NTCC IN-HOUSE PROJECT REPORT

ON

EXPLORATORY DATA ANALYSIS ON SALES FORECASTING OF A STORE

A blue text on a white background

Description automatically generated with medium confidence

In partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Technology

Department of Computer Science and Engineering

Amity University, Uttar Pradesh

Submitted by

Vindhya Sree (A2305221413)

Under the guidance of

Dr. Renuka Arora

DECLARATION

I, Vindhya Sree, a student of Bachelors of Technology in Computer Science and Engineering, hereby declare that the project report entitled ‘Exploratory Data Analysis on sales forecasting of a store’ submitted to the Department of Computer Science and Engineering, ASET, Amity University, Noida, is a project report of the work done by me under the guidance of Dr. Renuka Arora. This work is submitted in partial fulfilment for the award of the degree of B. Tech CSE. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

Vindhya Sree

A2305221413

5CSE4Y

COMPLETION CERTIFICATE

On the basis of declaration submitted by student Vindhya Sree of B Tech (CSE), I hereby certify that the Inhouse Project titled ‘Exploratory Data Analysis on sales forecasting of a store’ which is submitted to Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science is an original contribution with existing knowledge and faithful record of work carried out by him under my guidance and supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

A picture containing text, handwriting, font, white

Description automatically generated

ACKNOWLEDGEMENT

It gives me great sense of pride and pleasure to present the project report undertaken during the six weeks of In-house project training. I owe special debt to my Project Faculty Guide Dr. Renuka Arora, Associate professor, ASET(CSE), Amity University Uttar Pradesh for her constant support and guidance throughout the course of my work. My deepest thanks to my Project Guide Dr. Renuka Arora, Associate professor, ASET(CSE), Amity University Uttar Pradesh for guiding and correcting various documents with attention.

I would like to express my special thanks of gratitude to my guide Dr. Renuka Arora and the institution ASET who gave me the opportunity and immense pleasure to carry out this project entitled ‘Exploratory Data Analysis on sales forecasting of a store’ and guided me throughout every doubt that arose during. This project helped me in gaining a lot of exposure and knowledge in the field of Machine Learning and Time-series forecasting, in general.

I would also like to thank my parents and peers who assisted me in finalizing and completing the project within a stipulated time frame.

Vindhya Sree

A2305221413

5CSE4Y

INDEX

Declaration ….………………………………………….……………………… 2

Completion Certificate…………………………………………………………... 3

Acknowledgement ……………………………………………………………….4

Index ……………………………………………………………………………. 5

Table of Figures ………………………………………………………………… 7

Abstract …………………………………………………………………………. 8

Literature Review ………………………………………………………………. 9

Chapter 1 : Introduction ………………………………………………………… 12

1.1 Essential library and module imports 12

1.2 Loading dataset 13

Chapter 2 : Data preprocessing ………………………………………………….. 13

2.1 Datetime format conversion and indexing 13

2.2 Plotting weekly sales over time 13

Chapter 3 : Rolling metrics and monthly moving averages ……………………... 14

3.1 Calculation of rolling mean and standard deviation 14

3.2 Plots of monthly moving averages [4, 6, 8, 12-months] 15

3.3 Visualisation of monthly sale trends over different years 16

Chapter 4 : Seasonal Decomposition …………………………………………….. 16

4.1 Plots of decomposition components 16

4.2 Concatenation of decomposition components 17

Chapter 5 : Generating forecasted values …………………………………………18

5.1 Installation of Prophet 18

5.2 Generation of forecasted values for future dates 18

5.3 Visualisation plots of forecasted time series 19

Chapter 6 : Dicky-Fuller Test ……………………………………………………..20

6.1 Dicky-fuller test for ‘sales’ data 20

6.2 Dicky-Fuller test for ‘sales\_log’ data 20

6.3 Dicky-Fuller test for ‘sales\_diff1’ data 21

6.4 Dicky-Fuller test for ‘sales\_log\_diff1’ data 22

6.4 Dicky-Fuller test for ‘sales\_log\_diff2’ data 22

Chapter 7 : Autocorrelation and partial autocorrelation charts ………………….. 23

Chapter 8 : Model summary analysis …………………………………………….. 23

Chapter 9 : Grid search for best parameter configuration for model fitting ………25

9.1 Model fitting with the best parameter configuration obtained 25

Chapter 10 : Calculation of error metrics ………………………………………….26

Chapter 11 : 95% and 99% confidence intervals …………………………………..26

Chapter 12 : Diagnostic plots and final output …………………………………….27

12.1 Final output with y\_smooth and y\_hat 28

Conclusion ………………………………………………………………………. 29

Future prospects ………………………………………………………………….. 30

Bibliography ……………………………………………………………………… 32

TABLE OF FIGURES

|  |  |
| --- | --- |
| List of Figures | Page |
| Fig 1. Line plot of weekly sales over time indicated by ‘sales’ | 14 |
| Fig 2. Sale trends over time analysis(4, 6, 8, 12 for monthly-moving averages) | 15 |
| Fig 3. Line plot visualisation of monthly sale trends over different years | 16 |
| Fig 4. Visual representation of contribution of each component to overall time-series | 17 |
| Fig 5. Plot of forecasted sales values for future dates | 19 |
| Fig 6. First plot displays forecasted sales over time. Second plot displays individual components of the forecasted time series. | 19 |
| Fig 7a. Dicky Fuller Test: Plot of ‘sales’ data | 20 |
| Fig 7b. Dicky Fuller Test: Plot of ‘sales\_log’ data | 21 |
| Fig 7c. Dicky Fuller Test: Plot of ‘sales\_diff1’ data | 21 |
| Fig 7d. Dicky Fuller Test: Plot of ‘sales\_log\_diff1’ data | 22 |
| Fig 7e. Dicky Fuller Test: Plot of ‘sales\_log\_diff2’ data | 22 |
| Fig 8. Autocorrelation and partial autocorrelation charts | 23 |
| Table 1 SARIMAX model results | 24 |
| Fig 9. Plot of forecast along with confidence interval band | 27 |
| Fig 10. Diagnostic Plot | 28 |
| Fig 11. Final output plot with y\_smooth and y\_hat | 29 |

ABSTRACT

This project report focuses on sales forecasting of a baking product for a superstore like Walmart using exploratory data analysis (EDA) technique in Python within a Jupyter Notebook. The motive is to analyse the historical sales data of the baking product and develop a reliable model for predicting future sales trends.

