**“FORECASTING GOLD PRICE USING ARIMA MODEL”**

By

**Ujjwala Vayuvegula**

**107219437060**

**Under the supervision of**

**Mrs. L V Kamala Devi,**

**HOD, Commerce Department.**

**Submitted in partial fulfillment**

**For the award of the degree of**

# BACHELOR OF COMMERCE (HBA)



**Bhavan’s Vivekananda College**

**of Science, Humanities & Commerce**

**Autonomous College -Affiliated to Osmania University**

**Re-accredited with ‘A’ Grade by NAAC**

Sainikpuri, Secunderabad-500094

**2021-2022**

**DECLARATION**

I hereby declare that this project entitled **“Forecasting Gold Price using ARIMA”** has been prepared by me in partial fulfillment of the requirement for award of Degree, Bachelor of Commerce (HBA).

I also hereby declare that this project report is the result of my own effort and that it has not been submitted to any other university or institution for the award of any other degree or diploma.

Place : Hyderabad **Name of the candidate**

Date : 28-07-2022 Ujjwala Vayuvegula

**CERTIFICATION**

This is to certify that the project report titled **Forecasting Gold Price using ARIMA** submitted in partial fulfilment for the award of B.Com (HBA) program of Department of Commerce, Bhavan’s Vivekananda College of Science Humanities & Commerce, Sainikpuri, Secunderabad was carried out by (**Ujjwala Vayuvegula)** under my guidance. This has not been submitted to any other university or institution for the award of any other degree, diploma or certificate.

Name of the Guide: Mrs. L V Kamala Devi

Place : Hyderabad Signature of Guide

**ACKNOWLEDGEMENT**

I take this opportunity to express my deep sense of gratitude and indebtedness to **Mrs. B. Niraimathi,** Officiating Principal, Bhavan’s Vivekananda College of Science, Humanities and Commerce and **Mrs. L.V. Kamala Devi,** Head, Department of Commerce for their co-operation extended to me in the preparation of this project work.

I sincerely thank the project co-ordinator **P.R. Venugopal** for his support. I am profoundly thankful to my project guide **(Mrs. L V Kamala Devi)** for her valuable guidance and suggestions for the successful completion of my project.

I would also like to thank all other faculty members of Department of Commerce for giving valuable suggestions and guidance.

**Name of the candidate**

Ujjwala Vayuvegula

**TABLE OF CONTENTS:**

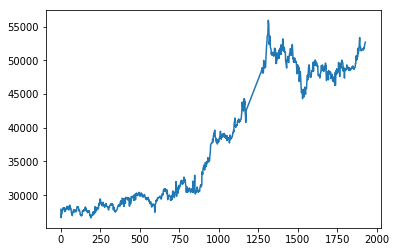
|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **TITLE** | **PAGE NO** |
| **1** | **INTRODUCTION** | **7** |
| **1.1** | **Literature Review** | **13** |
| **1.2** | **Data and Methodology** | **14** |
| **2** | **THEORATICAL FRAMEWORK** | **16** |
| **2.2** | **Machine Learning For Time series** | **21** |
| **2.3** | **Error Metrics for Time series** | **23** |
| **2.4** | **Python For Time series** | **27** |
| **2.5** | **ARIMA Model** | **29** |
| **3** | **DATA ANALYSIS** | **42** |
| **4** | **INTERPRETATION** | **64** |
| **5** | **FINDINGS , SUGGESTIONS & CONCLUSION** | **66** |
| **5** | **BIBILIOGRAPHY** | **69** |
| **6** | **ANNEXURE**  **DATASET** | **70** |

CHAPTER 1

**INTRODUCTION**

“All that glitters is not gold” is a proverb used to depict that not everything that looks attractive is not so, in reality. This yellow metal has grab lot of attention for every class of people as investment purpose. Gold is considered a valuable asset that never goes out of trend. People investing in gold have mainly two primary objectives, one being it is a hedge against inflation as over a period of time, the return on gold investment is in line with the rate of inflation, next to mix your investment basket and hence diversify the risk and will help you reduce the overall volatility of your portfolio. Investing in gold have evolved over a period of time for traditional ways by buying jewelleries or by modern way as purchasing gold coins and bars or by investing in Gold Exchange traded fund (Gold ETF).

One of the most important minerals in the world is gold. Despite making valuable commodities, gold acts as a reserve in any country. A gold reserve is an amount of gold held by the central bank of any country for the purpose of the guarantee to be used to pay or trading in the world market and hence increase the country economically. Amongst all mineral in the world, gold is the most popular selection for the investment. The price of gold is affected by different factors, thus making the movement of price to be unstable. These factors include inflation rate, demand and supply, and political issue among others. Inflation as one of the signs of economic growth, when it increases it is obvious pushes the gold price higher, while when having a low supply of any commodity, the price of that commodity increases. moreover, when countries fear the value of the dollar will be fallen since the dollar is the world's marketing currency, the gold price will eventually increase since many demands for gold will be available. Because of its importance, other literature has termed a safe haven during financial crises. Therefore, the price of gold moving up and down and very uncontrollable. Historically, gold had been used as a form of currency in various parts of the world including USA. In recent times also, gold has maintained its value and has been used as a means for assessing the financial strength of a country. Big investors have also been attracted to this precious metal and invested huge amounts in it. Recently, emerging world economies, such as China, Russia, and India have been big buyers of gold, whereas USA, South Africa, and Australia are among the big seller of this commodity. Chinese and Indian traditional events also affect the price of the gold. In that time more money is poured for purchase of this commodity. Small investors also find this commodity for safe investment rather than alternate investment options, which bear in-built investment risks. Internal financial conditions of the aforementioned countries play a vital role for setting spot rates for gold. Governmental investments in gold are largely decided by their financial conditions, and interest rates, as they are indicators of the strength of their economy. The spot price is the current market price at which commodity is purchased or sold for immediate payment and delivery. It is differentiated from the futures price, which is the price at which the two parties agree to transact on future date. Gold spot rates are decided twice a day based on supply and demand in gold market. Fractional change in gold price may result in huge profit or loss for these investors as well as government banks. Forecasting rise and decline in the daily gold rates, can help investors to decide when to buy (or sell) the commodity. Most of investors would like to keep a portion of their total assets in gold because it is low-to-negative correlation with stocks and bonds thereby make it an excellent portfolio diversifier. Gold investors may depend on historical data of gold price to forecast future prices prior to making their investment decision. The main reason for forecasting is to minimize risk when making a decisive move. Figure1 shows the movement of the gold price for the period of about 5years thus from 2017 to December 2022. Nevertheless, the gold price can be forecasted ahead, and that makes possible to make the future decision. The movement of the gold price is time-series order means changing with time, therefore doing the forecasting with such kind of data has been challenging for a while until the application of machine learning and deep learning introduced in the game of economics and statistics. This project gives an insight of forecasting of gold price through time-series ARIMA model. The need for this study is to provide financial institutions, investors, mining companies and related firms with an effective accurate forecasting model to examine gold price fluctuations in order to make correct decisions**.**

****

**Figure1**

Forecasting is a process in management to assist decision making. It is also described as the process of estimation in unknown future situations. In a more general term, it is commonly known as prediction which refers to estimation of time series or longitudinal type data. The most popular model for this method is the Box-Jenkins model introduced by Box-Jenkins. He has suggested the time-series autoregressive integrated moving average (ARIMA) model for forecasting. Like any other such methods, it requires historical time series data on the variable under forecasting. It assumes that the future values of a time series have a clear and definite functional relationship with current, past values and white noise.

**TIME SERIES**

Definition :  *Time series analysis* is the endeavour of extracting meaningful summary and statistical information from points arranged in chronological order. It is done to diagnose past behaviour as well as to predict future behaviour.

Time series data and its analysis are increasingly important due to the massive production of such data through, for example, the internet of things, the digitalization of healthcare, and the rise of smart cities. In the coming years we can expect the quantity, quality, and importance of time series data to grow rapidly.

As continuous monitoring and data collection become more common, the need for competent time series analysis with both statistical and machine learning techniques will increase. Indeed, the most promising new models combine both of these methodologies.

**HISTORY OF TIME SERIES IN DIVERSE APPLICATIONS**

Time series analysis often comes down to the question of causality: how did the past influence the future? At times, such questions (and their answers) are treated strictly within their discipline rather than as part of the general discipline of time series analysis. As a result, a variety of disciplines have contributed novel ways of thinking about time series data sets. The following are the areas where the time-series was used initially.

* Medicine
* Weather
* Economics
* astronomy

**THE ORIGINS OF STATISTICAL TIME SERIES ANALYSIS**

Statistics is a very young science. Progress in statistics, data analysis, and time series has always depended strongly on when, where, and how data was available and in what quantity. The emergence of time series analysis as a discipline is linked not only to developments in probability theory but equally to the development of stable nation states, where recordkeeping first became a realizable and interesting goal. We covered this earlier with respect to a variety of disciplines. Now we’ll think about time series itself as a discipline.

One benchmark for the beginning of time series analysis as a discipline is the application of autoregressive models to real data. This didn’t happen until the 1920s. Udny Yule, an experimental physicist turned statistical lecturer at Cambridge University, applied an autoregressive model to sunspot data, offering a novel way to think about the data in contrast to methods designed to fit the frequency of an oscillation. Yule pointed out that an autoregressive model did not begin with a model that assumed periodicity:

*When periodogram analysis is applied to data respecting any physical phenomenon in the expectation of eliciting one or more true periodicities, there is usually, as it seems to me, a tendency to start from the initial hypothesis that the periodicity or periodicities are masked solely by such more or less random superposed fluctuations—fluctuations which do not in any way disturb the steady course of the underlying periodic function or functions…there seems no reason for assuming it to be the hypothesis most likely a priori.*

Yule’s thinking was his own, but it’s likely that some historical influences led him to notice that the traditional model presupposed its own outcome. As a former experimental physicist who had worked abroad in Germany (the epicenter for the burgeoning theory of quantum mechanics), Yule would certainly have been aware of the recent developments that highlighted the probabilistic nature of quantum mechanics. He also would have recognized the dangers of narrowing one’s thinking to a model that presupposes too much, as classical physicists had done before the discovery of quantum mechanics.

As the world became a more orderly, recorded, and predictable place, particularly after World War II, early problems in practical time series analysis were presented by the business sector. Business-oriented time series problems were important and not overly theoretical in their origins. These included forecasting demand, estimating future raw materials prices, and hedging on manufacturing costs. In these industrial use cases, techniques were adopted when they worked and rejected when they didn’t. It probably helped those industrial workers had access to larger data sets than were available to academics at the time (as continues to be the case now). This meant that sometimes practical but theoretically underexplored techniques came into widespread use before they were well understood.

**THE ORIGINS OF MACHINE LEARNING TIME SERIES ANALYSIS**

Early machine learning in time series analysis dates back many decades. An oft-cited paper from 1969, “The Combination of Forecasts,” analysed the idea of combining forecasts rather than choosing a “best one” as a way to improve forecast performance. This idea was, at first, abhorrent to traditional statisticians, but ensemble methods have come to be the gold standard in many forecasting problems. Ensembling rejects the idea of a perfect or even significantly superior forecasting model relative to all possible models.

More recently, practical uses for time series analysis and machine learning emerged as early as the 1980s, and included a wide variety of scenarios:

* Computer security specialists proposed anomaly detection as a method of identifying hackers/intrusions.
* Dynamic time warping, one of the dominant methods for “measuring” the similarity of time series, came into use because the computing power would finally allow reasonably fast computation of “distances,” say between different audio recordings.
* Recursive neural networks were invented and shown to be useful for extracting patterns from corrupted data.

Time series analysis and forecasting have yet to reach their golden period, and, to date, time series analysis remains dominated by traditional statistical methods as well as simpler machine learning techniques, such as ensembles of trees and linear fits. We are still waiting for a great leap forward for predicting the future.

