A black background with white text

Description automatically generated with low confidence

**MSc Data Science Project**

**7PAM2002-0509-2023**

**Department of Physics, Astronomy and Mathematics**

**Data Science FINAL PROJECT REPORT**

**Analysing the role of store characteristics and economic factors in Walmart Sales using Machine Learning**

**Student Name and SRN:**

Milu Paul, 21063319

Supervisor: Michael Kuhn

Date Submitted: 29/08/2024

Word Count: 7,334**DECLARATION STATEMENT**

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at [Assessment Offences and Academic Misconduct](https://www.herts.ac.uk/__data/assets/pdf_file/0007/237625/AS14-Apx3-Academic-Misconduct-v17.0.pdf) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6). I have not used chatGPT, or any other generative AI tool, to write the reportor code (other than where declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Milu Paul

Student Name signature: Milu Paul

Student SRN number: 21063319

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

# **Abstract**

This paper evaluates the effectiveness of three time-series models—LSTM, ARIMA, and Prophet—applied to sales data from Walmart, with a focus on prediction accuracy and the models' ability to capture trends, seasonal patterns, and irregular behaviors. This paper involves an EDA phase, to visualize and identify the factors affecting the weekly sales. Then after modeling and predicting the data, using Root Mean Squared Error (RMSE) we measure the accuracy. While LSTM and ARIMA were shown to struggle with fluctuating data, it was clear that Prophet performed better than other models in the majority of areas, particularly in capturing seasonal shifts. Therefore, I decide to use Prophet for further prediction with additional regressors. However, it results in accuracy reduction. This also suggests the impact of additional features affecting the models in prediction process.

**Contents**

[1: Introduction 4](#_Toc133253239)

[1.1. Overview 4](#_Toc133253240)

[1.2. Research Question 4](#_Toc133253241)

[1.3. Objectives 4](#_Toc133253241)

[2: Background 5](#_Toc133253243)

[2.1. Dataset 5](#_Toc133253244)

[2.2. Literature Review 5](#_Toc133253245)

[2.3. Algorithms 7](#_Toc133253246)

[2.3.1. .ANN 7](#_Toc133253249)

[2.3.2 LSTM.. 8](#_Toc133253250)

[2.3.3. ARIMA 9](#_Toc133253251)

[2.3.4. Prophet 9](#_Toc133253252)

[3: Methodology 10](#_Toc133253255)

[3.1. Tools and Techniques 10](#_Toc133253256)

[3.2. EDA and Visualization 11](#_Toc133253257)

[3.3. Data Pre-processing 17](#_Toc133253257)

[3.4. Classification Models 17](#_Toc133253257)

[3.4.1. LSTM Model 17](#_Toc133253249)

[3.4.2. ARIMA Model 18](#_Toc133253249)

[3.4.3. Prophet Model 18](#_Toc133253249)

[4: Results and Conclusion 18](#_Toc133253258)

[4.1. Results 18](#_Toc133253259)

[4.1.1. LSTM 19](#_Toc133253249)

[4.1.2. ARIMA 19](#_Toc133253249)

[4.1.3. Prophet Model 19](#_Toc133253249)

[4.2. Prophet with additional regressors 20](#_Toc133253260)

[4.3. Limitations 21](#_Toc133253260)

[4.4. Future work 21](#_Toc133253261)

[4.5. Conclusion 22](#_Toc133253261)

[5. Legal, Ethical and Professional Issues 22](#_Toc133253262)

6.[References 23](#_Toc133253263)

7.[Appendices 25](#_Toc133253264)

# **Introduction**

## **Overview**

Retail is one sector where a sales forecasting is important, which affects decisions about staff, marketing, and inventory. Major firms like Walmart use the ability to predict sales to cost-effectively make their business while also customer-oriented. This wide dataset from Walmart includes weekly sales data along with information on holidays, unusual temperatures, and other local weather-related anomalies across all US retail locations, as well as gasoline costs. Furthermore a number of macroeconomic features, such as the CPI and unemployment rates for good measure.

For forecasting future values from historical data, time series analysis is very important. It is not easy to make good sales plans, even though they are very important anyway. Depending only on the experience to predict sales may not be accurate. However, with the adoption of machine learning (ML) and deep learning approaches for sales prediction has always led to a significant advancement in the field of trade; more so retail.

