(1,1,0) with tanh Activation Function predictions vs actual



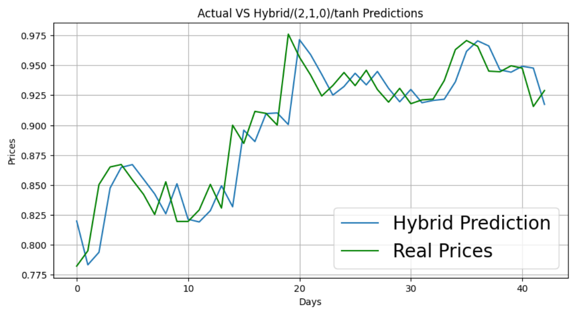
**(c)** Hybrid/(1,1,1) with linear Activation Function predictions vs actual



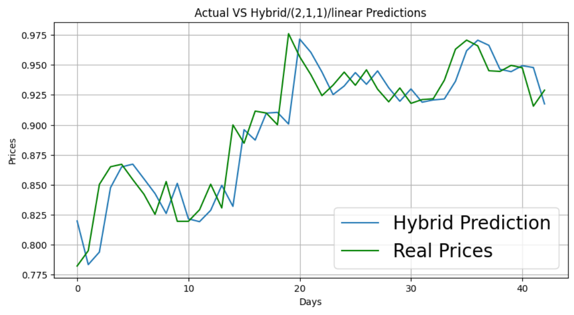
**(d)** Hybrid/(1,1,1) with tanh Activation Function predictions vs actual



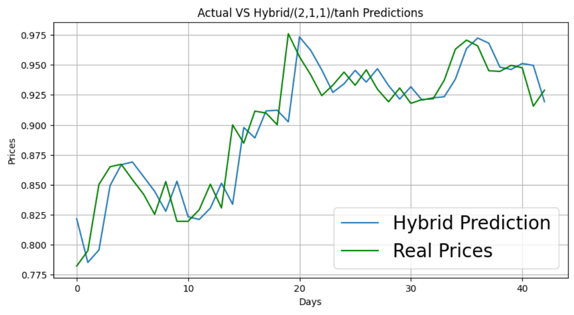
**(e)** Hybrid(2,1,0) with linear Activation Function predictions vs actual



**(f)** Hybrid (2,1,0) with tanh Activation function predictions predictions vs actual



**(g)** Hybrid (2,1,1) with linear Activation Function predictions vs actual



**(h)** Hybrid (2,1,1) with tanh Activation function predictions vs actual

**Figure 4.9:** Daily Prices of BTC Predicted with Hybrid Models

## 4.10 Final Results

In the table we comparing best the models, using metrics like RMSE and MAPE to assess their accuracy in forecasting daily Bitcoin data.

| The best Models | order | RMSE | MAPE | |
|---|---|---|---|---|
| ANN | - | linear | 1735.94799 | 0.01994308 |
| ARIMA | (1,1,0) | - | 1611.48890 | 0.01812232 |
| **Hybrid** | **(1,1,0)** | tanh | **1611.48676** | **0.018122319** |

**Table 4.16:** Comparing the 3 best Models

We can see that the hybrid/(1, 1, 0) model with the tanh activation function exhibited better results in terms of accuracy and precision metrics compared to the others models.

## 4.11 Best Models Forecast

The forecasts of the best models of the actual daily Bitcoin were plotted. Each plot illustrates the models' predictions and their alignment with observed price movements over time.These visual comparisons allow for an evaluation of how effectively each model captures Bitcoin's price volatility and overall trend.