**LITERATURE REWIEW**

Prediction analysis has become more challenging due to increasing of availability of data which are not stable. Researchers and academicians are still working on the best way in finance and economics to conquer this challenge.

Abdullah Lazim (2012) has made a study on forecasting of gold bullion coin prices through ARIMA model. He had concluded by suggesting that the gold bullion coin selling prices are in upward trends and could be considered as a worthy investment. The result suggests that ARIMA (2, 1, 2) is the most suitable model to be used for forecasting gold bullion coin prices.

Banhi (2016) and in their research paper analysis uses Arima model to make forecasting of future gold price in India. They considered the data from July 1990 to Feb 2015. This study suggested that ARIMA (0, 1, 1) is the best to be used in forecasting gold prices in India since the ARIMA (0, 1, 1) has the least value of RMSE, MAPE, and MAE. In this study, they only used one method of ARIMA with different parameters. The comparison with other model did not occur. Arima can be used in prediction of livestock product as in 2019.

Xiaohui Yang (2018) in their research analysis uses daily gold price data from July 1st 2013 to June 29th, 2018 and performed ARIMA model to predict future gold prices. Their objective is to understand the efficiency of gold prices and make great investment choices. They have done forecasting for already passed years and compared the results with the reality and then calculated the relative error.

D Makala and Z Li (2021) in their research paper predicted gold price using ARIMA and SVM. This study is aimed at predicting the future price of gold using deep learning technology. Data used in this study research are daily gold price, that can be retrieve from the World Gold Council. The prices from the World Gold Council are indicated as per troy ounce. The dataset consists the daily prices from the January 1979 to December 2019. The results of this study revealed that SVM(Poly) is found to performs much better compared to the other SVM(RBF) and Arima model.

Naliniprava Tripathy (2017) The present paper aims to establish and corroborate the prediction of gold price in India. The paper uses the monthly time series data to discover the forecasting of gold price. The study used ARIMA model to predict the gold price. The study has further used different forecasting technique such as MAE, RMSE, MAPE, Max AE, and MAPE to determine the accuracy of the model. The result shows that ARIMA (0, 1, 1) is the best model for gold price prediction since the BIC is low and MAPE, Max AE and MAE are the least.

Dr. M. Massarrat Ali Khan (2013) has done a study to forecast the gold price and to check the accuracy of the model. He used Box-Jenkins, Auto Regressive Integrated Moving Average (ARIMA) methodology for building forecasting model. Results suggest that ARIMA (0,1,1) is the most suitable model to be used for predicting the gold price. For testing the forecasting accuracy Root Mean Square Error, Mean Absolute Error, and Mean Absolute Percentage Error are calculated.

P B Saranya (2020) has done a study on to forecast gold prices. The title of the study is “Modelling and forecasting gold prices using Arima”. The most popular tool the Box-Jenkins ARIMA was used to forecast the prices. The empirical results indicated that the adjusted ARIMA model provides better scope for predicting the prices in the near future. The results show that the gold prices have an increasing trend.

Chaku, Gabriel, Abdulrazaq, Aliyu, Adehi, Timnan (2022) have done a study on gold price forecasting. Title of the study is “Time Series Modelling and Forecasting of Gold Prices on International Financial Markets”. Application of SARIMA model in modelling and forecasting average monthly gold prices was carried out in this study. Data on gold from January 2015 to December 2020 was obtained. Monthly adjusted close prices were used for the analysis. The gold price data was stationary after first difference (D = -3.8426, P = 0.02183< 0.05). SARIMA (0,0,0) (0,1,1) [12] was identified as the best model that fit the gold price data with minimum AIC and BIC. Forecast of gold prices from January, 2021 to December 2025 was obtained. Forecast shows a rise and fall of the average monthly gold price over the forecast period (2021-2025).

**DATA & METHODOLOGY**

The data required for this study is collected from secondary sources like trusted websites, articles, journals etc. For the quantitative data, CMIE and few other reliable websites have been used to get the historic data of gold prices. In this study data of past 5 years that is from the year 2017 to 2021 has been considered. Daily fluctuations in the prices have been observed. The data contains two columns, dates and price of gold of 10grams per each day. The dates start from 02-01-2017. Objective of this study is to forecast gold prices for the year 2022 to 2023. This study deals with univariate time-series analysis.

There are 4 types of forecasting methods in time series. They are ARIMA model (Autoregressive Integrated Moving Average), SARIMA model (Seasonal Autoregressive Integrated Moving Average), VAR (Vector Autoregression), LSTM (Long Short-Term Memory network). In this study we are using ARIMA model for predicting gold price as the data is univariate.

The development of ARIMA model encompasses predominantly three steps such as identification, estimation, diagnostic checking. Before using ARIMA model, it is important to check the stationarity of the data. So, the study uses unit root test. The selection of the regression model will be made by observing the autocorrelation function (ACF) and partial autocorrelation function (PACF). We’ll look into it in detail in further chapters.

CHAPTER 2

**THEOROTICAL FRAMEWORK**

The Series of data points recorded over a specified period of time is called Time-series data**.** Time-series is a sequence of data points taken sequentially over time, or that a time-series is the result of a stochastic process.

Many disciplines, such as finance, public administration, energy, retail, and healthcare, are dominated by time-series data. Large areas of micro- and macro-economics rely on applied statistics with an emphasis on time-series analyses and modelling. The following are examples of time-series data:

* Daily closing values of a stock index
* Number of weekly infections of a disease
* Weekly series of train accidents
* Rainfall per day
* Sensor data such as temperature measurements per hour
* Population growth per year
* Quarterly earnings of a company over a number of years

This is only to name but a few. Any data that deals with changes over time is a time-series.

Depending on the frequency, a time series can be of yearly (ex: annual budget), quarterly (ex: expenses), monthly (ex: air traffic), weekly (ex: sales qty), daily (ex: weather), hourly (ex: stock price), minutes (ex: inbound calls in a call canter) and even seconds wise (ex: web traffic).

Since time is the primary index of the dataset, by implication, time-series datasets describe how the world changes over time. They often deal with the question of how the past influences the presence or future.

The increase of monitoring and data collection brings with it the need for both statistical and machine learning techniques applied to time-series to predict and characterize the behaviour of complex systems or components within a system. An important part of working with time-series is the question of how the future can be predicted based on the past. This is called forecasting.

A time series can be multivariate time series and univariate time series. A Multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables. This dependency is used for forecasting future values. A univariate time series has only one time dependent variable. This variable depends only on its past values.

**Patterns in time series**

Time-Series mostly come as discrete-time, where the time difference between each point is the same. The most important characteristics of time-series are the following:

* Long-term movements of the values (**trend**)
* Seasonal variations (**seasonality**)
* Irregular or cyclic components

A trend is the general direction in which something is developing or changing, such as a long-term increase or decrease in a sequence. An example of where a trend can be observed would be global warming, the process by which the temperatures on our planet have been rising over the last half-century.

Seasonality is a variation that occurs at specific regular intervals of less than a year. Seasonality can occur on different time spans, such as daily, weekly, monthly, or yearly. An example of weekly seasonality would be sales of ice cream picking up each weekend. Also, depending on where you live, ice cream might only be sold in spring and summer. This is a yearly variation.

Other than seasonal changes and trends, there is variability that's not of a fixed frequency or that rises and falls in a way that's not based on seasonal frequency. Some of these we might be able to explain based on the knowledge we have.

As an example of cyclic variability that's irregular, bank holidays can fall on different calendar days each year, and promotional campaigns could depend on business decisions, such as the introduction of a new product. As an example of cyclic changes that are not seasonal, changes at the scale of milliseconds or that take place over time periods longer than a year would not be called seasonal effects. **Stationarity** is the property of a time-series not to change its distribution over time as described by its summary statistics. If a time-series is stationary, it means that it has no trend and no deterministic seasonal variability, although other cyclical variability is permitted. Therefore, the removal of irregular components, trends, and seasonal fluctuations is an intrinsic aspect of applying these models. The models then forecast what's left after removing seasonality and trend: business cycles. Thus, to apply classical models, a time-series usually should be decomposed into different components. Thus, classical models are usually applied as follows:

1. Test for stationarity
2. Differencing [if non-stationarity detected]
3. Fit method and forecast
4. Add back the trend and seasonality

Classical time-series modelling approaches were introduced by George Box and Gwilym Jenkins in 1970 in their book *"*Time-Series Analysis Forecasting and Control*."* Most importantly, they formalized the ARIMA and ARMAX models and described how to apply them to time-series forecasting.

**Stationary data**



**Figure4**

**Non-Stationary Series**

**Figure 5**

**TIME SERIES ANALYSIS MODELS AND TECHNIQUES**

Just as there are many types and models, there are also a variety of methods to study data. Here are the three most common.

* **Box-Jenkins ARIMA models:** These univariate models are used to better understand a single time-dependent variable, such as temperature over time, and to predict future data points of variables. These models work on the assumption that the data is stationary. Analysts have to account for and remove as many differences and seasonalities in past data points as they can. Thankfully, the ARIMA model includes terms to account for moving averages, seasonal difference operators, and autoregressive terms within the model.
* **Box-Jenkins Multivariate Models:** Multivariate models are used to analyze more than one time-dependent variable, such as temperature and humidity, over time.
* **Holt-Winters Method:** The Holt-Winters method is an exponential smoothing technique. It is designed to predict outcomes, provided that the data points include seasonality.

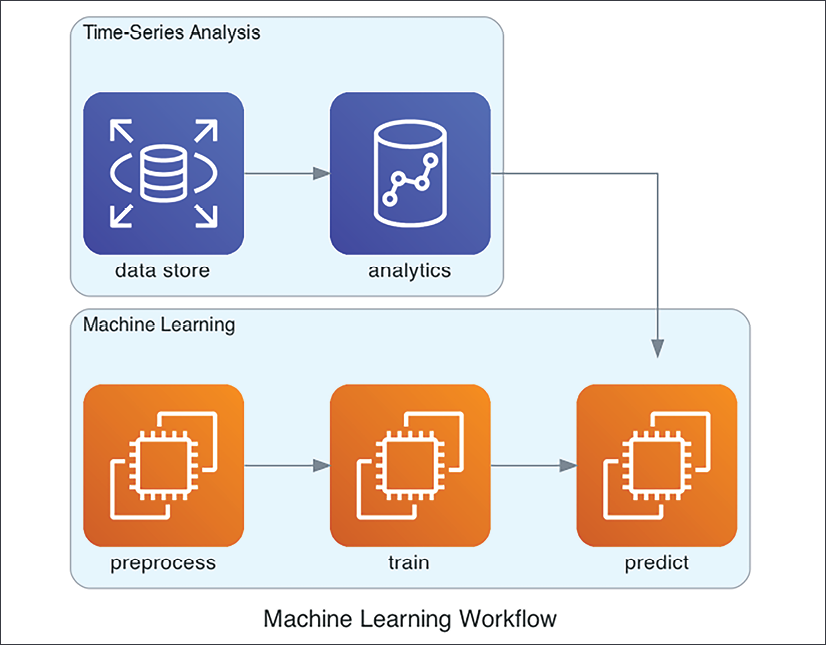
**MACHINE LEARNING FOR TIME SERIES**

Machine learning approaches for time-series are crucial in domains such as economics, medicine, meteorology, demography, and many others. Time-Series datasets are ubiquitous and occur in domains as diverse as healthcare, economics, social sciences, Internet-of-Things applications, operations management, digital marketing, cloud infrastructure, the simulation of robotic systems, and others. These datasets are of immense practical importance, as they can be leveraged to forecast and predict the detection of anomalies more effectively, thereby supporting decision making.

The technical applications within machine learning for time-series abound in techniques. A few applications are as follows:

* Curve fitting
* Regression
* Classification
* Forecasting
* Segmentation/clustering
* Anomaly detection
* Reinforcement learning

The machine learning workflow can be separated into three processes, as shown in the following diagram. (figure 6)



**Figure 6**

We first must transform (or preprocess) our data, train or fit a model, and then we can apply the trained model to new data. This diagram, very simplistic perhaps, puts the focus on the three different stages of the machine learning process. Each stage comes with its own challenges and particularities for time-series data.