In this paper, we are comparing three machine learning techniques that is, LSTM, ARIMA and Prophet in forecasting Walmart sales data. Conventional statistical techniques like ARIMA (Pathak, 2020) have proven helpful in tackling trends and seasonality in time series forecasting. They are far more efficient at capturing complicated nonlinear relationships. Facebook developed Prophet, a modern forecasting tool that works well with commercial time series data that show strong seasonal trends, holidays, and outliers—all of which are typical in retail data (Choudhary, 2018). On the other hand, deep learning methods like as LSTM and GRU (Phi, 2018) demonstrate proficiency in identifying complex patterns in sequence-based data and also handling large datasets with various data types – attributes that are critical for retail settings.

## **1.2. Research Questions**

Is there a difference in performance between Deep Learning, Statistical Methods, and Prophet in forecasting sales data in Walmart weekly data?

**1.3 Objectives**

* Create a ML pipeline to implement three types of forecasting automatically on passing the dataset.
* Give a comparison graph of the three methods using a dashboard.
* Train and test the Walmart weekly data.
* Create an automated system that takes input data as csv, pre-process it, train the model, train with additional features and give the results.
* Create an interactive UI for this system.

# **2. Background**

## **2.1. Dataset**

The dataset used for the project is Walmart dataset from Kaggle, a popular open source data platform. It includes eight columns, and a total of 6,435 entries. The store ID, date, weekly sales, holiday flag, temperature, gasoline price (in dollars), Consumer Price Index (CPI), and unemployment rates are among the features included in this dataset. Weekly sales average roughly $1046965, with a wide range of values from roughly $209986 to $3818686. Holiday weeks, which denote specific sale conditions like Thanksgiving, Christmas, etc., are indicated in about 7% of records. Along with fuel prices ranging from $2.472 to $4.468 per gallon sold in all Walmart stores nationwide, the database also includes environmental factors like temperatures between -2.06 and 100.14 degrees Fahrenheit. Additionally, the entire Consumer Price Index (CPI), which measures unemployment percentages and inflation rates for each year's first quarter in the United States, is included.

Dataset Link: https://www.kaggle.com/datasets/yasserh/walmart-dataset/data

## **2.2. Literature Review**

In Latha et al.'s (2023) work, a study on the use of machine learning (ML) models to sales forecasting is presented with a focus on Walmart's XGBoost sales prediction model. The need for efficient ways to use the massive amounts of data produced by modern business contexts is the inspiration behind this work. Sales forecasting is crucial for cost reduction, inventory rationalization, and overall customer happiness. It makes use of the quick and effective XGBoost method, a kind of gradient boosting, and feature engineering to enhance the sales projections in the study.

The applied dataset is derived from Walmart and contains 673,682 records; it refers to the information about the sales in 84 stores in a particular time span. Several processes go into data pre-processing in this research as part of data preparation: cleaning, and integration of data, transforming the data features and reducing the data.

In this paper, they use the XGBoost, Light Gradient Boosting Machine (LightGBM), K-Nearest Neighbours (KNN), and Random Forest (RF), for sales’ prediction. Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Squared Error (MSE) or the R-squared (R2) score are used to evaluate the dependent variable. Among the models, XGBoost is particularly focused in this work as it is considered to be one of the most effective for big and wide datasets. The model meanwhile attains a near index of accuracy of 81%. 34% in predicting sales and is even better than straight regressor and Random Forest. The study also brings in an advancement of GBDT known as the LightGBM algorithm, which also works well in large datasets. The comparison of the suggested models reveals that LightGBM performs better than other models for reasonably high accurate estimates and with better management of huge datasets since XGBoost also maintains a balance in terms of both efficient.

As proved in the empirical analysis, the developed XGBoost is more effective than other ML methods in the sales prediction and this is indicated by the RMSE. For this reason, the model is supported by the assessment phase's final evaluation, which was based on a comparison with actual sales data. Finally, the studies focuses on the relevance and application of the ML models in sales forecasting and their entry in improving inventory management and business approaches. The study also shows that even more complex models like XGBoost or LightGBM also offer a significant advantage over the models already in use due to which businesses can have improved decision-making regarding sales. The paper emphasises the importance of the preparation of the data and feature selection in the development of accurate forecasting models and make the way for future work in the field of sales forecasting.