Initially, it involves importing essential Python libraries and loading the sales dataset obtained from the superstore. Exploratory Data Analysis (EDA) helps in exploring the dataset's structure and detecting missing values and based on the data then obtained, the summary statistics are generated. Then, the correlation analysis is performed to uncover any relationships between variables, providing valuable inferences and conclusions.

Visualizations play an important role in understanding the patterns of the sales. Various plots and charts are generated to visualize sales trends over time which aid in identifying patterns, outliers, trends and potential features influencing performance of the sale trends.

Further, the relationship between different features and the target variables is studied that serve as valuable inputs for further analysis and modelling. For sales forecasting, a time series forecasting model using the ARIMA (Autoregressive Integrated Moving Average) methodology is formulated. The model is trained on historical sales data and used to generate forecasts for the next 36 months. Then the forecasted sales are compared with the actual sales data to assess the model's performance.

This project report demonstrates the application of EDA in Python within a Jupyter Notebook, for sales forecasting of a baking product in a Walmart store dataset. The report also highlights the significance of exploratory data analysis, visualization, and the ARIMA model in understanding historical sales patterns and predicting future trends.

LITERATURE REVIEW

Time series forecasting is an important area of forecasting in which past observations of the same variable are collected and analyzed to develop a model describing the underlying relationship. The model is then used to extrapolate the time series into the future. This modeling approach is particularly useful when little knowledge is available on the underlying data generating process or when there is no satisfactory explanatory model that relates the prediction variable to other explanatory variables [26]. While the project focuses on exploratory data analysis (EDA), any statistical analysis is used to begin with an informal, exploratory examination of the data and this is called exploratory data analysis [28]. It is used for seeing what the data can tell us beyond formal modeling and thereby contrasts traditional hypothesis testing. The objectives of EDA are to:

* Enable unexpected discoveries in the data
* Suggest hypotheses about the causes of observed phenomena
* Assess assumptions on which statistical inference will be based
* Support the selection of appropriate statistical tools and techniques
* Provide a basis for further data collection through surveys or experiments

Seasonality is the presence of variations that occur at specific regular intervals less than a year such as weekly, monthly or quarterly and it may be caused by various factors such as weather, vacation and holidays and consists of periodic, repetitive and generally regular and predictable patterns in the levels of a time series [21].

The term stationary data indicates that data mean, covariance, and [autocorrelation](https://www.sciencedirect.com/topics/earth-and-planetary-sciences/autocorrelation) are constant for a time period of the time series [29]. Since stationarity is an assumption underlying many statistical procedures used in [time series analysis](https://en.wikipedia.org/wiki/Time_series_analysis), non-stationary data are often transformed to become stationary. The most common cause of violation of stationarity is a trend in the mean, which can be due either to the presence of a [unit root](https://en.wikipedia.org/wiki/Unit_root) or of a deterministic trend [24].

As the data is pre-processed and visualised, the data proves to be seasonal so SARIMAX model is used. Theoretically, an extension to ARIMA that supports the direct modeling of the seasonal component of the series is called SARIMAX with an exploratory variable. A rolling analysis of a time series model is often used to assess the model’s stability over time. A common technique to assess the constancy of a model’s parameters is to compute parameter estimates over a rolling window of a fixed size through the sample. If the parameters are truly constant over the entire sample, then the estimates over the rolling windows should not be too different. If the parameters change at some point during the sample, then the rolling estimates should capture this instability [31].

In time series analysis, the partial autocorrelation function (PACF) gives the partial correlation of a stationary time series with its own lagged values, regressed the values of the time series at all shorter lags. It contrasts with the autocorrelation function which does not control for other lags [22]. An autoregressive model (AR) is a representation of a type of random process which is used to describe certain time-varying processes in nature and specifies that the output variable depends linearly on its own previous values [26]. The moving-average model (MA model), also known as moving-average process, is a common approach for modeling [univariate](https://en.wikipedia.org/wiki/Univariate) time series. The moving-average model specifies that the output variable is [cross-correlated](https://en.wikipedia.org/wiki/Cross-correlation) with a non-identical to itself random-variable. Together with the [autoregressive (AR) model](https://en.wikipedia.org/wiki/Autoregressive_model), the moving-average model is a special case and key component of the more general [ARMA](https://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model) and [ARIMA](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average) models of [time series](https://en.wikipedia.org/wiki/Time_series) [26].

The seasonal 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. Time series are usually decomposed into:

* The [trend component](https://en.wikipedia.org/wiki/Trend_estimation) at time *t*, which reflects the long-term progression of the series. A trend exists when there is a persistent increasing or decreasing direction in the data. The trend component does not have to be linear.
* ��The cyclical component at time *t*, which reflects repeated but non-periodic fluctuations. The duration of these fluctuations depends on the nature of the time series.
* ��The seasonal component at time *t*, reflecting [seasonality](https://en.wikipedia.org/wiki/Seasonality). A seasonal pattern exists when a time series is influenced by seasonal factors. Seasonality occurs over a fixed and known period.
* ��The irregular component (or "noise") at time *t*, which describes random, irregular influences. It represents the residuals or remainder of the time series after the other components have been removed [25].

Then comes the concept of Dicky Fuller Test for stationarity. It tests the null hypothesis that a unit root is present in an autoregressive (AR) time series model. If the series is stationary then it has a tendency to return to a constant mean [25]. ADF stands for Augmented Dicky Fuller Test is fundamentally a statistical significance test which means there is a hypothesis testing involved with a null hypothesis and test statistic is computed and p-values are reported with which inferences can be made as to whether the time series is stationary or not.