This can also help to think about the data flow from input to transform to training to prediction. We should keep in mind the available historical data and its limitations, as well as the future data points that are to be used for predictions.

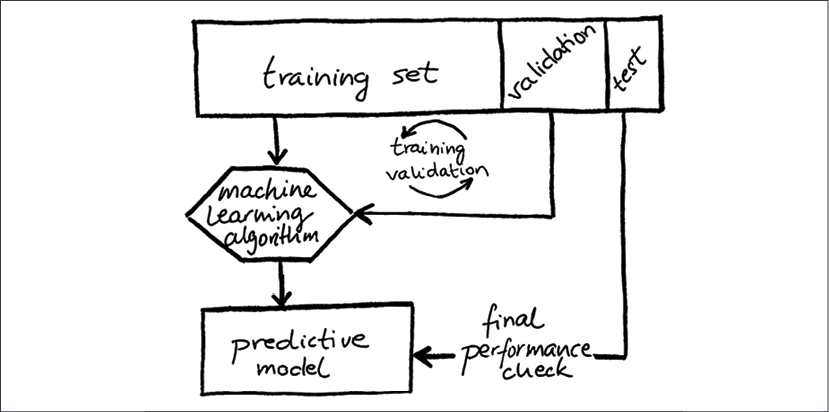
**CROSS VALIDATION**

Here's a well-known saying in machine learning attributed to George Box, whom we've encountered several times already in this book: "All models are wrong, but some are useful."

Machine learning algorithms make repeatable decisions and, given the correct controls, these decisions can be free from the cognitive biases that underlie much of human decision making. The point is to make sure that our model is useful by validating performance. In machine learning, the process of testing a model on data it hasn't seen in training is called cross-validation (sometimes, **out-of-sample testing**).

To ensure that parameters estimated on a dataset of limited size are still valid for more data, we must go through a validation that makes sure that the quality holds up. For validation, we usually split the dataset into at least two parts, the training set and the test set. We estimate parameters on a training set, and then run the model on the test set to get an idea of the quality of the model on unseen data points. This is illustrated in the following diagram (figure 7).

Usually, in machine learning we would shuffle points randomly before splitting between training and test. However, in time-series, we would take older data points for training and newer points for testing. For instance, having 1 year of data available on the email opening propensities of customers, we would train a model on 9 months' worth of data, validate our model on 2 months' worth of data, and test the final performance on the dataset.



**Figure7**

The use of validation and test can be seen as a nested process in the sense that the test set checks the main testing process that involves the validation dataset. Often, the separation of validation and test sets is omitted, so the dataset is split only into training and test sets.

A note about terminology: while a **loss function** is part of the optimization for training your model, a **metric** is used to evaluate your model. The evaluation can be post-hoc, after training, or during training as additional information.

**ERROR METRICS FOR TIME SERIES**

Time-series data is defined as a set of data points containing details about different points in time. Generally, time-series data contains data points sampled or observed at an equal interval of time.

For the different applications that we discussed earlier, we need to be able to quantify the performance of the model, be it a regression, classification, or another type of model, and choose a metric that captures the performance we want to achieve. Once we have chosen a metric for our model, we can then build and train models to improve them. Often, we'd start with a simpler model and then try to improve on the performance of this simpler model as a baseline. In the end, we want to find the model that is best according to our metric.

Generally, for an error measure, the smaller the values, the better the prediction (or the forecast). In changing the parameters of our model, we want to reduce the error. There's not just a single metric that's apt for the purpose of any arbitrary application or dataset. Depending on the dataset, you might have to search and try different error metrics and see which one best captures your objective. In some circumstances, you might even want to define your own metric.

**REGRESSION**

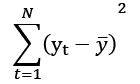
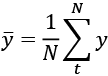
Time-Series regression is the task of identifying patterns and signals in the features in relation to the behaviour of time-series. During training, when your regression model gives a result on the training set, we can utilize a metric that compares the model output to the training set values, and during validation, we can calculate the same measure to know how good our regression predictions line up to the validation set targets. The error metric summarizes the difference between the values predicted by your machine learning model and the actual values.

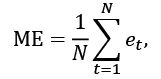
If  is a prediction of the model for time step t, and the actual target value is y***t***, intuitively, for a particular point, t, of our dataset, the **forecast error** (also **prediction error** or **residual**) is the difference between the actual values of the target and the values our model predicts:

This compares the actual target Y to the predicted targets . According to this formula, the error is negative if the prediction is higher than the actual target value. The **sum of squares of the residuals** (SS, also **residual sum of squares**) ignores the direction of the error:

While both the residual and the squared residual could already be used to measure the performance of predictions over a time-series, they are not commonly used as a regression metric or loss.

 The most commonly used metric for regression: the **coefficient of determination**. This is a relatively simple formula based on a ratio of the sum of the squares of the residuals, SS, and the total sum of squares, TSS, a measure of the variability:

In this fraction, the nominator is the sum of the squares of the residuals, SS, the unexplained variance.The denominator is TSS, the total sum of squares. This is defined as , where  is the mean of the series, . The total sum of squares represents the explained variance of the time-series.

Naively, we could take the average error, where we just take the mean over the forecast error – the mean error:

Here, *N* is the number of points (or the number of discrete time steps). We calculate the error for each point and then take the mean over all these errors.

If the ME is positive, the model systematically underestimates the targets, if it's positive, it overestimates the targets on the whole. While this can be useful, it's a serious problem for an error metric, however, because the effects of positive and negative errors cancel each other out. Therefore, a low ME does not mean that predictions are good, rather that the average is close to zero.

Furthermore, most regression models include a constant term that is equal to the mean of the target, so this value would be exactly 0. In conclusion, our naïve measure is useless in practical settings.

In the case of the ME, the residual operation is the identify function, which means the residual doesn't change. More often, the square or absolute functions are used. The integration of the errors is often the (arithmetic) mean, but sometimes the median; however, it can be a more complex operation.

In practice, the most popular error metrics are the mean squared error (MSE), mean absolute error (MAE), and the root mean squared error (RMSE). These most important error metrics are defined in the following table: (figure 8)

|  |  |
| --- | --- |
| Metric Name | Definition |
| Mean squared error |  |
| Mean absolute error |  |
| Root mean squared error |  |

**Figure8**

With the **mean squared error (MSE)**, we calculate the residual for each point, then square them, so positive and negative errors don't cancel each other out. Then we take the mean over these squared errors. An MSE of 0 indicates perfect performance. This can happen with toy datasets that you can play around with for fun; however, in practice, this will only happen if you made a mistake in building your dataset or in validation, because real life is always more complex than you can capture with a model.

The **mean absolute error (MAE)** is very similar to the MSE, only instead of squaring the residuals, we take their absolute values. As opposed to the MSE, all errors contribute in linear proportion (rather than being squared).

A major difference between taking the absolute versus taking the square is in how outliers or extreme values are treated. The square function forces a higher weight on values that are very different. With the MSE, the error grows quadratically instead of linearly as is the case with the MAE. This means that the MSE punishes extreme values much more strongly and, as a result, it is less robust to outliers in the dataset than the MAE. The distribution of the errors is a major concern in choosing an error metric that's right for the job.

Another common metric is the **root mean squared error (RMSE)**, or **root mean square deviation (RMSD),** which, as the name suggests, is the square root of the MSE. In that sense, RMSE is a scaled version of the MSE. Which one to take between the two is a presentation choice – both of them would lead to the same models.

What makes the RMSE interesting as a choice is that it comes in the same units and scale as the predicted variable, which makes it more intuitive. Finally, the RMSE is equivalent to the standard deviation or the error. This connection between standard deviation and the distribution of errors is quite meaningful, and you can summarize the error distribution with other measures such as the standard error or confidence interval

The **mean percentage error (MAPE)** is the mean average error normalized by the target. 0 represents a perfect model, and higher than 1 means the model's predictions are systematically higher than the targets. The MAPE doesn't have an upper bound. Additionally, since it deals with percentages in terms of the target (scaling or dividing by the targets), positive and negative residuals are treated differently. As a result, if the prediction is bigger than the target, the MAPE is higher than for the same error in the other direction. Therefore, depending on the sign of the residual, the MAPE is higher or lower!

**PYTHON FOR TIME SERIES**

For time-series, there are two main languages, R and Python, and it's worth briefly comparing the two and describing what makes Python special. Python is one of the top programming languages by popularity. According to the TIOBE from February 2021, it is only surpassed in popularity by C and Java.

R's community consists of statisticians and mathematicians, and R's strengths lie in statistics and plotting (ggplot). The weakness of R is its tooling and the virtual absence of consistent code style conventions.

On the other side, Python has been catching up in statistics and scientific computing with libraries such as NumPy, SciPy, and pandas, and it has overtaken R in both usage and usability for data science.

Python stands out in terms of machine learning libraries. The following libraries are written entirely or mainly in Python:

* Scikit-learn is written in Python and Cython (a Python dialect similar to the C programming language). It provides implementations of a very large set of algorithms for training and evaluating machine learning models.
* Statsmodels provides statistical tests, and models such as the generalized linear model (GLM), ARMA, and many more.
* Keras is an abstraction for training neural networks in Python that interact with TensorFlow and other libraries.

Some of the most popular machine learning frameworks – ones that see lots of use for development and have a large range of scalable algorithms, such as TensorFlow, PyTorch, and XGBoost – are also mainly written in Python or provide first-class interfaces for Python.

Furthermore, being a general-purpose language, Python is ideal if you want to go beyond just data analysis. With Python, you can implement the full data flow necessary for building an end-to-end machine learning system that you can deploy and integrate with the backend platforms of your company.

The term **time-series analysis** (**TSA**) refers to the statistical approach to time-series or the analysis of trend and seasonality. It is often an ad hoc*e*xploration and analysis that usually involves visualizing distributions, trends, cyclic patterns, and relationships between features, and between features and the target(s).

A part of TSA is collecting and reviewing data, examining the distribution of variables (and variable types), and checking for errors, outliers, and missing values. Some errors, variable types, and anomalies can be corrected, therefore EDA is often performed hand in hand with preprocessing and feature engineering, where columns and fields are selected and transformed.

Here are a few crucial steps for working with time-series:

* Importing the dataset
* Data cleaning
* Understanding variables
* Uncovering relationships between variables
* Identifying trend and seasonality
* Preprocessing (including feature engineering)
* Training a machine learning model

**MODEL ESTIMATION**

Box-Jenkins ARIMA model is one of the extensively used models for forecasting. The model uses no independent variable, but the prediction is made only from the historical series of a variable. The next step is to identify the model. The regressor that would be chosen to form the model will be selected from the variable lag of time of AR(p) and MA(q).

**ARIMA MODEL**

Analysts, multinational corporations, dealers in foreign exchange, exporters, importers and speculators commonly believe that past patterns provide an indication of future movement at least in the short run. Box and Jenkins (1976) ARIMA model is one of the widely used models for predicting the gold price today. The model assumes that the future values of time series have a functional relationship with current, past values and white noise. The model uses the historical value of the series for prediction. ARIMA model takes historical data and decomposes it into an Autoregressive process.

An ARIMA model is characterized by 3 terms: p, d, q where,

p is the order of the AR term

q is the order of the MA term

d is the number of differencing required to make the time series stationary

If a time series, has seasonal patterns, then you need to add seasonal terms and it becomes SARIMA, short for ‘Seasonal ARIMA’.

**MOVING AVERAGE AND AUTOREGRESSION**

Classical models can be grouped into families of models –  **moving averages** (**MA**), **autoregressive** (**AR**) models, ARMA, and ARIMA. These models were formalized and popularized over time in books and papers by many mathematicians and statisticians, including Peter Whittle (1951) and George Box and Gwilym Jenkins (1970). But let's start earlier.