The study (Neba, 2024) presented study of time series analysis, focusing on a case study of Walmart sales data. In order to study the seasonal trends, patterns and variations time series is essential. The main goal of this paper is comparing performance of different models, including Prophet, ARIMA, SARIMA, Exponential Smoothing and Gaussian Processes. This paper also include autocorrelation analysis, smoothing techniques to identify temporal dependencies in the data.

The dataset used in this paper is Walmart sales data from Kaggle, from the period of 2010 to 2012. The main purpose of this study is to assess the effectiveness in capturing the inherent patterns and variability present in Walmart sales by evaluating the performance of our time series models against historical data. This paper also discuss about the real-world uses of time series methods and how they affect the decision making in prediction process. To improve model robustness they use winsorization technique to handle outliers. For measuring model performance they use RMSE and MAE metrics.

After modelling and prediction of the models, Gaussian process model perform better than other models. ARIMA, SARIMA and exponential smoothing show moderate performance. Prophet show slightly worse performance than the others. This paper summarize by suggesting that even though new techniques improved model predictions, the old techniques are still good at interpretability.

LightGBM algorithm is the subject of the study (Qiao, 2020) to improve the prediction of sales for retail data from Walmart through creating a sales forecasting model. A good sales forecast helps a company to predict the sales thus helping in planning and budgeting. The paper is a reminder that only through such higher order techniques of machine learning can one recover meaningful patterns from appropriate sales data, and thereby help formulate better strategies for organisations.

The data used in this study is based on sale data for Walmart stores in three US states. The data includes different features like products, store information, and temporal fields. They use one hot encoding to convert the categorical information into numerical information and to filter out irrelevant data for model training, maximum possible prices and standard deviations are calculated. This study uses three different models that is Linear Regression, SVM and LightGBM. Linear regression assumes the data to be fitted using a linear predictive equation. SVM that has originated from the pattern recognition models is also checked for forecasting efficiency. But from the result of the evaluation of our models, the LightGBM model has a higher predicting capability compared to Linear Regression and SVM. These models are assessed by using the root mean square error RMSE as the performance measure. The LightGBM model is again the lowest at 2 for the RMSE value . 09, compared to 3. 35 for Linear Regression and 2.88 for SVM.

LightGBM is an open-source framework that is built on top of GBDT and optimised for its performance. Among its values, it can support parallel training and is especially useful in cases when the number of features is high. LightGBM’s fundamental concept is the training of weak classifiers to develop a robust model in the successive iterations. It works on the level-wise growth, in which the leaf with the highest split gain will be chosen for further splitting, resulting in higher accuracy. Several tricks are implemented in the LightGBM model to optimize the results as follows; a faster computation time and more efficient cache utilisation result from the use of histogram-based decision trees and the Gradient-based One-Side Sampling (GOSS) technique is used to cut the computational costs .This approach guarantees that the model is only working on the most relevant data set and in so doing, speeds up the model without reducing the model’s quality. Secondly, there is the Exclusive Feature Bundling (EFB) technique that eliminates excessive features and joins those that are dependencies to improve computational speed and overcome computational issues.

The findings thus suggest that not only does the LightGBM model afford higher accuracy of the forecast, but it also affords information about feature importance. Sales are influenced by price fluctuations and a number of time-related records, such as those that deal with a weekend or holiday. This study demonstrates the need for extreme caution when choosing which features to include in a model and when pre-processing the required data in order to create a successful sales forecasting model. It also suggests the degree to which feature engineering and data preprocessing are important, for any sales forecast models in this field of employment.

## **2.3. Algorithms**

### **2.3.1. ANN (Artificial Neural Networks)**

ANNs are computational systems meant to emulate the human brain. They are neural circuits formed from neurons that analogously take inputs, perform computations, and become enlightened from the inputs. Each neuron is going to take the input and pass it through a mathematical function and then move it to the next layer. The links between neurons have a value known as weight and these are changed during training in order to improve the model. A basic form of an artificial neuron known as perceptron makes a decision on the weighted inputs that have gone through an activation function and make the network learn non-linear so that it can interpret various patterns. Some of the activation functions include sigmoid, ReLU, and tanh. Perceptrons connected in formation are termed as multi-layer perceptron abbreviated as MLP; the layers present between the input layer and the output layer is termed hidden layers. More hidden layers in the network enhance the capacity of the network to learn complex relationships but at the same time; that also increases the probability of overfitting the data (Singh, 2021).