ARIMA procedure requires the best parameter configuration to fit the model Non-seasonal ARIMA models are generally denoted ARIMA(p,d,q) where [parameters](https://en.wikipedia.org/wiki/Parameter) p, d, and q are non-negative integers, p is the order (number of time lags) of the [autoregressive model](https://en.wikipedia.org/wiki/Autoregressive_model), d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the [moving-average model](https://en.wikipedia.org/wiki/Moving-average_model). Seasonal ARIMA models are usually denoted ARIMA(p,d,q)(P,D,Q)m, where m refers to the number of periods in each season, and the uppercase P,D,Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model. A widely used metric to evaluate forecast model accuracy is the Mean Absolute Percentage Error (MAPE). It is intuitively appealing as it penalises under- and over-prediction relative to the actual outcome in a symmetrical way. The formula can be expressed as [23]:

A picture containing text, font, white, line

Description automatically generated

Where At is the actual value and Ft is the forecast value. Their difference is divided by the actual value At. The absolute value of this ratio is summed for every forecasted point in time and divided by the number of fitted points n [26].

**CHAPTER 1 : INTRODUCTION**

The project report is implemented to achieve exploratory data analysis (EDA) for sales forecasting of a baking product in a superstore like Walmart. The dataset is obtained from a trusted source like Kaggle. The generic process for ARIMA models are: Create a time series visualisation, make the time series stationary, plot autocorrelation and partial autocorrelation charts, build an ARIMA or SARIMAX model based on the seasonality of the data collected, and then use the trained model to predict sales for the following 36 months. The datasets acquired are known as ‘train.csv’, ‘test.csv’, ‘features.csv’ and ‘stores.csv’ having 45 stores and 3 types of baking products named as A, B and C with the respective sales.

**CHAPTER 1.1 : ESSENTIAL LIBRARY AND MODULE IMPORTS**

Initially, the necessary libraries and settings are set up and imported for conducting machine learning, statistical and time-series analysis tasks, particularly related to time series analysis. The ‘warning’ module allows us to control the display of warning messages. The ‘itertools’ module provides various functions for efficient looping and iteration. The 'Pandas' library is a well-known, strong data analysis and manipulation package. Large, multi-dimensional arrays and mathematical operations are supported by the 'NumPy' library. The pyplot module from the 'matplotlib' library is imported since it is a Python charting library for building static, animated, and interactive visualisations. Additionally, the Jupyter notebook-specific magic command '%matplotlib inline' permits the presentation of matplotlib plots right inside the notebook. The modules 'statsmodels.api', 'statsmodels.tsa.api', and 'statsmodels.formula.api' are imported because statsmodels is a Python library that offers a variety of statistical models and statistical tests, various functions and classes for analysing time-series data, and allows for the specification of statistical models using a formula syntax resembling that of R. Finally, plt.style.use(‘bmh’) sets the style of the plots generated by matplotlib to ‘bmh’ which stands for “Bayesian Methods for Hackers” and provides a visually pleasing style for data visualizations.

**CHAPTER 1.2 : LOADING DATASET**

Now, data is read from two CSV (Comma-separated values) files names ‘train.csv’ and ‘test.csv’ and variables are assigned to them named ‘train’ and ‘test’ respectively which can be used to access and manipulate the data in the files. It reads the contents of the files using the ‘read\_csv()’ function from the Pandas library. Data from the CSV files is loaded into Pandas DataFrame objects ‘train’ and ‘test’ which provide a tabular structure to work with the data. These DataFrames can be used for various machine learning and data analysis tasks like data preprocessing, feature engineering, model training and evaluation.

**CHAPTER 2 : DATA PREPROCESSING**

**CHAPTER 2.1 : DATETIME FORMAT CONVERSION AND INDEXING**

The 'train' and 'test' DataFrames both have a modified 'Date' column. The 'to\_datetime()' pandas function transforms the 'Date' column in the 'train' and 'test' DataFrames into a datetime data type, parses the input, and provides a Series with datetime data. Additionally, using '.dt.month', '.dt.year', and '.dt.dayofweek', new columns 'Month', 'Year', and 'Dayofweek' are formed in the 'train' and 'test' DataFrames. It is useful since it separates the month, year, and day of the week components from the "Date" column, respectively. To enable the DataFrame to be used as a time-series DataFrame, the dates are specified as the index. Date becomes the primary row identifier when the "Date" column is specified as the index. It is crucial for working with time-series data as it enables time-based indexing, slicing, and resampling operations and makes it simple to align data depending on dates. The time-series is extracted into the data object named ‘sales’ wherein index is date and values represent sum of weekly sales for each date by grouping ‘train’ DataFrame by ‘Date’ column and calculating sum of ‘Weekly\_Sales’ column for each unique date.

**CHAPTER 2.2 : PLOTTING WEEKLY SALES OVER TIME**

The line plot of sum of weekly sales over time is generated. Fig 1 visualises the trend and patterns in the sales data with the x-axis representing years and the y-axis representing weekly sales. The ‘sales’ series is calculated through grouping and summation of ‘Weekly\_Sales’ column.