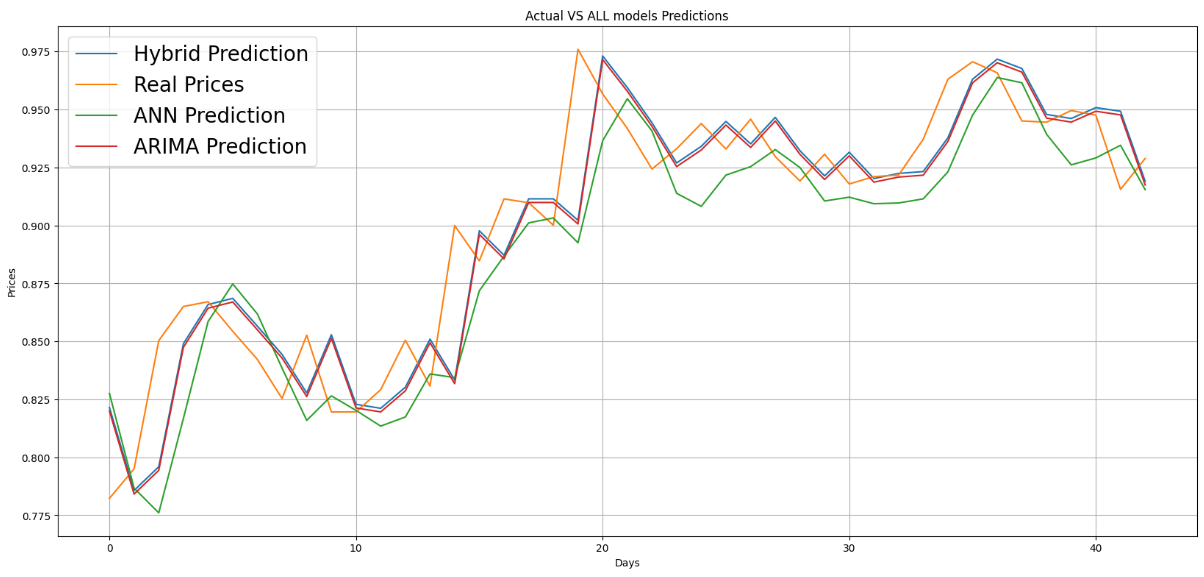


**Figure 4.10:** Plot of the Final Forecasts of the best Models

# Conclusion

Time series analysis and Forecasting is an important problem that spans many fields including business and industry, government, economics, environmental sciences, medicine, social science, politics, and finance.
Forecasting in finance provides valuable insights and tools for investors, financial institutions, and corporations to navigate uncertainty, optimize financial decisions, manage risks effectively, and achieve long-term financial objectives.
Forecasting problems are often classified as short-term, medium-term, and long-term. Short-term forecasting problems involve predicting events only a few time periods (days, weeks, and months) into the future. Medium-term forecasts extend from 1 to 2 years into the future, and long-term forecasting problems can extend beyond that by many years.
The accuracy of time series forecasting is fundamental to many decision processes and hence the research for improving the effectiveness of forecasting models has never stopped.

This thesis used a hybrid model that combines the ARIMA model and neural networks to forecast non-seasonal time series data. By employing the linear ARIMA model alongside the nonlinear ANNs model, the hybrid approach aims to capture different patterns within the time series data. This integrative model leverages the strengths of both ARIMA and ANNs in linear and nonlinear modeling. The combined approach is designed to improve forecasting performance by addressing the complexity of both linear and nonlinear structures.

One of the perspectives of this work is to combine ARIMA model with other models, such as:

- ARIMA-SVR Hybrid Model which Combines ARIMA with Support Vector Regression (SVR), where ARIMA handles the linear aspects and SVR captures the nonlinear relationships.
- ARIMA-GARCH Hybrid Model which Combines Integrates ARIMA for modeling the mean structure with GARCH (Generalized Autoregressive Conditional Heteroskedasticity) for capturing volatility clustering in time series data.
- ARIMA-Exponential Smoothing Hybrid Model which Combines ARIMA with Exponential Smoothing (ETS) methods to leverage their complementary strengths.
- ARIMA-LSTM Hybrid Model which combines Integrates ARIMA with Long Short-Term Memory (LSTM) networks, which are a type of recurrent neural network (RNN) that is effective for sequence prediction.
- ARIMA-Random Forest Hybrid Model which Combines ARIMA for linear modeling with Random Forest, which is a robust ensemble learning method capable of capturing complex nonlinear relationships.
- ARIMA-XGBoost Hybrid Model which Combines Integrates ARIMA with XGBoost, an efficient and scalable implementation of gradient boosting.
- Decomposition-ANN Hybrid Model which Decomposes the time series into trend, seasonal, and residual components, then applies different models to each component.