ANN training can be done by modifying the weights in order to arrive to zero in the error function that consists of the difference between the output and the target values. This is made possible by back propagation whereby the mistake is taken through layers and a correction made to rectify the wrong calculation. There are often used for updating weight vectors gradient descent algorithms, with the help of the first derivative of the loss function, in which there is information about the error in the model’s predictions. The loss function depends on the type of the problem with mean squared errors used for regression problems and cross-entropy for the classification problems respectively.

ANNs have several hyper parameters, which determine the learning process, and include the learning rate, which controls the size of the steps in the gradient descent algorithm. A high learning rate may lead to a faster and quicker convergence of the model on the cost function but may also lead to convergence that exists on the surface which is an unstable state whereas a lower rate of learning may take a very long time, converge very slowly or sometimes fail to converge and may end up in a local minimum. Other factors that we need to consider include how many layers are there in the model, how many neurons does each of the layers and the fact that as the number of layers and neurons increase the model gains the ability to learn the features it needs, but at the same time there are higher chances of overfitting (Anand, 2023). To reduce overfitting it include regularization, dropout and early stopping. Here are some regularization techniques such as L1 and L2 which targets large weights to avoid over-fitting the model to the training data. Dropout randomly drops some neurons out during training in order to reduce over fitting, while early stopping stops training when the cross-entropy loss of the validation set ceases to decrease.

Weight initialization is also essential in training of ANNs since a poor initialization could slow down convergence or results in getting stuck in local optima. There are standard procedures that include Xavier or He initialization, to feed the initial weights in light to the number of neurons in the layers. There are additional methods known as optimizers which work hand in hand with gradient descent, more common of which are Adam, RMSprop, and SGD in which Adam is for now the most preferred one as it offers learning rates for weight adjustment for the different weights in question (Abdolrasol *et al.*, 2021).

### **2.3.2. LSTM (Long Short-Term Memory)**

RNNs – LSTM – are a kind of networks that are developed for input data sequences: temporal, sequential, or long (Mittal, 2019). While addressing the vanishing gradient troubles in normal RNNs, LSTMs contain a memory cell which can contain information for a long duration. The memory cell is managed by three gates: the input gate, the forget gate, the output gate: these three entities control the information input, output, and within an individual memory cell. Since LSTMs can handle long-range dependencies in data, their architecture makes them useful for time series analysis, language modeling, and speech recognition.

LSTM networks also have several hyper parameters, but primary they include the number of layers, the number of memory cells (units), and the sequence length. Raising these values enables the network to detect higher level patterns, at the same time, it boosts the risk of over fitting and the computation load. The learning rate, batch size and the optimizer also affect the working of LSTM networks. Dropout is used in LSTMs to make the network more regularized and it basically helps to remove certain units during training and this helps prevent over-fitting. Weight initialization is also important in order and poor weight initialization impacts the learning of the network (*Understanding LSTM networks*, no date).

### **2.3.3. ARIMA (Autoregressive Integrated Moving Average)**

ARIMA are other models which are common in the analysis and forecasting of time series. ARIMA integrates the aspects of autoregressive integrated moving average models which embraces the use of previous values to make future values and moving average models which predict future values by means of past forecast errors. The ARIMA model also has a differencing component to change non-stationary time series to stationary time series and thus can handle a wider kind of time series data. The ARIMA process has three orders, which are the autoregressive order, the differencing order and the moving average order, values of which are selected according to the nature of the data and the forecast objectives (Fuqua School of Business, no date).

However, there are some drawbacks associated with the use of ARIMA technique, most especially where the pattern of data has non-linear relationship or in cases where multiple time series are involved. To tackle these issues, some new extensions of the ARIMA model have been introduced, and they are SARIMA, which adds the seasonal differencing; ARIMAX which includes exogenous variables; and VARIMA, which can handle multiple time series (Artley, 2022). The ARIMA models form a flexible approach for modelling and forecasting of time series data by including the features of autoregressive moving average model together with differencing strategy to tackle non-stationary. That is why the proposed model is widely used throughout different fields: it is flexible, and, what is more important it is efficient.

### **2.3.4. Prophet**

Prophet, which is created by Facebook, a highly flexible and flexible model designed to handle time series with seasonality and data loss problems. Prophet was intended to be used with data that included daily cycles, weekly or yearly cycles, the effects of holidays, and non-linear trends of any kind. From its design, the model is easy to use with greater stability to typical time series problems and is flexible thus suitable for most business operations forecasting needs such as sales, demand and other performance markers (Dieckmann, 2024).