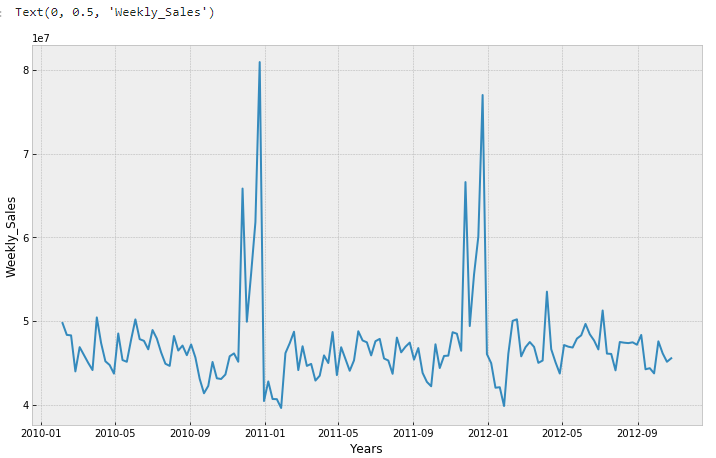


Fig 1. Line plot of weekly sales over time indicated by ‘sales’

**CHAPTER 3 : ROLLING METRICS AND MONTHLY MOVING AVERAGES**

**CHAPTER 3.1 : CALCULATION OF ROLLING MEAN AND STANDARD DEVIATION**

The rolling mean and standard deviation for ‘sales’ series is calculated using a window size of 12 and displayed in a tabular form. The ‘rolling()’ method creates a rolling window object and ‘.mean()’ calculates mean within each window [1]. The rolling mean actually provides a smoothed average value indicating the overall trend while the rolling standard deviation measures the dispersion/volatility around the mean. The ‘round()’ function is used to round the values to 4 decimal places and the resultant is assigned to ‘sales\_mean’ series. The rolling standard deviation of ‘sales’ series is also calculated using a window size of 12 with the ‘std()’ method computing standard deviation within each window resulting in values representing the average of the sales over the preceding 12 months and the ‘round()’ function rounding the values to 4 decimal places and assigning the resultant to ‘sales\_mean’ series. The trend and variability of sales data over time is analysed.

**CHAPTER 3.2 : PLOTS OF MONTHLY MOVING AVERAGES [4, 6, 8 AND 12 MONTHS]**

A 2x2 grid of subplots is created and plots of the original sales data are presented along with the different moving averages. In Fig 2, a set of subplots are arranged in 2x2 grids with the widths and heights using ‘set\_figheight()’ and ‘set\_figwidth()’ functions. Each display the sales data and the 4-month, 6-month, 8-month and 12-month moving averages with the help of the ‘rolling(window=4).mean()’ function which calculates the rolling mean with a window size of 4 months and plots the values. A legend is also added to the subplots positioning it at the ‘best’ location. Similarly, the subplots for different moving averages like 6 months, 8 months and 12 months are configured and plotted alongside original sales data with appropriate labels, titles and legends set for each subplot. Fig 2 shows sale trends over time analysis with different window sizes 4, 6, 8, 12 months for moving averages.

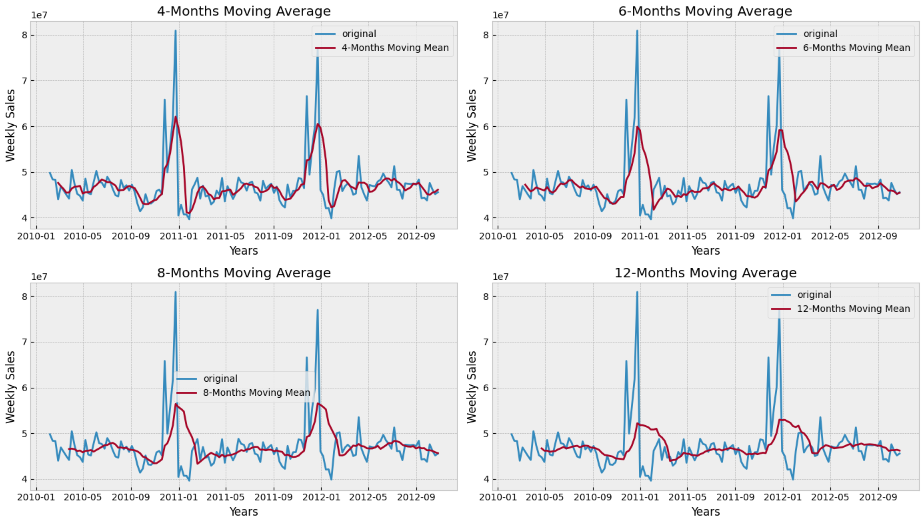


Fig 2. Sale trends over time analysis(Window sizes 4, 6, 8, 12 for monthly moving averages)

**CHAPTER 3.3 : VISUALISATION OF MONTHLY SALE TRENDS OVER DIFFERENT YEARS**

Next, a pivot table is created from the ‘train’ DataFrame to analyze monthly sales data and then the monthly sales trend is plotted. The Pandas library function ‘pivot\_table()’ creates a pivot table by aggregating weekly sales values based on year and month columns and rows are ordered by the months of the year from January (1) to December (12) to ensure that monthly sales are plotted in the correct chronological order. Fig 3 visualises monthly sale trends over different years and the variation of sales across months of the year to observe any seasonal patterns or trends.

A picture containing plot, text, line, diagram

Description automatically generated

Fig 3. Line plot visualisation of monthly sale trends over different years

**CHAPTER 4 : SEASONAL DECOMPOSITION**

**CHAPTER 4.1 : PLOTS OF DECOMPOSITION COMPONENTS**

The seasonal decomposition of the additive ‘sales’ time-series is performed using ‘seasonal\_decompose()’ function from the ‘statsmodels.tsa’ module with the parameter explolate\_trend set to 8 allowing extrapolation of the trend components to handle missing values at the beginning and the end of time-series. The decomposition components are plotted displaying the observed values, the trend component, the seasonal component and the residual component in Fig 4. Theoretically, the trend component represents the long-term pattern or direction of the series, the seasonal component shows the repetitive pattern occurring within each season, and the residual component represents the random variation remaining after removing the trend and seasonal components.

A picture containing text, line, screenshot, plot

Description automatically generated

Fig 4. Visual representation of contribution of each component to the overall time-series

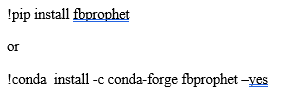
**CHAPTER 4.2 : CONCATENATION OF DECOMPOSITION COMPONENTS**

A DataFrame ‘dec\_output’ is created which consolidates the decomposition components obtained from seasonal decomposition i.e first concatenation of decomposition components and then assigning of column names ‘Observed’, ‘Trend’, Seasonal’, ‘Irregular’ to the columns of ‘dec\_output’ DataFrame using columns attribute. Next, the Time Series Index is calculated by multiplying ‘Trend’, ‘Seasonal’ and ‘Irregular’ columns element-wise, assigned to a new column ‘TSI’ in the DataFrame and dec\_output is displayed.