# Appendix A

```
[ ]: #IMPORTING LIBRARIES
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
import itertools
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import warnings
warnings.filterwarnings("ignore")
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.stattools import adfuller
from keras.models import Sequential
from keras.layers import LSTM,Dropout
from sklearn.metrics import␣
 ↪mean_squared_error,mean_absolute_error,mean_absolute_percentage_error
from keras.layers import Dense,Activation
#getting data ranges(starrt date and end date)
start="2020-01-01"
end="2024-06-13"
#downlouding data
data = yf.download("BTC-USD",start=f'{start}',end=f'{end}')["Close"]
#ploting BTC data
plt.figure(figsize=(10,5))
plt.plot(data,color="b",label="Data")
plt.legend(fontsize=20)
plt.title("Data")
plt.ylabel("Prices")
plt.xlabel("Dates")
plt.grid()
plt.show()
#applying ADF test
adf_before_diff=adfuller(data)
```

```
adf_before_diff
#difrencing data
data_dif=data.diff().dropna()
#ploting data after diffrencing
plt.figure(figsize=(10,5))
plt.plot(data_dif,color="b",label="Data")
plt.legend(fontsize=20)
plt.title("Data after difrencing")
plt.ylabel("Prices")
plt.xlabel("Dates")
plt.grid()
plt.show()
#ploting auto correlation and partial auto correlation
plot_pacf(data_dif)
plot_acf(data_dif)
plt.show()
#calculatig AIC and BIC and
best_aic = np.inf  # Initialize with a large number
best_p = None       # Will determine the best p
best_q = None       # Will determine the best q

# Iterate over possible values of p and q
for p in range(1, 3):   # Range is 1 to 2 (exclusive)
    for q in range(2):  # Range is 0 to 1 (exclusive)
        try:
            # Fit ARIMA model with given p and q
            model = sm.tsa.ARIMA(data, order=(p, 1, q)).fit()
            # Get AIC for the current model
            current_aic = model.aic
            # Compare current AIC with the best found so far
            if current_aic < best_aic:
                best_aic = current_aic
                best_p = p
                best_q = q
        except:
            continue
# Print the best values found
print(f"Best (p, q) values: ({best_p}, {best_q})")
print(f"Best AIC: {best_aic}")
#predicting with ARIMA
training_data=data[:-50].tolist()
test=data[-50:].tolist()
error_list=[]
predicted_list=[]
for t in range(len(test)):
    model = ARIMA(training_data, order=(1,1,0))
    model_fit = model.fit()
    predicted_value = model_fit.forecast()
    predicted_value = predicted_value[0]
    error_list.append(test[t] - predicted_value)
    predicted_list.append(predicted_value)
    obs = test[t]
    training_data.append(obs)
#ploting forecasts
plt.figure(figsize=(10,5))
plt.plot(predicted_list[7:],color="b",label="ARIMA Predictions")
plt.plot(test[7:],color="g",label="Real Prices")
plt.ylabel("Prices")
plt.xlabel("days")
plt.title("ACTUAL VS ARIMA predictions")
plt.legend(fontsize=20)
plt.show()
#scaling data
scaler = MinMaxScaler(feature_range=(0, 1))
data = scaler.fit_transform(np.array(data).reshape(-1,1))
days_touse=50
trains=[]
target=[]
for i in range(days_touse,len(data)):
    trains.append(data[i-days_touse:i].tolist())
    target.append(data[i])
train=np.array(trains).reshape(len(trains),50,1)
target=np.array(target).reshape(-1,1)
split = -50
X_train, X_test = train[:split], train[split:]
y_train, y_test = target[:split], target[split:]
#building ANN model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(train.shape[0],␣
 ↪1)))
#model.add(Dropout(0.2))
model.add(LSTM(100,activation="linear"))
#model.add(Dropout(0.2))
model.add(Dense(1,activation="linear"))
#training model
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=27, batch_size=32)
pred=model.predict(X_test)
preds=scaler.inverse_transform(pred)
tests=scaler.inverse_transform(y_test.reshape(-1,1))
#ploting ANN forcasts
plt.figure(figsize=(10,5))
plt.plot(preds,color="b",label="ANN predictions")
plt.plot(tests,color="g",label="Real Prices")
plt.ylabel("Prices")
plt.xlabel("days")
plt.title("ACTUAL VS ANN predictions")
plt.legend(fontsize=20)
plt.grid()
plt.show()
# building a hybrid model
#getting train and test errors from ARIMA
train_error=data[:-50]
training_error=ARIMA(train_error,order=(1,1,0)).fit().resid
model = Sequential()
model.add(LSTM(100,return_sequences=True, input_shape=(7, 1)))
model.add(LSTM(100))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
train_X,train_Y = [],[]
for i in range(0 , len(training_error) - 7):
    train_X.append(training_error[i:i+7])
    train_Y.append(training_error[i+7])
new_train_X,new_train_Y = [],[]
for i in np.array(train_X):
    new_train_X.append(i.reshape(-1))
for i in np.array(train_Y):
    new_train_Y.append(i.reshape(-1))
new_train_X = np.array(new_train_X)
new_train_Y = np.array(new_train_Y)
model.fit(new_train_X,new_train_Y, epochs=50, batch_size=32)
test_data = []
for i in error_list:
    try:
        test_data.append(i[0])
    except:
        test_data.append(i)
test_data = np.array(test_data)
test_X,test_Y = [],[]
for i in range(0 , len(test_data) - 7):
    test_X.append(test_data[i:i+7])
    test_Y.append(test_data[i+7])
new_test_X,new_test_Y = [],[]
for i in test_X:
    new_test_X.append(i.reshape(-1))
for i in test_Y:
    new_test_Y.append(i.reshape(-1))
new_test_X = np.array(new_test_X)
new_test_Y = np.array(new_test_Y)
#predicting with hybrid model
predictions = model.predict(new_test_X)
```

```
Y = pd.DataFrame(new_test_Y)
pred = pd.DataFrame(predictions)
pred_final = predictions + predicted_list[7:]
pred_hybrid_gt = pd.DataFrame(pred_final[-1])
#ploting hybrid model forecasts
plt.figure(figsize=(10,5))
plt.plot(pred_hybrid_gt,label="Hybrid Prediction")
plt.plot(test[7:],color="g",label="Real Prices")
plt.ylabel("Prices")
plt.xlabel("Days")
plt.title("Actual VS Hybrid/(1,1,0)/tanh Predictions")
plt.legend(fontsize=20)
plt.grid()
plt.show()
#models preformance
print("rmse of hybrid model =",np.
  ↪sqrt(mean_squared_error(pred_hybrid_gt,test[7:])))
print("rmse of ANN model =",np.sqrt(mean_squared_error(preds[7:],test[7:
  ↪])))
print("rmse of ARIMA model =",np.
  ↪sqrt(mean_squared_error(predicted_list[7:],test[7:])))
print("mae of hybrid model =",mean_absolute_error(pred_hybrid_gt,test[7:
  ↪]))
print("mae of ANN model =",mean_absolute_error(preds[7:],test[7:]))
print("mae of ARIMA model =",mean_absolute_error(predicted_list[7:
  ↪],test[7:]))
print("mape of hybrid model␣
  ↪=",mean_absolute_percentage_error(pred_hybrid_gt,test[7:]))
print("mape of ANN model =",mean_absolute_percentage_error(preds[7:
  ↪],test[7:]))
print("mape of ARIMA model␣
  ↪=",mean_absolute_percentage_error(predicted_list[7:],test[7:]))
```