One of Prophet's core strengths lies in its ability to decompose time series data into three key components: trend, seasonality and festive times. The trend factor is used to measure the increase or the decrease of the data set in the long run, whereas the seasonal variation factor computes rhythms, within cycles that may be month-end, weekly or annual in nature. Holiday component enables the model to consider the impacts of holidays and other repeating phenomena which have major influence on the time series. This decomposition is beneficial for Prophet specifically in cases where there is a definite seasonality in the data, which is rather usual in the retail, financial, and many more trading and business activities.

Simplicity and strong tolerance to outliers are other major strengths of Prophet. Still, in comparison with many traditional time series models, Prophet is not very sensitive to ‘outliers’, meaning that even if there is an analytically unmanageable data point, the accuracy of the forecast will not be dramatically affected. This feature is especially important in practical usage because, due to promotions, changes in the market, etc., the actual data can be quite noisy, and the predictive model needs to take this into account.

The case, with existing arrangement to cover missing piece of data, makes the model even more practical. As we will discuss more in subsequent chapters, many real data sets are such that the actual observations for one or more time periods are unavailable due to holidays, weekends or otherwise irregular observation schedules. Prophet thus smoothly fills these gaps or interpolates them in a sense to enable uninterrupted forecasting without having to prepare data heavily (Melanie, 2024).

The simplicity of the implementation of Prophet, which is the primary reason for choosing it among other models, and the ability to address most of the issues related to time series analysis, has placed Prophet among the favorites for data scientists and analysts.

**3. Methodology**

## **3.1. Tools and Techniques**

We have done the project on the Google colaboratory in python language. We have used essential libraries and tools for processing and modelling the data that is mentioned below:

* Pandas: A popular data manipulation library for loading, cleaning and preparing the data for analysis in the form of a csv file which is Walmart.csv, and for handling time series data.
* Matplotlib: A plotting library, which is helpful in developing some of the graphics like the histogram and line plots for the analysis of the distribution aspect and the trends of data.
* Seaborn: A statistical data visualization library that comes on top of Matplotlib and is used to create beautiful plots such as histograms, scatter plots, and the box plots among others.
* StandardScaler (from sklearn): A tool that prepares data in a manner similar to those of models like the logistic regression machine (LSTM) will, for example, remove the mean and scale the results to unit variance.
* ARIMA (from statsmodels): A regression equation that is employed in the modelling of a time series so as to obtain optimal forecasts that incorporate elements of the autoregressive moving average nature.
* Prophet (from Facebook's Prophet library): A forecasting technique used where data is analysed in segments of time, and further divided into, trends, cycles and holiday effects for accurate prediction of future values especially in situations where some data may not be collected or is in intervals that may not be regular.
* Keras (with Tensor Flow backend):
  + LSTM (Long Short-Term Memory): One of the recursive neural networks that can be used in time series prediction and which has the ability of capturing long-term temporal dependencies in a series.
  + Sequential model: It is just another linear stack of layers in Keras which are used to build and train the LSTM network.
  + Dense layer: A dense layer in the form of a neural network used to prepare the final output from the LSTM model.
* RMSE (Root Mean Square Error): A measure used to determine the quality of the forecast produced by LSTM, ARIMA, and Prophet Models and how well these are to the actual values.
* Train-Test Split: A method to split a dataset into training and a testing set which can be used for the testing of the general performance of the model.
* Time Series Resampling: Gathering of data within various time integration (e. g. weekly) in order to allow time series analysis and modelling.
* Box Plot: A technique that is applied in an attempt to identify the presence of outliers and in visualizing the nature of the data in correspondence to the features together with Weekly Sales, Temperature, Fuel Price, CPI and level of unemployment.
* Hyper parameter Tuning: Fine tuning of LSTM and other model by changing the values of parameters such as number of epochs, batch size, learning rate and similar values.
* Outlier Detection (using IQR method): A method of assessment to define and employ tools to study the skewness of the data where necessary via the IQR method.
* Feature Engineering (in Prophet): Using more variables in the Prophet model, (for example, Temperature, Fuel Price, CPI, Unemployment) helps to enhance the accuracy of the Prophet model.