**CHAPTER 5 : GENERATING FORECASTED VALUES**

**CHAPTER 5.1 : INSTALLATION OF PROPHET**

Prophet is a method for time-series forecasting that uses an additive model to suit non-linear trends with seasonality that occurs annually, monthly, daily, and on weekends as well as during holidays. Moving on, the command is used to install the fbprophet library in the code cell:



The ‘Date’ column is grouped with ‘train’ DataFrame and sum of ‘Weekly\_Sales’ for each date is calculated. A new column is created named ‘ds’ that contains index values (dates) of the grouped DataFrame. Next, the ‘train’ DataFrame is modified and updated to include only ‘ds’ and ‘Weekly\_Sales’ columns (target variable renamed as ‘y’)

**CHAPTER 5.2 : GENERATION OF FORECASTED VALUES FOR FUTURE DATES**

A Prophet model object named ‘sales\_model’ is created and ‘seasonality\_mode’ is set to additive indicating that the model will consider additive seasonality. The interval\_width parameter is set to 0.95 specifying a 95% confidence interval for forecasted values. The model is then fit to the ‘train’ DataFrame and future dates for forecasting are generated. The trained model is applied to make predictions for those dates and the forecasted sales are stored in ‘sales\_forecast’ DataFrame. The make\_future\_dataframe() method of the sales\_model object is used to generate these dates. The periods parameter is set to 36, indicating the number of periods (in this case, months) to generate. The ‘freq’ parameter is set to 'MS' (which stands for month start) signifying that the generated dates will be the first day of each month. The ‘predict’ method of the sales\_model object is used to generate the forecasted sales values for the future dates in Fig 5.

A picture containing text, plot, line, screenshot

Description automatically generated

Fig 5. Plot of forecasted sales values for future dates

**CHAPTER 5.3 : VISUALISATION PLOTS OF FORECASTED TIME SERIES**

In Fig 6, two plots are generated to visualize the forecasted sales and the components of forecasted time-series using the Prophet library. The forecasted sales from the ‘sales\_forecast’ DataFrame are plotted using plot() method of ‘sales\_model’ object. The ‘sales\_forecast’ DataFrame contains the forecasted dates and corresponding sales predictions. The first plot shows forecasted sales over time and the second one displays individual components like trend, seasonality, etc. of the forecasted time series. The plots provide enough evidence about forecasted sales patterns and contributions of individual components to overall forecast plot.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated with low confidence

Fig 6. First plot displays forecasted sales over time. Second plot displays individual components of the forecasted time series.

**CHAPTER 6 : DICKY-FULLER TEST**

**CHAPTER 6.1 : DICKY-FULLER TESTS FOR ‘SALES’ DATA**

The Dicky-Fuller test is performed for stationarity on the ‘sales’ time series data. The test results are printed and the ‘sales’ data is plotted in Fig 7a. The adfuller function from statsmodels.tsa.stattools module is imported that is used to perform Dicky-Fuller test for stationarity and ‘sales’ is passed as an argument to the function. Then a Pandas series ‘dfoutput’ is created including first 4 elements of the test results and index is set to descriptive labels for each element like ‘ADF Test Statistic’, ‘p-values’, ‘Lags used’ and ‘Number of observations used’.

A picture containing text, line, plot, font

Description automatically generated

Fig 7a. Dicky Fuller Test: Plot of ‘sales’ data

**CHAPTER 6.2 : DICKY-FULLER TESTS FOR ‘SALES\_LOG’ DATA**

The ‘sales’ time series data is transformed into its logarithmic form ‘sales\_log’ using the ‘np.log10()’ function from the NumPy library and the logarithmic data is plotted in Fig 7b. The first order difference of the ‘sales’ data is calculated, the NaN values are removed resulting from differencing, the differenced data is plotted and then the Dicky-Fuller test is performed on the ‘sales\_log’ data to assess stationarity.

A picture containing text, line, plot, screenshot

Description automatically generated

Fig 7b. Dicky Fuller Test: Plot of ‘sales\_log’ data

**CHAPTER 6.3 : DICKY-FULLER TESTS FOR ‘SALES\_DIFF1’ DATA**

The first order differences of the ‘sales’ data are calculated, the NaN values resulting from differencing are removed, the differenced data is plotted in Fig 7c and then the Dicky-Fuller test is performed on the ‘sales\_diff1’ data to assess stationarity.

A picture containing text, plot, line, screenshot

Description automatically generated

Fig 7c. Dicky Fuller Test: Plot of ‘sales\_diff1’ data

**CHAPTER 6.4 : DICKY-FULLER TESTS FOR ‘SALES\_LOG\_DIFF1’ DATA**

The first order differences of the ‘sales’ data are calculated, the NaN values resulting from differencing are removed, the differenced data is plotted in Fig 7d and then the Dicky-Fuller test is performed on the ‘sales\_log\_diff1’ data to assess stationarity.

A picture containing text, plot, line, screenshot

Description automatically generated

Fig 7d. Dicky Fuller Test: Plot of ‘sales\_log\_diff1’ data

**CHAPTER 6.5 : DICKY-FULLER TESTS FOR ‘SALES\_LOG\_DIFF2’ DATA**

The first order differences of the ‘sales’ data are calculated, the NaN values resulting from differencing are removed, the differenced data is plotted in Fig 7e and then the Dicky-Fuller test is performed on the ‘sales\_log\_diff2’ data to assess stationarity.