The **moving average** marked the beginning of modern time-series predictions. In a moving average, the average (usually, the arithmetic mean) of values is taken over a specific number of time points (time frame) in the past.

More formally, the **simple moving average**, the unweighted mean over a period of k points, is formulated as:

where *x****i*** represents the observed time-series.

The moving average can be used to smooth out a time-series, thereby removing noise and periodic fluctuations that occur in the short term, effectively working as a low-pass filter. Therefore, as mathematician Reginald Hooker pointed out in a publication in 1902, the moving average can serve to isolate trend and oscillatory components. He conceptualized trend as the direction in which a series is moving when oscillations are disregarded.

The moving average can smooth the trend and cycle over the history of a time-series; however, as a model, the moving average can be used to forecast into the future as well. The time-series is a linear regression of the current value of the series against observed values (error terms). The moving average model of order *q*, *MA(q)*, can be denoted as:

where  is the average (expectation) of x***t*** (usually assumed to be 0),  are parameters, and  is random noise.

The invention of **AR** techniques dates back to a paper by British statistician Udny Yule, a personal friend of Hooker's, in 1927 *("*On a Method of Investigating Periodicities in Disturbed Time-Series with special reference to Wolfer's Sunspot Numbers*"*). An **autoregressive model** regresses the variable on its own lagged values. In other words, the current value of the value is driven by immediately preceding values using a linear combination.

These models work on the assumption that the data is stationary. Analysts have to account for and remove as many differences and seasonalities in past data points as they can. Thankfully, the ARIMA model includes terms to account for moving averages, seasonal difference operators, and autoregressive terms within the model. So how to make a series stationary?

Andrey Kolmogorov defined the term **stationary process** in 1931, although Louis Bachelier had introduced a similar definition earlier (1900) using different terminology. **Stationarity** is defined by three characteristics:

1. Finite variation
2. Constant mean
3. Constant variation

Constant variation means that the variation of the time-series in a window between two points is constant over time: , although it can change with the size of the window.

This is weak stationarity. In the literature, unless otherwise specified, usually stationarity means weak stationarity. Strict stationarity means that a time-series has a probability density function that is unchanged over time. In other words, under strict stationarity, the joint distribution over  is the same as over .

In 1938, Norwegian mathematician Herman Ole Andreas Wold described the decomposition of stationary time-series. He observed that stationary time-series can be expressed as the sum of a deterministic component (autoregressive) and a stochastic component (noise). This decomposition is termed after him today, as **Wold's decomposition**.

This leads to the formulation of the autoregressive model of order , AR(p), as:

where  is a model parameter, c is a constant, and  represents noise. In this equation, p is a measure of the autocorrelation between successive values of the time-series.

This work was later, in 1951, generalized to multivariate time-series in a Ph.D. thesis ("*Hypothesis Testing in Time-Series*") by New Zealander Peter Whittle, with Wold as his advisor. Peter Whittle is also credited with the integration of the AR and MA models into one, as the **autoregressive moving average** (**ARMA**). This was another milestone in the history of time-series modelling, bringing together the work of Yule and Hooker.

The ARMA model consists of two types of lagged values, one for the autoregressive component and the other for the moving average component. Therefore, we write *ARMA (p, q)*, with the first parameter p indicating the order of the autoregression, and the second, *q*, the order of the moving average, as:

ARMA assumes that the series is stationary. In practice, to ensure stationarity, preprocessing has to be applied.

ARIMA (p, d, q) includes a data preprocessing step, called **integration**, to make the time-series stationary, which is by replacing values by subtracting the immediate past values, a transformation called **differencing**.

While ARIMA type models effectively consider stationary processes, the **Seasonal Auto Regressive Integrative Moving Average** models (**SARIMA**), developed as an extension of the ARMA model, can describe processes that exhibit non-stationary behaviours both within and across seasons.

Seasonal ARIMA models are usually stated as ARIMA(p,d,q)(P,D,Q)m. The parameters deserve more explanation:

* m denotes the number of periods in a season
* P, D, Q parametrize the autoregressive, integration, and moving average components of the seasonal part
* p, d, q refers to the ARIMA terms, which we've discussed previously

P is a measure of autocorrelation between successive seasonal components of the time-series.

We can write out the seasonal parts to make this clearer. **Seasonal Autoregression**, **SAR**, can be stated as:

where s is the length of the seasonality.

Similarly, the **seasonal moving average**, **SMA**, can be written as follows:

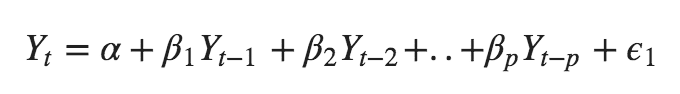
Please note that each of these components will use a distinct set of parameters. For example, the model SARIMA (0,1,1) (0,1,1)12 process will contain a non-seasonal MA (1) term (with the corresponding parameter ) and a seasonal MA (1) term (with the corresponding parameter ).

The model integration is parametrized by d, which is the number of times differences have been taken between current and previous values. As mentioned, the three parameters stand for the three parts of the model.

There are some special cases; ARIMA(p,0,0) stands for AR(p), ARIMA (0, d,0) for I(d), and ARIMA (0,0, q) is MA(q). I (0) is sometimes used as a convention to refer to stationary time-series, which don't require any differencing to be stationary.

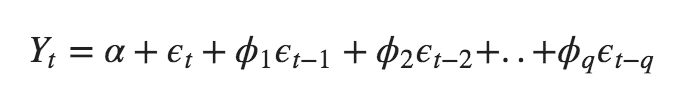
An ARIMA model can be understood by outlining each of its components as follows:

**Auto regressive (AR)** refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.

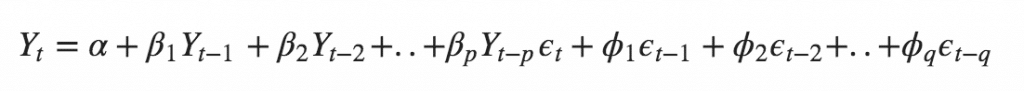


**Integrated (I)** represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).

**Moving average (MA)**incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.



An ARIMA model is one where the time series was differenced at least once to make it stationary and you combine the AR and the MA terms. So the equation becomes:



**MODEL SELECTION AND ORDER**

The parameter q in ARMA would typically be 3 or less, but this is more a reflection of computing resources rather than statistics. To set the parameters p and q, we would typically look at the autocorrelation and partial autocorrelation plots, where we could see peaks in the correlation for each lag.

Model selection is the methodology for deciding between competing models. One of the main ideas in model selection is Occam's razor, named after the English Franciscan friar and scholastic philosopher William of Ockham, who lived between circa 1287 and 1347.

According to Occam's razor, when choosing between competing solutions, one should prefer the explanation with the fewest assumptions. Ockham argued based on this idea that the principle of divine interventions is so simple that miracles are a parsimonious explanation. This rule, also called "lex parsimoniae" in Latin, expresses that a model should be parsimonious, which means that it should be simple yet have high explanatory power.

In science, simpler explanations are preferred out of the principle of falsifiability. The simpler a scientific explanation, the easier it can be tested, and possibly refuted – this lends the model scientific rigor.

ARMA and other models are usually estimated with the **maximum-likelihood estimation** (**MLE**). In MLE, this means maximizing a likelihood function so that, given the parameters of the model, the observed data is most likely.

One of the most commonly used model selection criteria for the maximum-likelihood method is the **Akaike information criterion** (**AIC**), after Hirotugu Akaike, who published it first in English in 1973.

AIC takes the log-likelihood l from the maximum-likelihood method and the number of parameters k in the model.

This is saying that the AIC equals two times the number of parameters minus two times the log-likelihood. In model selection, we would prefer the model with the lowest AIC, which means it has few parameters, but also high log-likelihood.

For an ARIMA model, we could write more specifically:

The **Bayesian Information Criterion** (**BIC**) was proposed for model selection a few years later (1978), by Gideon Schwarz, and looks very much like AIC. It additionally takes N, the number of samples in the dataset:

**UNDERSTANDING ARIMA RESULTS**

After creating an autoregression model, check the results to see if your model makes sense and how well it performs. Using statsmodels or any other library will print something out like the below (figure 9)



**Figure9**

**STEPS TO BE PERFORMED AFTER BUILDING ARIMA MODEL**

1. Review general information
2. Determine term significance
3. Analyse model assumptions
4. Compare models and improve the fit

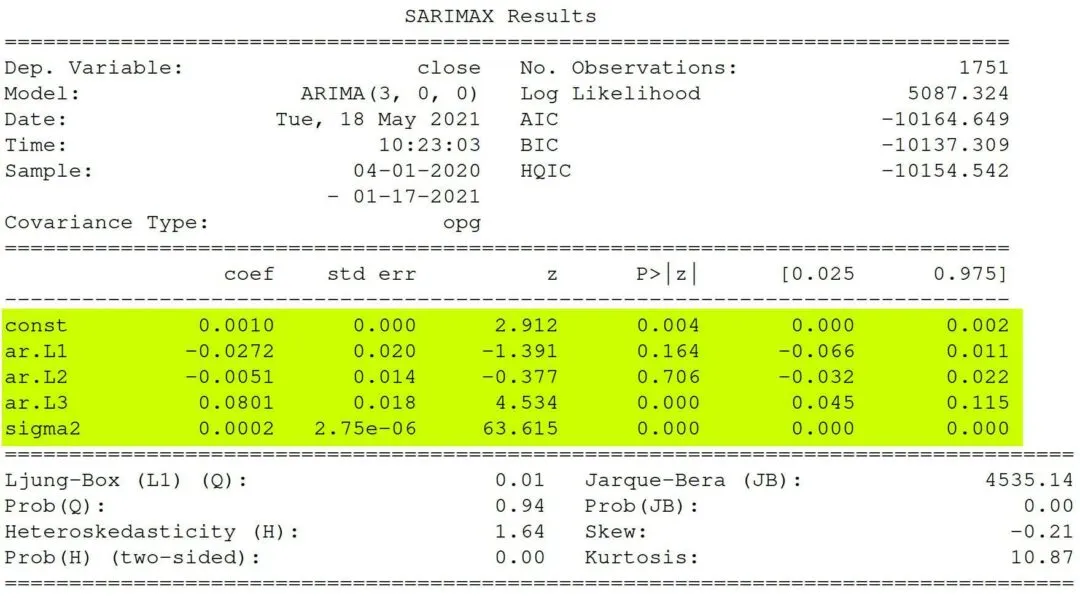
**REVIEW GENERAL INFORMATION**

The yellow-coloured part in figure4 is the general information. This information is pretty self-explanatory:

* Dep. Variable – What we’re trying to predict.
* Model – The type of model we’re using. AR, MA, ARIMA.
* Date – The date we ran the model
* Time – The time the model finished
* Sample – The range of the data
* No. Observations – The number of observations.