## **3.2. EDA and Visualizations**

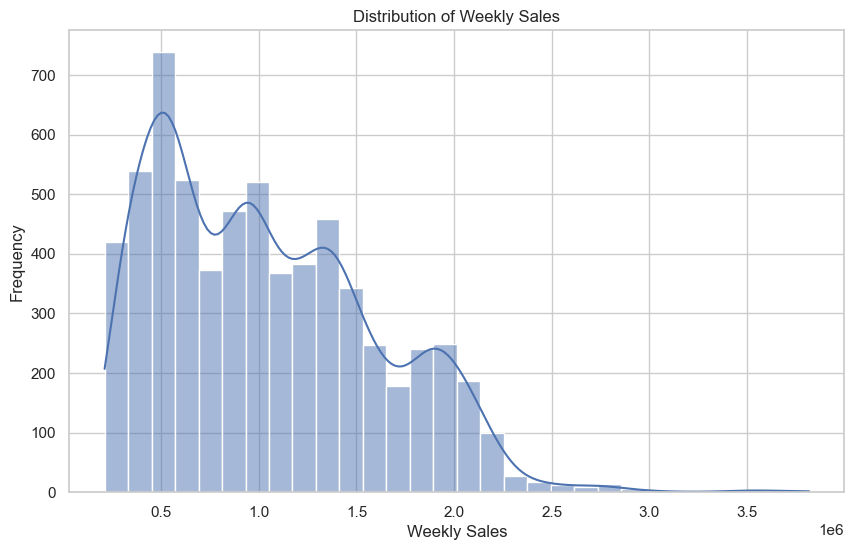


Fig 1: Plot showing distribution of weekly sales

The first plot is a histogram and a Kernel Density Estimate plot which show the weekly sales distribution in the dataset. The x-axis denotes the number of weekly sales values, while the y - axis denotes how often those values occur. The numbers are representing in 1 million notation. The vertical axis of the histogram represents the numbers of ranges of weekly sales that fall into each category of the horizontal axis. The curve drawn above the bars which is called KDE curve is an estimate of the probability density function of the data. It makes the data distribution to be continuous for better understandings of the actual trend of the data. In this plot, it looks like the weekly sales have positive skewness, which means that a frequency of sales is dense within a range of low sales and highly stretched towards the high sales. This distribution also shows that more often there are many weak sales figures in the week and there may be extremely high sales figures in the week as well. From the graph it is evident that most of the stores have moderate sales and sales more than 3.5 million occurred less than 100 times. Thus the weekly sales distribution is very important for making predictions from the data.