A picture containing text, line, handwriting, plot

Description automatically generated

Fig 7e. Dicky Fuller Test: Plot of ‘sales\_log\_diff2’ data

**CHAPTER 7 : AUTOCORRELATION AND PARTIAL AUTOCORRELATION CHARTS**

Fig 8 shows the autocorrelation and partial autocorrelation plots of the'sales\_log\_diff2' data after the Dicky-Fuller test is used to analyse stationarity and evaluate its stationarity qualities. The 'lags' parameter specifies the number of lags to include in the ACF plot and the PACF plot, respectively, and the 'alpha' parameter regulates the transparency of the plot markers. These parameters are used to plot the autocorrelation function and partial autocorrelation function in Fig 8. The ACF plot can be used to find time series with substantial lag correlations. The PACF plot aids in determining the direct correlation between observations made at various lags while consideration the impact of intermediate lags.

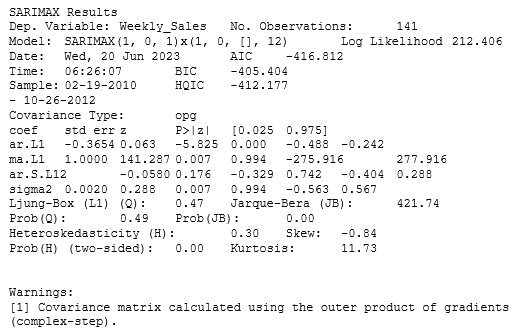
A picture containing line, plot, text, diagram

Description automatically generated

Fig 8. Autocorrelation and partial autocorrelation charts

**CHAPTER 8 : MODEL SUMMARY ANALYSIS**

Now, model fitting and summary analysis of a SARIMA model is performed as the data proves to be seasonal from the inferences derived beforehand. The ‘order’ parameter specifies the non-seasonal p, d, q order of the model and the ‘seasonal\_order’ parameter specifies the seasonal P, D, Q order of the model. The SARIMA model has an AR(1) component, an MA(1) component and a seasonal component with a seasonal differencing of order 12 and no seasonal autoregressive or moving average components. The ‘enforce\_stationarity’ and ‘enforce\_invertibility’ parameters are set to False to avoid enforcing stationarity and invertibility constraints on the model. Then a model summary is generated of the fitted SARIMA model including information on model coefficients, standard errors, p-values and other important metrics and helps in model appropriation. Then a list of values [0,1] is created for both non-seasonal AR (p) and MA (q) components and for the differencing parameter of SARIMA model. It defines range of possible values to be used for grid searching the optimal model parameters. All possible combinations of values in p, d, q are generated using the ‘itertools.product()’ function in form of list of tuples. Similarly, the same is done for the seasonal components of the SARIMA model. Table 1 shows the results of the SARIMAX model and information related to it and the summary analysis.



**A picture containing text, screenshot, font, number

Description automatically generated**

Table 1. SARIMAX model results

**CHAPTER 9 : GRID SEARCH FOR BEST PARAMETER CONFIGURATION FOR MODEL FITTING**

The SARIMA model's parameters (p, d, q, P, D, and Q) are searched on a grid to determine which configuration has the lowest Akaike Information Criterion (AIC). Better-fitting models are indicated by lower AIC values. To do this, a variable ‘best\_aic’ is initialized to positive infinity and it stores the lowest AIC value found during the grid search. The variables ‘best\_pdq’ and ‘best\_seasonal\_pdq’ are initialised to ‘None’ and they store the best parameter values for the non-seasonal and seasonal components of the SARIMA model. The list of seasonal and non-seasonal parameter combinations ‘seasonal\_pdq’ and ‘pdq’ is iterated over. The SARIMA model is then fit with the current parameter combination using the variable ‘temp\_model’ which stores temporarily the SARIMA model object during grid search. If the model fitting is successful, the AIC value is calculated and compared to the current best AIC value. If the current model's AIC value is lower than the most recent best AIC value, the variables "best\_aic," "best\_pdq," and "best\_seasonal\_pdq" are changed. After the grid search, the best model configuration found including parameter values and corresponding AIC values.

**CHAPTER 9.1 : MODEL FITTING WITH THE BEST PARAMETER CONFIGURATION OBTAINED**

The fitting is performed by first creating a SARIMA model with the best parameter configuration ‘best\_model’ and the model is fit using the fit() function. The fitted SARIMA model’s summary tables contains information about model coefficients. The ‘pred\_dynamic’ variable generates dynamic predictions using the fitted model and then it is used with predicted\_mean() function to extract predicted mean values from dynamic predictions which are the forecasted values for the specific time period. The predicted mean values are transformed back to their original scale by taking the power of 0. This is necessary as the ‘sales\_log’ variable was transformed using the logarithm (base 10) to stabilize the variance. The actual sales data is extracted from ‘2012-09-14’ onwards corresponding to the time period for which the dynamic predictions were made.

**CHAPTER 10 : CALCULATION OF ERROR METRICS**

The Mean Absolute Percentage Error (MAPE) for last two years of the forecast is calculated which measures the average percentage difference between the actual and forecasted values. The Mean Squared Error (MSE) value for the forecasts is calculated which measures the average squared difference between the actual and forecasted values. Then, the Root Mean Squared Error (RMSE) value for the forecasts is calculating by taking the square root of the MSE value. Future forecasts for the next 12 time steps are generated using the fitted SARIMA model ‘best\_results’.

A picture containing text, font, screenshot, line

Description automatically generated

**CHAPTER 11 : 95% AND 99% CONFIDENCE INTERVALS**

The variable ‘n\_steps’ are specified as the number of future time steps for which forecasts are generated. The forecasts are generated for the next 36 time steps using the fitted SARIMA model ‘best\_results’ with a confidence level of 99% and 95% and it is stored in the variables ‘pred\_uc\_99’ and ‘pred\_uc\_95’. The get\_forecast() method is used here with the ‘steps’ parameter set to 36 and ‘alpha’ parameter set to 0.01 and 0.05. The confidence intervals for 99% and 95% confidence level forecasts are extracted, the conf.int() method is applied to ‘pred\_uc\_99’ and ‘pred\_uc\_95’ objects and stored in ‘pred\_ci\_99’ and ‘pred\_ci\_95’ variables.