**DETERMINE TERM SIGNIFICANCE**

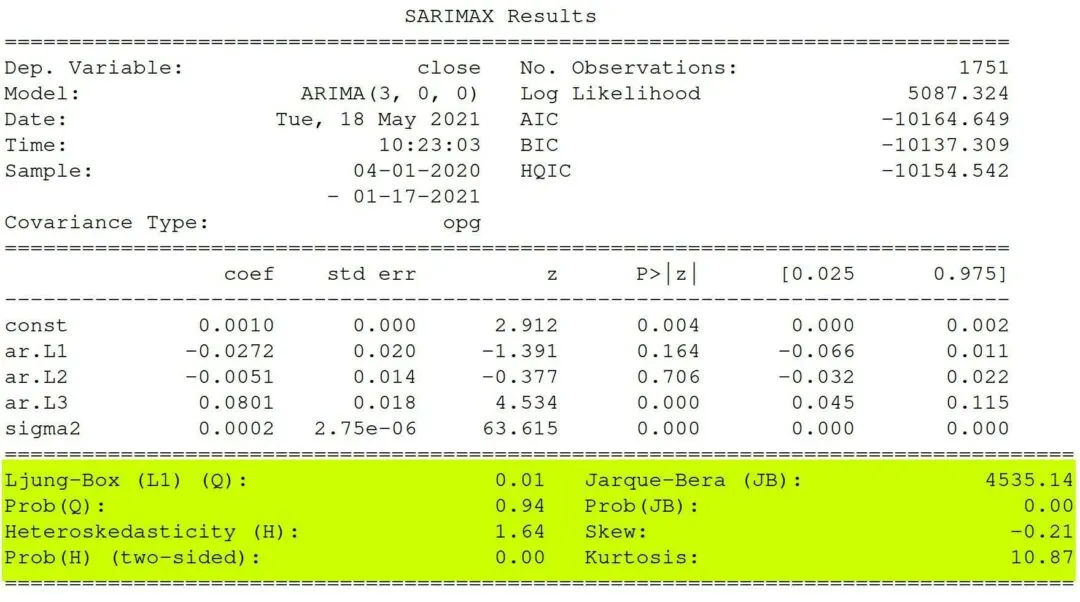
We want to make sure each term in our model is statistically significant. The null for this section is that each coefficient is NOT statistically significant. Therefore, we want each term to have a p-value of less than 0.05, so we can reject the null hypothesis with statistically significant values.

****

**Figure 10**

**REVIEW ASSUMPTIONS**

Next, we want to make sure our model meets the assumption that the residuals are independent, known as white noise. If the residuals are not independent, we can extract the non-randomness to make a better model.



**Figure 11**

**Ljung-Box**

The Ljung Box test, pronounced “Young” and sometimes called the modified Box-Pierce test, tests that the errors are white noise.

The Ljung-Box (L1) (Q) is the LBQ test statistic at lag 1 is, the Prob(Q) is 0.01, and the p-value is 0.94. Since the probability is above 0.05, we can’t reject the null that the errors are white noise.

If you’re interested in seeing all of the Ljung-Box test statistics and p-values for the lags, you can use a Ljung-Box diagnostic function.

**HETEROSCEDASTICITY**

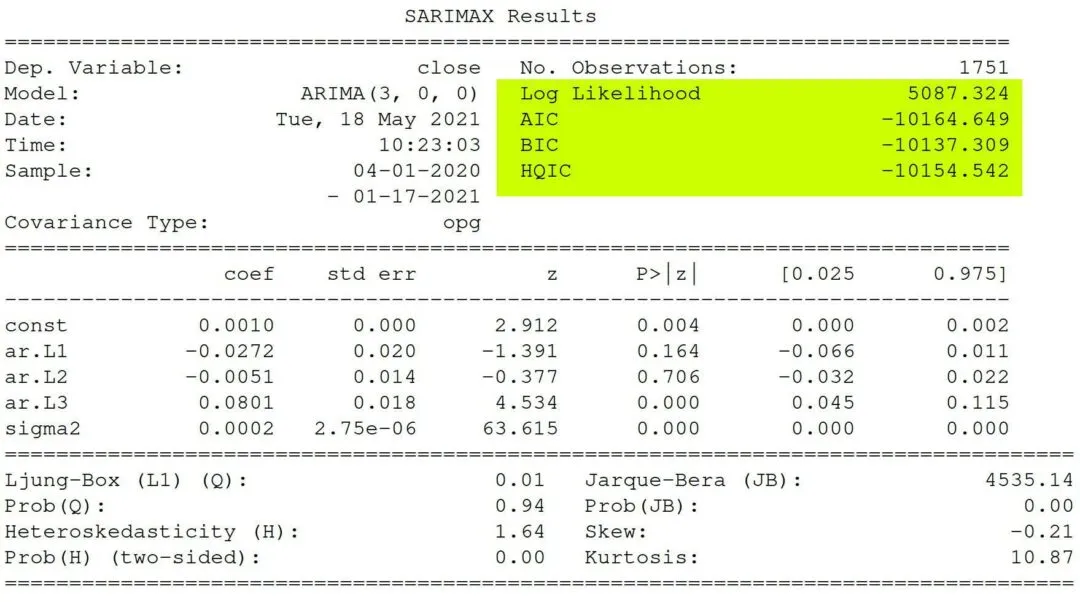
Heteroscedasticity tests that the error residuals are homoscedastic or have the same variance. The summary performs White’s test. Our summary statistics show a test statistic of 1.64 and a p-value of 0.00, which means we reject the null hypothesis and our residuals show variance.

**Jarque-Bera**

Jarque-Bera tests for the normality of errors. It tests the null that the data is normally distributed against an alternative of another distribution. We see a test statistic of 4535.14 with a probability of 0, which means we reject the null hypothesis, and the data is not normally distributed. Also, as part of the Jarque-Bera test, we see the distribution has a slight negative skew and a large kurtosis.

**FIT ANALYSIS**

The Log-Likelihood, AIC, BIC, and HQIC help compare one model with another.

****

**Figure 12**

**Log-Likelihood**

The log-likelihood function identifies a distribution that fits best with the sampled data. While it’s useful, AIC and BIC punish the model for complexity, which helps make our ARIMA model parsimonious.

**Akaike’s Information Criterion**

Akaike’s Information Criterion (AIC) helps determine the strength of the linear regression model. The AIC penalizes a model for adding parameters since adding more parameters will always increase the maximum likelihood value.

**Bayesian Information Criterion**

Bayesian Information Criterion (BIC), like the AIC, also punishes a model for complexity, but it also incorporates the number of rows in the data.

**Hannan-Quinn Information Criterion**

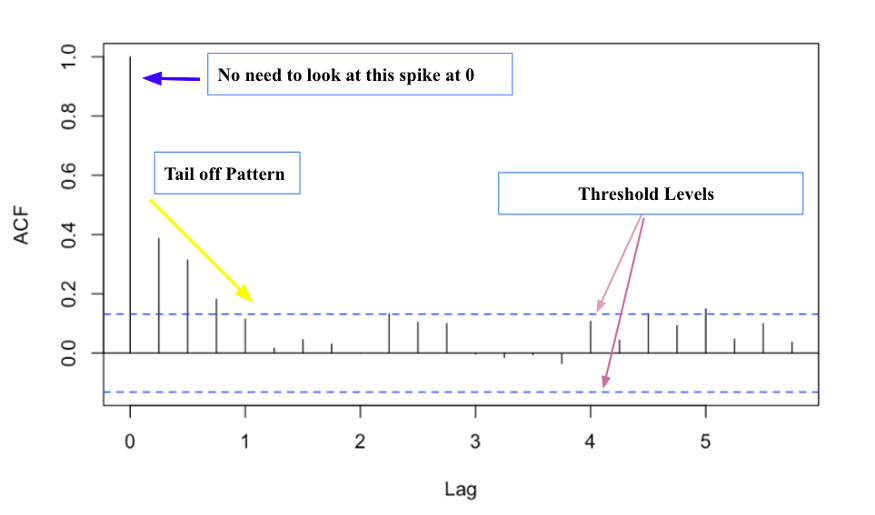
Hannan-Quinn Information Criterion (HQIC), like AIC and BIC, is another criterion for model selection; however, it’s not used as often in practice.

**ACF &PACF**

In time series analysis, Autocorrelation Function (ACF) and the partial autocorrelation function (PACF) plots are essential in providing the model’s orders such as p for AR and q for MA to select the best model for forecasting.

**The basic guideline for interpreting the ACF and PACF plots are as following:**

1. Look for tail off pattern in either ACF or PACF.
2. If tail off at ACF → AR model → Cut off at PACF will provide order p for AR(p).
3. If tail off at PACF → MA model → Cut off at ACF will provide order q for MA(q).
4. Tail of at both ACF and PACF → ARMA model



**Figure 13**

* The two blue dash lines pointed by purple arrows represent the significant threshold levels. Anything that spikes over these two lines reveals the significant correlations.
* When looking at ACF plot, we ignore the long spike at lag 0 (pointed by the blue arrow). For PACF, the line usually starts at 1.
* The lag axes will be different depending on the times series data.

CHAPTER 3

**DATA ANALYSIS**

**IMPORTING REQUIRED LIBRARIES**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **statsmodels.api** **as** **sm**

**import** **seaborn** **as** **sns**

**Pandas**

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

**Numpy**

**Numpy**is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

**Matplotlib.pyplot**

**Matplotlib**is a plotting library for creating static, animated, and interactive visualizations in Python.

**Statsmodels.api**

statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

**Seaborn**

**Seaborn**is a library mostly used for statistical plotting in Python. It is built on top of Matplotlib and provides beautiful default styles and colour palettes to make statistical plots more attractive.

**LOADING DATA**

data=pd.read\_csv("C:/Users/ushak/Desktop/Book 18.csv")

data=pd.DataFrame(data)

**REMOVING UNWANTED COLUMNS**

data=data.drop(['Prices.1','Prices.2'],axis=1)

data.shape

OUTPUT: (1229, 2)

**EXPLORATORY DATA ANALYSIS**

*# Getting top 5 rows of the data.*

data.head()

OUTPUT: Date Prices

0 13-04-2022 52700.0

1 12-04-2022 52562.5

2 11-04-2022 52500.0

3 08-04-2022 52212.5

4 07-04-2022 51987.5

*# Getting info of data*

data.info()

OUTPUT:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1229 entries, 0 to 1228

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 1229 non-null object

1 Prices 1229 non-null float64

dtypes: float64(1), object(1)

memory usage: 19.3+ KB

*# Getting data types*

data.dtypes

OUTPUT:

Date object

Prices float64

dtype: object

Right now our index is actually just a list of strings that look like a date, we’ll want to adjust these to be timestamps, that way our forecasting analysis will be able to interpret these values.

*# converting date into datetime format*

data['Date']=pd.to\_datetime(data['Date'],format='**%d**-%m-%Y')

*# describe data*

data.describe(include='all')

OUTPUT:

| **Date** | **Prices** |
| --- | --- |
| **count** | 1229 | 1229.000000 |
| **unique** | 1229 | NaN |
| **top** | 2017-10-30 00:00:00 | NaN |
| **freq** | 1 | NaN |
| **first** | 2017-01-02 00:00:00 | NaN |
| **last** | 2022-04-13 00:00:00 | NaN |
| **mean** | NaN | 37832.173230 |
| **std** | NaN | 9393.835797 |
| **min** | NaN | 26630.000000 |
| **25%** | NaN | 29201.000000 |
| **50%** | NaN | 34064.500000 |
| **75%** | NaN | 48482.500000 |
| **max** | NaN | 55950.000000 |

*# arranging data in assending order*

data=data.sort\_values('Date',ascending=**True**)

*# getting top 10 rows of arranged data*

data.head()

OUTPUT:

| **Date** | **Prices** |
| --- | --- |
| **1228** | 2017-01-02 | 27918.5 |
| **1227** | 2017-01-03 | 26676.0 |
| **1226** | 2017-01-04 | 27091.9 |
| **1225** | 2017-01-05 | 27078.6 |
| **1224** | 2017-01-06 | 27246.4 |
| **1223** | 2017-01-09 | 27311.2 |
| **1222** | 2017-01-10 | 27441.5 |
| **1221** | 2017-01-11 | 27599.0 |
| **1220** | 2017-01-12 | 27877.6 |
| **1219** | 2017-01-13 | 27795.3 |

*# getting last 10 rows of arranged data*

data.tail()

OUTPUT:

Date Prices

9 2022-03-31 51687.5

8 2022-04-01 51520.0

7 2022-04-04 51905.0

6 2022-04-05 51750.0

5 2022-04-06 51912.5

4 2022-04-07 51987.5

3 2022-04-08 52212.5

2 2022-04-11 52500.0

1 2022-04-12 52562.5

*# getting min date and max date*

mindate=min(data.Date)

maxdate=max(data.Date)

print(mindate)

print(maxdate)

OUTPUT:

2017-01-02 00:00:00

2022-04-13 00:00:00

**FILLING MISSING DATES**

seq=pd.DataFrame(pd.date\_range(mindate,maxdate))

seq.head(10)

seq.tail(10)