# Bibliography

Abu-Mostafa, Y. S. and Atiya, A. F. (1996). Introduction to financial forecasting. *Applied intelligence*, 6:205–213.

Ali, A., Ghazali, R., and Deris, M. M. (2011). The wavelet multilayer perceptron for the prediction of earthquake time series data. In *Proceedings of the 13th International Conference on Information Integration and Web-based applications and services*, pages 138–143.

Alsuwaylimi, A. A. (2023). Comparison of arima, ann and hybrid arima-ann models for time series forecasting. *Inf. Sci. Lett*, 12(2):1003–1016.

Altun, H., Bilgil, A., and Fidan, B. (2007). Treatment of multi-dimensional data to enhance neural network estimators in regression problems. *Expert Systems with Applications*, 32(2):599–605.

Ashoori, S. and Mohammadi, S. (2011). Compare failure prediction models based on feature selection technique: empirical case from iran. *Procedia Computer Science*, 3:568–573.

Bullinaria, J. A. (2013). Recurrent neural networks. *Neural Computation: Lecture*, 12(1).

Chatfield, C. (2000). *Time-series forecasting*. Chapman and Hall/CRC.

de Gooijer, J. G. (2017). Elements of nonlinear time series analysis and forecasting. page 626.

Douglas C. Montgomery, Cheryl L. Jennings, M. K. (2015). Introduction to time series analysis and forecasting. page 671.

Eshel, G. (2020). The yule walker equations for the ar coefficients.

Gebhard Kirchgassner, J. W. (2007). Introduction to modern time series analysis. *Springer Berlin Heidelberg New York*, page 227.

Hamid, S. A. and Habib, A. (2014). Financial forecasting with neural networks. *Academy of Accounting and Financial Studies Journal*, 18(4):37.

Howlett, R. J. and Jain, L. C. (2001). *Radial basis function networks 2: new advances in design*, volume 2. Springer Science & Business Media.

Jianqing Fan, Q. Y. (2003). Time series analysis and its applications with r examples. page 569.

Khashei, M. and Bijari, M. (2010). An artificial neural network (p, d, q) model for time-series forecasting. *Expert Systems with applications*, 37(1):479–489.

Klieštik, T. (2013). Models of autoregression conditional heteroskedasticity garch and arch as a tool for modeling the volatility of financial time series. *Ekonomicko-manažerské spektrum*, 7(1):2–10.

Lek, S. and Park, Y. (2008). Artificial neural networks. In *Encyclopedia of Ecology, Five-Volume Set*, pages 237–245. Elsevier Inc.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Mileris, R. and Boguslauskas, V. (2011). Credit risk estimation model development process: Main steps and model improvement. *Engineering Economics*, 22(2):126–133.

Peter J. Brockwell, R. A. D. (2002). Introduction to time series and forecasting. page 449.

Peter J. Brockwell, R. A. D. (2006). Time series: Theory and methods. page 589.

Robert H. Shumway, D. S. S. (2011). Nonlinear time series: Nonparametric and parametric methods. page 202.

Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 8(4):e1249.

Santra, R. (2023). Dickey fuller and augmented dickey fuller(adf) test.

Sharma, S., Sharma, S., and Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316.

Suda, J. (2013). Stationary time series analysis.

Suzuki, K. (2011). *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand.

Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175.