The confidence intervals are processed which are obtained from SARIMA model forecasts and DataFrames are created to store the forecasted values along lower and upper confidence intervals for both 95% and 99% confidence levels. This is performed by first specifying the number of future time steps for which forecasts are generated. Then a date range starting from last date in the ‘sales’ DataFrame and consisting of the number of periods with a monthly frequency set to ‘MS’. It creates a datetime index for the forecasted values. The DataFrames ‘fc\_95’ and ‘fc\_99’ are created that combines forecasted values with lower and upper confidence intervals for 95% and 99% confidence levels. This data is stacked column-wise and indexed with ‘idx’ datetime index. The columns are labelled as ‘forecast’, ‘lower\_ci\_95’, ‘upper\_ci\_95’ and ‘forecast’, ‘lower\_ci\_99’, ‘upper\_ci\_99’.

*A picture containing screenshot, text, plot, line

Description automatically generated*

Fig 9. Plot of forecast along with confidence interval band

The forecasted values and confidence intervals from both 95% and 99% confidence levels are combined into a single DataFrame ‘fc\_all’. The columns are reordered to have ‘forecast’ as the first column followed by the lower and upper confidence intervals for the 95% level and 99% level. Then necessary converters for matplotlib are registered to handle certain datetime objects. The observed sales data from the ‘sales’ DataFrame is plotted in Fig 9 and forecasted values from the ‘fc\_all’ DataFrame on the same figure axis with label as ‘Forecast’ and an alpha value set to 0.7 for transparency. The area between lower and upper confidence intervals for the 99% confidence level in Fig 9 is filled with a black color and an alpha value set to 0.25 indicating uncertainty range.

**CHAPTER 12 : DIAGNOSTIC PLOTS AND FINAL OUTPUT**

Finally, the diagnostic plots for the SARIMA model stored in ‘best\_results’ including standardised residual plots, histograms normal Q-Q plot and correlogram are plotted in Fig 10. The ‘lags’ parameter is set to 30 specifying number of lags to be included in correlogram plot. Exponential Smoothing is also incorporated here as it gives accurate and reliable forecasts to predict the next period. ExponentialSmoothing model is imported from statsmodels.tsa.holtwinters module. The best smoothing parameter values of alpha, beta and gamma are taken to be 0.4, 0.2 and 0.01. The model is initialised with the specified parameters like trend and seasonal set to ‘add’ and seasonal\_periods set to 12. Then the ExponentialSmoothing model is fit to the sales data and ‘optimized’ parameter set to True allowing the model to automatically optimize the smoothing parameters. The forecasted values ‘yh’ are generated for specified number of periods that is 36 months ahead using the fitted model. The original sales data is plotted with the label ‘Weekly\_Sales’. After that, the smoothed values obtained from the ExponentialSmoothing model and then the forecasted values are plotted with label ‘y\_smooth’ and ‘y\_hat’ respectively in Fig 11 achieving the final output.

A picture containing text, plot, line, diagram

Description automatically generated

A picture containing text, line, plot, screenshot

Description automatically generated

Fig 10. Diagnostic Plot

**CHAPTER 12.1 : FINAL OUTPUT**

A picture containing text, screenshot, line, plot

Description automatically generated

Fig 11. Final output plot with y\_smooth and y\_hat

CONCLUSION

In conclusion, the project report is centered around analysing the weekly sales data of a baking product of a superstore and developing a sales forecasting model. Through the exploration and examination of the data, several important findings and inferences were uncovered.

The analysis revealed a clear upward trend in weekly sales over time. This trend was effectively captured using a 12-month moving average, which exhibited a nearly linear pattern. This report also suggests that a linear regression model could be utilized to estimate the trend in the data accurately. Distinct seasonal patterns were identified within the data. The months of June, November, and December emerged as the peak months for weekly sales, displaying higher mean values compared to other months. While, January and October were identified as the months with the lowest sales performance.

It was also observed that the data series was non-stationary. This was concluded by the increasing trend in the rolling mean and the significantly low test statistic in comparison to the critical values. Additionally, a decomposition analysis of the time series revealed the presence of seasonality, trend, and irregular remainder components. The seasonal plot displayed a consistent month-to-month pattern, while the irregular remainder component exhibited certain patterns with high variation at the edges of the data.

With regard to the superstore's weekly sales data, the research offered insightful information. These discoveries can be used to strengthen decision-making procedures and create reliable sales forecasting models. The superstore may efficiently meet customer requests by optimising inventory management, marketing tactics, and resource allocation by comprehending the underlying trends, seasonality, and patterns in the data. The results of this analysis provide a strong basis for further investigation and improvement of sales forecasting methods, empowering the superstore to make wise business decisions and spur expansion in the cutthroat retail sector.

FUTURE PROSPECTS

Exploring the potential uses of exploratory data analysis (EDA) with the ARIMA model for sales forecasting gives intriguing opportunities from a research and development standpoint. Some of the crucial areas of interest for upcoming research on the same are as follows:

1. Optimisation of parameters: The method of optimising the ARIMA model's parameters with EDA still has to be further studied. In order to find the ideal values for the autoregressive (p), differencing (d), and moving average (q) components more efficiently, researchers might investigate sophisticated methodologies for analysing autocorrelation and partial autocorrelation plots. The accuracy and efficiency of the sales forecasting model will be improved as a result of this research.
2. Exogenous Variables Incorporation: Exogenous variables are variables whose cause is external to the model and whose role is to explain other variables or outcomes in the model. These variables can be more fully incorporated into the ARIMA model in future studies. Researchers can investigate techniques for successfully integrating exogenous factors by using EDA to find and analyse pertinent datasets, such as economic indicators, marketing tactics, or social media trends. With the help of this study, thorough and precise sales can be estimated and improved with our understanding of how outside influences affect sales.
3. Promotion of Seasonal ARIMA Modelling: Researchers might concentrate on extending the ARIMA model to Seasonal ARIMA (SARIMAX) modelling by building on the findings from the EDA. Advances in determining the proper seasonal order (P, D, Q) for SARIMAX modelling can be made by examining the seasonal trends contained in sales data utilising EDA approaches. This study will improve our capacity to identify and predict seasonal fluctuations in sales, resulting in more precise forecasts.
4. Model Evaluation and Improvement: Using EDA, researchers can investigate evaluation methods for ARIMA models in more depth. Researchers can contribute to the continued improvement of sales forecasting accuracy by creating unique ways for evaluating model performance, such as sophisticated residual analysis procedures or error measuring methodologies. This study will offer suggestions for improving the model and insights into potential areas for model improvement.
5. Visualising forecasts: Research efforts could concentrate on creative methods for displaying and communicating ARIMA-based sales projections utilising EDA approaches. It can be easier to comprehend and evaluate predicted sales patterns by experimenting with advanced visualisation techniques, interactive dashboards, or dynamic forecasting visualisations. This study will aid in making wise decisions and make it easier to explain forecast outcomes to stakeholders.
6. Continuous Data Exploration and Adaptation: Researchers should stress the significance of continuous data exploration by employing EDA methods. Iterative data analysis allows researchers to track sales data over time, spot changing trends and developing patterns. The ARIMA model will be made more proactive and adaptable as a result of this study, guaranteeing that it is still applicable and trustworthy in changing corporate situations.

EDA and the ARIMA model can be used to enhance the field of sales forecasting. The accuracy, model performance, and efficient use of sales forecasting in practical commercial applications will all benefit from this research and development viewpoint.

BIBLIOGRAPHY

1. <https://towardsdatascience.com/dont-miss-out-on-rolling-window-functions-in-pandas-850b817131db>
2. <https://medium.com/@DataSpace.NG/data-transformation-with-88-missing-data-supervised-learning-12c15bf7bae>
3. <https://www.geeksforgeeks.org/tag/python-matplotlib/page/67/>
4. <https://www.golinuxcloud.com/pandas-dataframe-rolling/>
5. <https://ijcrt.org/papers/IJCRT22A6150.pdf>
6. <http://facebook.github.io/prophet/>
7. <https://github.com/kirtigupta10007/Walmart-Store-Sales-Forecasting/blob/master/Walmart%20Stores%20Sales%20Prediction-%20Forecasting.ipynb>
8. <https://rdrr.io/cran/prophet/man/make_future_dataframe.html>
9. <https://towardsdatascience.com/null-hypothesis-and-the-p-value-fdc129db6502>
10. <https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide>
11. <https://analyticsindiamag.com/quick-way-to-find-p-d-and-q-values-for-arima/>
12. <https://stackoverflow.com/questions/67601211/futurewarning-statsmodels-tsa-arima-model-arma-and-statsmodels-tsa-arima-model>
13. <https://towardsdatascience.com/introduction-to-arima-for-time-series-forecasting-ee0bc285807a>
14. <https://towardsdatascience.com/interpreting-acf-and-pacf-plots-for-time-series-forecasting-af0d6db4061c>
15. <https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.acf.html>
16. <https://github.com/krishnaik06/ARIMA-And-Seasonal-ARIMA/blob/master/Untitled.ipynb>
17. <https://www.scribbr.com/statistics/akaike-information-criterion/>
18. <https://www.kaggle.com/code/bharatmadan/walmart-sales-prediction-and-eda>
19. <https://www.relataly.com/forecasting-beer-sales-with-arima-in-python/2884/>
20. <https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/>
21. <https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/>

1. <https://en.wikipedia.org/wiki/Seasonality>
2. <https://en.wikipedia.org/wiki/Partial_autocorrelation_function>
3. <https://en.wikipedia.org/wiki/Mean_absolute_percentage_error>
4. <https://en.wikipedia.org/wiki/Stationary_process>
5. <https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average>
6. Fattah J, Ezzine L, Aman Z, El Moussami H, Lachhab A. Forecasting of demand using ARIMA model. International Journal of Engineering Business Management. 2018;10. doi:10.1177/1847979018808673
7. G.Peter Zhang (2003) Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing, Volume 50, 2003, Pages 159-175, ISSN 0925-2312,

<https://doi.org/10.1016/S0925-2312(01)00702-0>. (<https://www.sciencedirect.com/science/article/pii/S0925231201007020>)

1. John D. Kechagias, Kyriaki-Evangelia Aslani, Nikolaos A. Fountas, Nikolaos M. Vaxevanidis, Dimitrios E. Manolakos, A comparative investigation of Taguchi and full factorial design for machinability prediction in turning of a titanium alloy, Measurement, Volume 151, 2020, 107213, ISSN 0263-2241, <https://doi.org/10.1016/j.measurement.2019.107213>.

(<https://www.sciencedirect.com/science/article/pii/S0263224119310796>)

1. Hmeda Musbah, Hamed H. Aly, Timothy A. Little, A proposed novel adaptive DC technique for non-stationary data removal, Heliyon, Volume 9, Issue 3, 2023, e13903, ISSN 2405-8440,

<https://doi.org/10.1016/j.heliyon.2023.e13903>. (<https://www.sciencedirect.com/science/article/pii/S2405844023011106>)

1. Zivot, E., Wang, J. (2003). Rolling Analysis of Time Series. In: Modeling Financial Time Series with S-Plus®. Springer, New York, NY.

<https://doi.org/10.1007/978-0-387-21763-5_9>

1. Jordi McKenzie, Mean absolute percentage error and bias in economic forecasting, Economics Letters, Volume 113, Issue 3, 2011, Pages 259-262, ISSN 0165-1765, <https://doi.org/10.1016/j.econlet.2011.08.010> (<https://www.sciencedirect.com/science/article/pii/S0165176511003119>)
2. H. Jiang, J. Ruan and J. Sun, "Application of Machine Learning Model and Hybrid Model in Retail Sales Forecast," 2021 IEEE 6th International Conference on Big Data Analytics (ICBDA), Xiamen, China, 2021, pp. 69-75,

doi: 10.1109/ICBDA51983.2021.9403224.