OUTPUT:

0

0 2017-01-02

1 2017-01-03

2 2017-01-04

3 2017-01-05

4 2017-01-06

5 2017-01-07

6 2017-01-08

7 2017-01-09

8 2017-01-10

9 2017-01-11

OUTPUT:

|  | **0** |
| --- | --- |
| **1918** | 2022-04-04 |
| **1919** | 2022-04-05 |
| **1920** | 2022-04-06 |
| **1921** | 2022-04-07 |
| **1922** | 2022-04-08 |
| **1923** | 2022-04-09 |
| **1924** | 2022-04-10 |
| **1925** | 2022-04-11 |
| **1926** | 2022-04-12 |
| **1927** | 2022-04-13 |

seq.columns=['date']

data.columns

**OUTPUT:** Index(['Date', 'Prices'], dtype='object')

data.all=pd.merge(seq,data,how='outer',left\_on=['date'],right\_on=['Date'],sort=**True**)

data.all.head(20)

**OUTPUT:**

| **date** | **Date** | **Prices** |
| --- | --- | --- |
| **0** | 2017-01-02 | 2017-01-02 | 27918.5 |
| **1** | 2017-01-03 | 2017-01-03 | 26676.0 |
| **2** | 2017-01-04 | 2017-01-04 | 27091.9 |
| **3** | 2017-01-05 | 2017-01-05 | 27078.6 |
| **4** | 2017-01-06 | 2017-01-06 | 27246.4 |
| **5** | 2017-01-07 | NaT | NaN |
| **6** | 2017-01-08 | NaT | NaN |
| **7** | 2017-01-09 | 2017-01-09 | 27311.2 |
| **8** | 2017-01-10 | 2017-01-10 | 27441.5 |
| **9** | 2017-01-11 | 2017-01-11 | 27599.0 |

data.all.tail(20)

OUTPUT:

| **date** | **Date** | **Prices** |
| --- | --- | --- |
| **1908** | 2022-03-25 | 2022-03-25 | 51687.5 |
| **1909** | 2022-03-26 | NaT | NaN |
| **1910** | 2022-03-27 | NaT | NaN |
| **1911** | 2022-03-28 | 2022-03-28 | 51712.5 |
| **1912** | 2022-03-29 | 2022-03-29 | 51637.5 |
| **1913** | 2022-03-30 | 2022-03-30 | 51687.5 |

data=data.all.drop(['Date'],axis=1)

**FILL NAN’S USING INTERPOLATE**

data['Prices']=data['Prices'].interpolate(method='linear',limit\_direction='forward')

data\_nomiss=data.copy()

data\_nomiss.head(20)

**OUTPUT:**

|  | date | Prices |
| --- | --- | --- |
| 0 | 2017-01-02 | 27918.500000 |
| 1 | 2017-01-03 | 26676.000000 |
| 2 | 2017-01-04 | 27091.900000 |
| 3 | 2017-01-05 | 27078.600000 |
| 4 | 2017-01-06 | 27246.400000 |
| 5 | 2017-01-07 | 27268.000000 |
| 6 | 2017-01-08 | 27289.600000 |
| 7 | 2017-01-09 | 27311.200000 |
| 8 | 2017-01-10 | 27441.500000 |
| 9 | 2017-01-11 | 27599.000000 |
| 10 | 2017-01-12 | 27877.600000 |

**CHECKING IF THERE ARE ANY NA VALUES**

data\_nomiss.isnull().sum()

**OUTPUT:**

date 0

Prices 0

dtype: int64

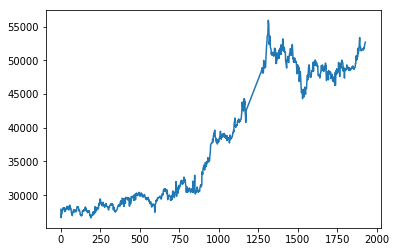
We can plot out this data quickly with matplolib

**PLOTTING THE DATA**

plt.plot(data\_nomiss.Prices)

**OUTPUT:**

[<matplotlib.lines.Line2D at 0x23b0f8b85c0>]

****

data\_nomiss['month']=data\_nomiss['date'].dt.month

data\_nomiss['year']=data\_nomiss['date'].dt.year

data\_nomiss.head()

**OUTPUT:**

date Prices month year

0 2017-01-02 27918.5 1 2017

1 2017-01-03 26676.0 1 2017

2 2017-01-04 27091.9 1 2017

3 2017-01-05 27078.6 1 2017

4 2017-01-06 27246.4 1 2017

**INDEXING EVERY DATE AND NAMING IT TIME SEQ**

data\_nomiss['time\_seq']=np.arange(1,len(data\_nomiss)+1)

data\_nomiss.tail()

data\_nomiss.head()

**OUTPUT:**

| date | Prices | month | year | time\_seq |
| --- | --- | --- | --- | --- |
| 1923 | 2022-04-09 | 52308.333333 | 4 | 2022 | 1924 |
| 1924 | 2022-04-10 | 52404.166667 | 4 | 2022 | 1925 |
| 1925 | 2022-04-11 | 52500.000000 | 4 | 2022 | 1926 |
| 1926 | 2022-04-12 | 52562.500000 | 4 | 2022 | 1927 |
| 1927 | 2022-04-13 | 52700.000000 | 4 | 2022 | 1928 |

**OUTPUT:**

| date | Prices | month | year | time\_seq |
| --- | --- | --- | --- | --- |
| 0 | 2017-01-02 | 27918.5 | 1 | 2017 | 1 |
| 1 | 2017-01-03 | 26676.0 | 1 | 2017 | 2 |
| 2 | 2017-01-04 | 27091.9 | 1 | 2017 | 3 |
| 3 | 2017-01-05 | 27078.6 | 1 | 2017 | 4 |
| 4 | 2017-01-06 | 27246.4 | 1 | 2017 | 5 |

fig=plt.subplots(figsize=(12,2))

ax=sns.boxplot(x=data\_nomiss['Prices'],whis=1.5)

**OUTPUT:**

****

**splitting into train and test**

x=data\_nomiss[['time\_seq']]

y=data\_nomiss['Prices']

data\_nomiss.set\_index('date',inplace=**True**)

train=data\_nomiss.loc[data\_nomiss.time\_seq<1446,]

test=data\_nomiss.loc[data\_nomiss.time\_seq>=1446]

print(train.shape)

print(test.shape)

**OUTPUT:**

(1445, 4)

(483, 4)

print(train.index.min())

print(train.index.max())

print(test.index.min())

print(test.index.max())

OUTPUT:

2017-01-02 00:00:00

2020-12-16 00:00:00

2020-12-17 00:00:00

2022-04-13 00:00:00

**PLOTTING TRAIN AND TEST DATASET**

fig, ax = plt.subplots()

lines = plt.plot(train.index, train.Prices, '-o')

plt.setp(lines, linewidth=0.5)

plt.xticks(rotation = 45)

plt.title('daily Price (Train)')

plt.xlabel('daily')

plt.ylabel('Price')

plt.legend(['Price'],loc = 'best')

plt.show()

fig, ax = plt.subplots()

lines = plt.plot(test.index, test.Prices, '-o')

plt.setp(lines, linewidth=0.5)

plt.xticks(rotation = 45)

plt.title('daily Price (test)')

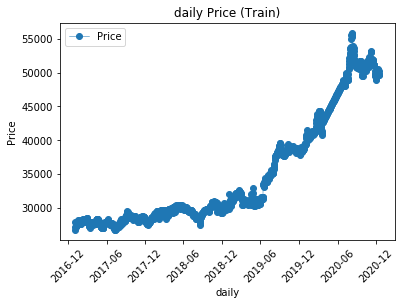
plt.xlabel('daily')

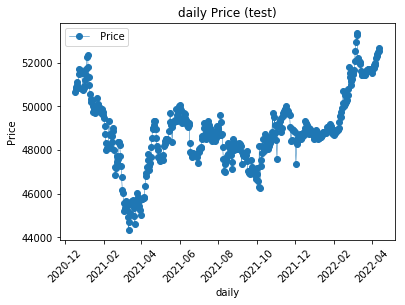
plt.ylabel('Price')

plt.legend([' Price'],loc = 'best')

plt.show()

**OUTPUT:**

****



DECOMPOSITION

Here we can see there is an upward trend. We can use statsmodels to perform a decomposition of this time series. The decomposition of time series is a statistical task that deconstructs a time series into several components, each representing one of the underlying categories of patterns. With statsmodels we will be able to see the trend, seasonal, and residual components of our data.

**from** **statsmodels.tsa.seasonal** **import** seasonal\_decompose

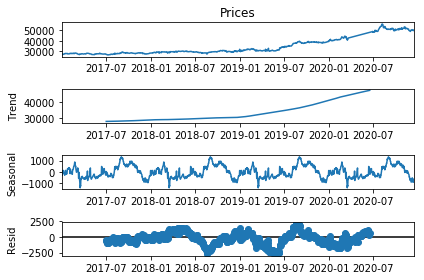
**from** **statsmodels.graphics.tsaplots** **import** plot\_acf,plot\_pacf

decomposition = seasonal\_decompose(train.Prices,freq=365)

decomposition.plot()

plt.show()

OUTPUT:



From the plot above we can clearly see the seasonal component of the data, and we can also see the separated upward trend of the data.

Trends can be upward or downward, and can be linear or non-linear. It is important to understand your data set to know whether or not a significant period of time has passed to identify an actual trend.

Irregular fluctuations are abrupt changes that are random and unpredictable.

**PLOTTING ACF AND PACF PLOTS**

*#### ACF: n th lag of ACF is the correlation between a day and n days before that.*

*#PACF: The same as ACF with all intermediate correlations removed.*

*#ACF*

plot\_acf(train.Prices, zero=**True**, lags=int(24\*0.5))

plt.axhline(y=-1.96/np.sqrt(len(train)), linestyle='--', color='gray')

plt.axhline(y=1.96/np.sqrt(len(train)), linestyle='--',color='gray')

*#PACF*

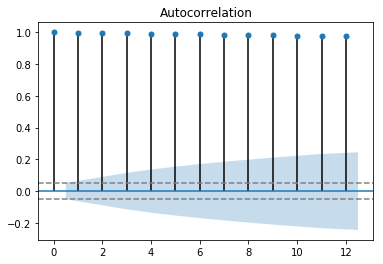
plot\_pacf(train.Prices,zero=**True**,lags=int(24\*0.5))

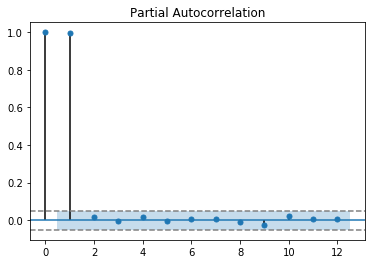
plt.axhline(y=-1.96/np.sqrt(len(train)),linestyle='--',color='gray')

plt.axhline(y=1.96/np.sqrt(len(train)),linestyle='--',color='gray')

plt.show()

**OUTPUT:**

****



*#### Looking at the Y scale in ACF we observe that both trend and seasonality is present.*

*#### Stationarize by differencing*

*# Before differencing*

plt.subplot(212)

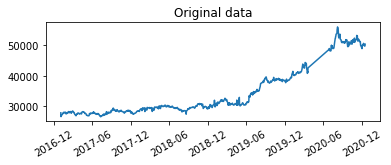
plt.plot(train.Prices)

plt.xticks(rotation=30)

plt.title("Original data")

plt.show()

OUTPUT:



*# After differencing*

plt.subplot(212)

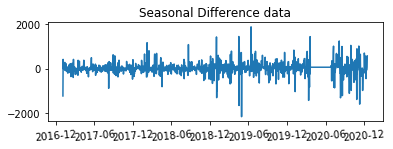
plt.plot(train.Prices.diff(1))

plt.xticks(rotation=365)

plt.title("Seasonal Difference data")

plt.show()

OUTPUT:



ACF AND PACF AFTER DIFFERENCING WITH 1

plot\_acf(train.Prices.diff(1).dropna(),zero=**True**,lags=12,title='Autocorrelation of differenced data')

plt.axhline(y=-1.96/np.sqrt(len(train)),linestyle='--',color='gray')

plt.axhline(y=1.96/np.sqrt(len(train)),linestyle='--',color='gray')

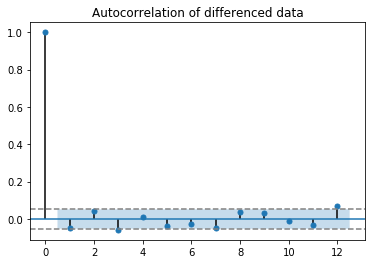
plot\_pacf(train.Prices.diff(1).dropna(),zero=**True**,lags=12,title='PartialAutocorrelation of differenced data')

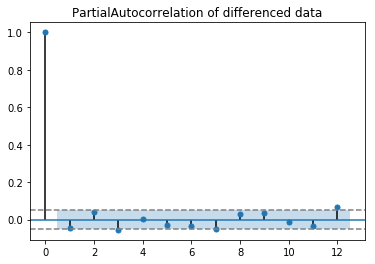
plt.axhline(y=-1.96/np.sqrt(len(train)),linestyle='--',color='gray')

plt.axhline(y=1.96/np.sqrt(len(train)),linestyle='--',color='gray')

plt.show()

OUTPUT:





**ADF TEST (DIFF1)**

**from** **statsmodels.tsa.stattools** **import** adfuller

result = adfuller(train.Prices.diff(1).dropna())

print('ADF Statistic: **%f**' % result[0])

print('p-value: **%f**' % result[1])

print('Critical Values:')

**for** key, value **in** result[4].items():

print('**\t%s**: **%.3f**' % (key, value))

OUTPUT:

ADF Statistic: -9.654067

p-value: 0.000000

Critical Values:

1%: -3.435

5%: -2.864

10%: -2.568

**ARIMA**

**from** **statsmodels.tsa.arima.model** **import** ARIMA

**from** **pmdarima.arima** **import** auto\_arima

model = auto\_arima(train.Prices, start\_p=0, start\_q=0)

model.summary()

**OUTPUT:**

|  |  |  |  |
| --- | --- | --- | --- |
| SARIMAX Results | | | |
| Dep. Variable: | y | No. Observations: | 1445 |
| Model: | SARIMAX(1, 1, 1) | Log Likelihood | -10140.964 |
| Date: | Wed, 27 Jul 2022 | AIC | 20289.928 |
| Time: | 18:01:32 | BIC | 20311.028 |
| Sample: | 0 | HQIC | 20297.803 |
|  | - 1445 |  |  |
| Covariance Type: | opg |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | coef | std err | z | P>|z| | [0.025 | 0.975] |
| intercept | 30.3004 | 13.876 | 2.184 | 0.029 | 3.104 | 57.497 |
| ar.L1 | -0.9479 | 0.030 | -31.956 | 0.000 | -1.006 | -0.890 |
| ma.L1 | 0.9132 | 0.037 | 25.019 | 0.000 | 0.842 | 0.985 |
| sigma2 | 7.38e+04 | 1156.861 | 63.792 | 0.000 | 7.15e+04 | 7.61e+04 |

|  |  |  |  |
| --- | --- | --- | --- |
| Ljung-Box (L1) (Q): | 0.01 | Jarque-Bera (JB): | 7604.27 |
| Prob(Q): | 0.93 | Prob(JB): | 0.00 |
| Heteroskedasticity (H): | 3.07 | Skew: | -0.46 |
| Prob(H) (two-sided): | 0.00 | Kurtosis: | 14.21 |

**Warnings:  
[1] Covariance matrix calculated using the outer product of gradients (complex-step).**

**p=1,q=1,d=1 and P=1,D=0,Q=1,m=12**

model\_sarima=ARIMA(train['Prices'].values,order=(1,1,1),seasonal\_order=(1,0,1,12))

model\_sarima\_fit=model\_sarima.fit()

print(model\_sarima\_fit.summary())

output:

SARIMAX Results

========================================================================================

Dep. Variable: y No. Observations: 1445

Model: ARIMA(1, 1, 1)x(1, 0, 1, 12) Log Likelihood -10139.729

Date: Wed, 27 Jul 2022 AIC 20289.459

Time: 23:58:00 BIC 20315.834

Sample: 0 HQIC 20299.303

- 1445

Covariance Type: opg

==============================================================================

coef std err z P>|z| [0.025 0.975]

------------------------------------------------------------------------------

ar.L1 -0.9423 0.036 -26.132 0.000 -1.013 -0.872

ma.L1 0.9096 0.043 21.310 0.000 0.826 0.993

ar.S.L12 -0.4941 0.234 -2.111 0.035 -0.953 -0.035

ma.S.L12 0.5566 0.223 2.493 0.013 0.119 0.994

sigma2 7.343e+04 1226.249 59.883 0.000 7.1e+04 7.58e+04

===================================================================================

Ljung-Box (L1) (Q): 0.03 Jarque-Bera (JB): 7008.39

Prob(Q): 0.86 Prob(JB): 0.00

Heteroskedasticity (H): 3.09 Skew: -0.41

Prob(H) (two-sided): 0.00 Kurtosis: 13.76

===================================================================================

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

**MODEL (1,1,1)(1,0,1,12) FITTING**

model\_sarima\_forecast = model\_sarima\_fit.forecast(steps = len(test))

print(model\_sarima\_forecast)

OUTPUT:

[50349.40616497 50353.87480443 50370.3580002 50426.20738549

50445.13587346 50373.85250726 50419.47651547 50425.45815603

50430.34692482 50399.65217492 50435.13839713 50531.35027452

50562.71744519 50569.43460552 50589.35100457 50594.9934014

50636.24982715 50602.804549 50659.09056483 50671.49647156

50682.7016738 50654.81455431 50687.14388543 50736.14280052

50767.94983551 50774.88875369 50795.14366614 50795.8352161

50839.29335311 50809.57929757 50866.91665465 50879.95608681

50891.78411599 50864.17388774 50896.19188725 50940.53519445

50972.38557163 50979.34638976 50999.63465026 50999.83803287

51043.51324671 51014.1671572 51071.60815387 51084.71009008

51096.59950333 51069.01661164 51101.00387845 51144.88813157

51176.74275022 51183.70576036 51203.99727683 51204.15255402

51247.84914122 51218.53936905 51275.99055311 51289.09868524

51300.99411911 51273.41395549 51305.39815935 51349.23717785

51381.09218235 51388.05544103 51408.34724618 51408.4978121

51452.19647455 51422.89031597 51480.34247225 51493.4512477

51505.34724292 51477.76738064 51509.75125014 51553.5858404

51585.44085063 51592.40416614 51612.69596745 51612.8461011

51656.54493589 51627.23916593 51684.6913858 51697.80025696

51709.69627527 51682.11647497 51714.10027924 51757.93446508

51789.78944364 51796.75279698 51817.04456569 51817.19468893

51860.8935085 51831.58780907 51889.04000301 51902.14891581

51914.0449042 51886.46514219 51918.44890786 51962.28308599

51994.13802926 52001.10141849 52021.39315183 52021.54330619

52065.24209212 52035.93643178 52093.38859104 52106.49754006

52118.39349339 52090.81376726 52122.79749703 52166.63170649

52198.48661419 52205.45003904 52225.74173682 52225.89192631…….]

PLOTTING THE MODEL

model\_forecast['SARIMA'] = model\_sarima\_forecast

plt.figure(figsize=(16,8))

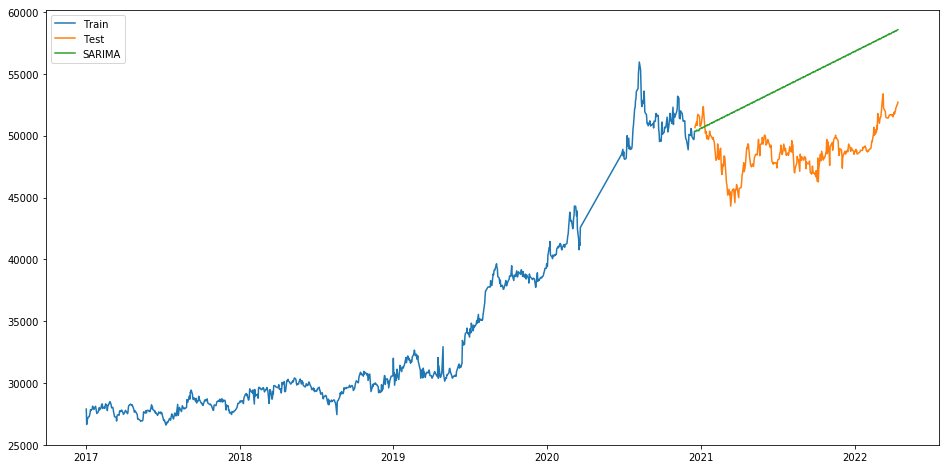
plt.plot( train['Prices'], label='Train')

plt.plot(test['Prices'], label='Test')

plt.plot(model\_forecast['SARIMA'], label='SARIMA')

plt.legend()

plt.show()

OUTPUT:

error\_metrics(train['Prices'], model\_sarima\_fit.forecast(steps=len(train)),test['Prices'],

model\_sarima\_fit.forecast(steps=len(test)))

OUTPUT:

\*\*\* Error metrics Train data \*\*\*

MAE Train: 27887.62587232186

MSE Train: 789983206.66629

RMSE Train: 28106.63990352262

MAPE Train: 0.847073159808068

\*\*\*

Error metrics Test Data \*\*\*

MAE Test: 5686.552461679636

MSE Test: 37070525.123097785

RMSE Test: 6088.55689988176

MAPE Test: 0.1170635592134034

model\_forecast

OUTPUT:

Prices month year time\_seq model\_arima\_forecast SARIMA

date

2020-12-17 50650.000000 12 2020 1446 50306.318778 50349.406165

2020-12-18 50792.500000 12 2020 1447 50321.676890 50353.874804

2020-12-19 50893.333333 12 2020 1448 50307.100018 50370.358000

2020-12-20 50994.166667 12 2020 1449 50320.935390 50426.207385

2020-12-21 51095.000000 12 2020 1450 50307.803799 50445.135873

... ... ... ... ... ... ...

2022-04-09 52308.333333 4 2022 1924 50314.198241 58457.603649

2022-04-10 52404.166667 4 2022 1925 50314.198241 58501.438920

2022-04-11 52500.000000 4 2022 1926 50314.198241 58533.292766

2022-04-12 52562.500000 4 2022 1927 50314.198241 58540.257252

2022-04-13 52700.000000 4 2022 1928 50314.198241 58560.547888

CHAPTER 4

**INTERPRETATIONS**

* After checking the data types we can see that the date is in object format. It was converted to datetime
* After plotting the decomposition graph we can see that there is a seasonality and trend in the data.
* After looking at the acf and pacf plots we can say that it is not-stationary. So, we have differenced it with1.
* In the adf test we can see that the p-value is less than 0.05 and the adf statistic is greater than critical value. With that we can conclude our data is stationary.
* The auto.arima function of python suggests that Sarimax(1,1,1) model suits our data.
* P values of the coeficients are less than 0.05 and AIC of this model is less when compared to other models.
* The Ljung-Box (L1) (Q) is the LBQ test statistic at lag 1 is, the Prob(Q) is 0.01, and the p-value is 0.93. Since the probability is above 0.05, we can’t reject the null that the errors are white noise.
* Heteroscedasticity tests that the error residuals are homoscedastic or have the same variance. The summary performs White’s test. Our summary statistics show a test statistic of 3.07 and a p-value of 0.00, which means we reject the null hypothesis and our residuals show variance.
* Jarque-Bera tests for the normality of errors. It tests the null that the data is normally distributed against an alternative of another distribution. We see a test statistic of 7604.27 with a probability of 0, which means we reject the null hypothesis, and the data is not normally distributed. Also, as part of the Jarque-Bera test, we see the distribution has a slight negative skew and a large kurtosis.

**CHAPTER 4**

**FINDINGS , SUGGESTIONS & CONCLUSIONS**

In this project we have tried to forecast gold prices. The project uses the daily time series data to discover the forecasting of gold prices. The study used SARIMA model to predict the gold price. The results show that SARIMAX (1,1,1)(1,0,1,12) model is a good fit for our data. The study has further used different forecasting techniques such as MAE, RSME, MAPE to determine the accuracy of the model. It resulted in low error. The AIC and BIC are low for this model when compared to other models. The p-value of coefficients is less than 0.05 which says that the coefficients are statistically significant. By looking at the Ljung-Box chi-square statistics we can say that the residuals are independent and that the model meets the assumption. Prices show an increasing trend for next year. The fluctuations show a linear increasing trend because the difference between the prices is low. It is steadily increasing.

Model selection plays a key role in forecasting. It may effect the results of the forecasts. Understanding the data is the most significant factor when choosing the appropriate forecasting model. Choose the correct model for your data to get more accurate results. Beware of overdifferencing your data which can manipulate your results.

**CONCLUSION**

In this project we have analysed the pattern of gold price fluctuations over the past 5 years. We have tried to build an effective and accurate forecasting model which could help financial institutions, investors, mining companies and related firms for decision making of when to buy and sell this commodity. This study uses Box-Jenkins ARIMA model to predict gold prices. The result show SARIMAX (1,1,1) is the best model for gold price prediction since BIC is low and error metrics is low. After fitting the model the predictions show that the gold price is steadily increasing.

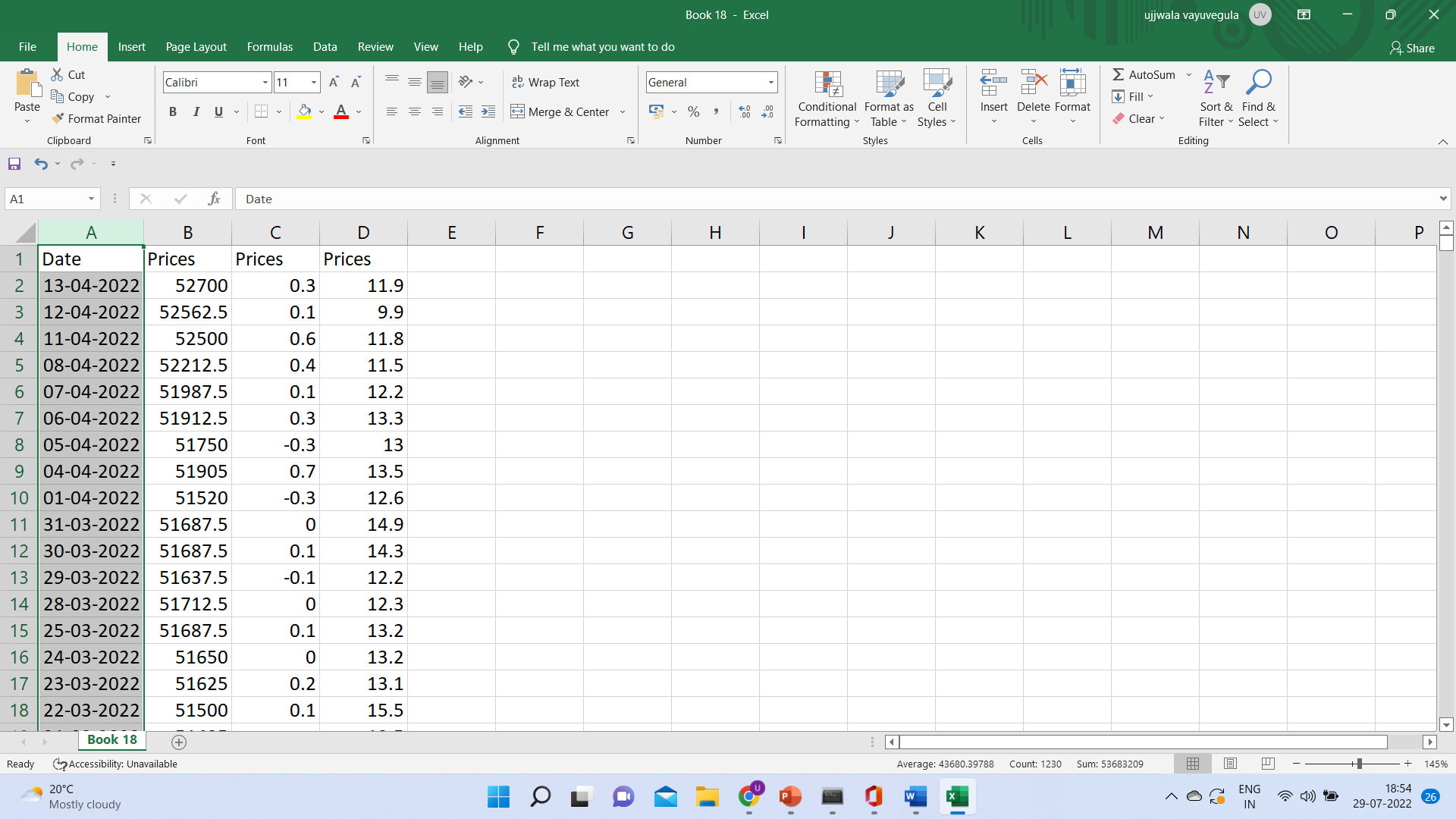
**CHAPTER 5**

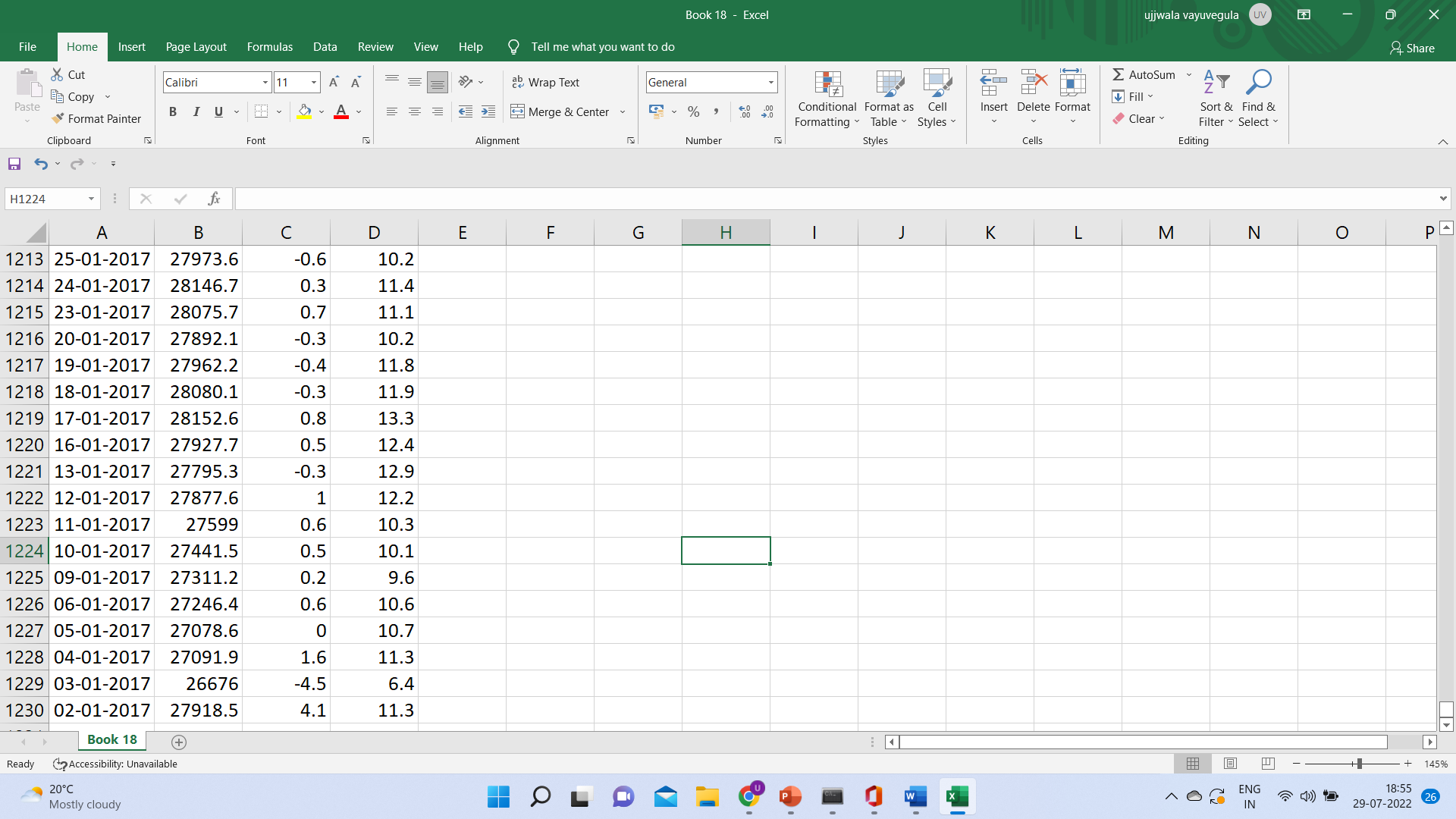
**BIBILOGRAPHY**

1. Banhi Guha and Gautam Bandyopadhyay (2016), Gold Price Forecasting Using ARIMA Model.
2. Xiaohui Yang\* IBSS Xi’an Jiaotong-Liverpool University Suzhou, China (2018), The Prediction of Gold Price Using ARIMA Model.
3. D Makala and Z Li 2021 J. Phys.: Conf. Ser. 1767 012022.
4. Naliniprava Tripathy (2017), Forecasting Gold Price with Auto Regressive Integrated Moving Average Model.
5. Subhasree Chatterjee (2018), Time series analysis using ARIMA model in R.
6. [**https://www.educba.com/time-series-analysis/**](https://www.educba.com/time-series-analysis/)
7. [**https://www.tableau.com/learn/articles/time-series-analysis**](https://www.tableau.com/learn/articles/time-series-analysis)
8. [**https://www.simplilearn.com/tutorials/data-science-tutorial/time-series-forecasting-in-r**](https://www.simplilearn.com/tutorials/data-science-tutorial/time-series-forecasting-in-r)
9. [**https://itl.nist.gov/div898/handbook/eda/section3/autocopl.htm**](https://itl.nist.gov/div898/handbook/eda/section3/autocopl.htm)
10. [**https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/**](https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/)
11. [**https://www.rdocumentation.org/packages/zoo/versions/1.8-9/topics/zoo**](https://www.rdocumentation.org/packages/zoo/versions/1.8-9/topics/zoo)
12. Dr. M. Massarrat Ali Khan (2013), Forecasting of Gold Prices (Box Jenkins Approach).
13. [**https://towardsdatascience.com/a-real-world-time-series-data-analysis-and-forecasting-121f4552a87**](https://towardsdatascience.com/a-real-world-time-series-data-analysis-and-forecasting-121f4552a87)
14. [**https://towardsdatascience.com/identifying-ar-and-ma-terms-using-acf-and-pacf-plots-in-time-series-forecasting-ccb9fd073db8**](https://towardsdatascience.com/identifying-ar-and-ma-terms-using-acf-and-pacf-plots-in-time-series-forecasting-ccb9fd073db8)
15. <https://towardsdatascience.com/a-real-world-time-series-data-analysis-and-forecasting-121f4552a87>
16. <https://towardsdatascience.com/identifying-ar-and-ma-terms-using-acf-and-pacf-plots-in-time-series-forecasting-ccb9fd073db8>

**ANNEXTURE**

DATA SET





LINK FOR THE DATA

[Book 18.csv](Book%2018.csv)