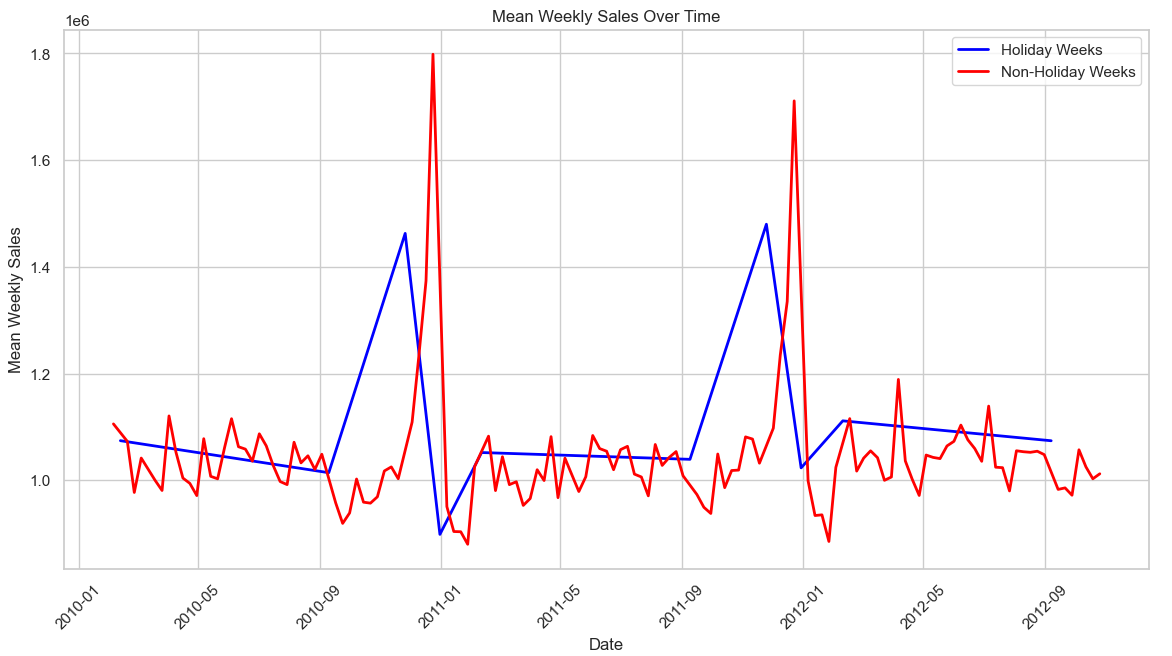


Fig 2: Weekly sales on holidays and non-holidays

This is a line graph plotting the weekly sales during holiday and non-holiday over time. The x-axis showing the time period and the y-axis showing the weekly sales in 10 million notations. The red line represent the weekly sales during non- holidays and the blue line represent the weekly sales during holidays. From this graph it is clear that blue line having largest peaks indicates the sales during Christmas, Thanksgiving, and Black Friday. Apart from busy weeks, especially from the month end to New Year season, non-holiday weeks are showing a regular level of sales without major rise or fall. So on comparison sales during holiday weeks are higher than sales during non-holiday weeks.

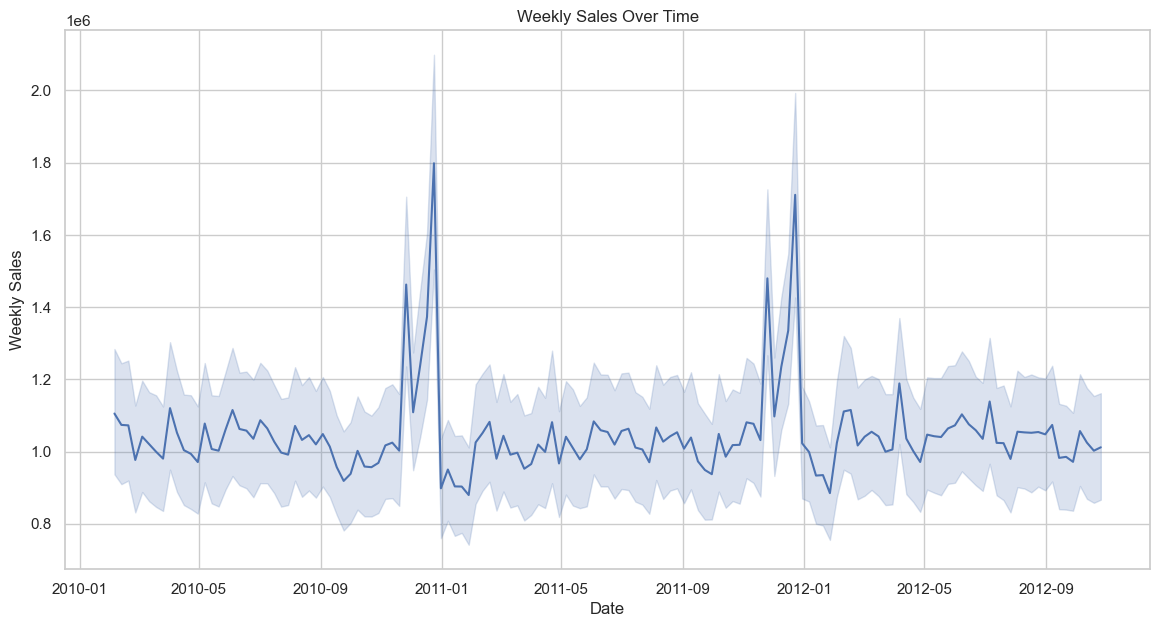


Fig 3: Line graph showing weekly sales over time

The second type of plot is a time series plot of the sales by week, this is a way of demonstrating the progression of the sales. The x axis is the date while the y axis has the weekly sales of the products. The following plot shows a pattern of sales for the period 2010 to 2012. The actual weekly sales are the blue line through the middle whereas the shaded area that encompasses the line is the confidence interval or margin of error on the weekly sales data. By observing the time series plot, one can easily identify certain point of times where there are large fluctuations in the sales, mostly at certain time intervals which could be related to festive seasons or certain discounted periods shown by large spikes. It also reveals phases of moderate sales and phases of decline in the sales. This time series plot best illustrates the trends and seasonality of the sales.

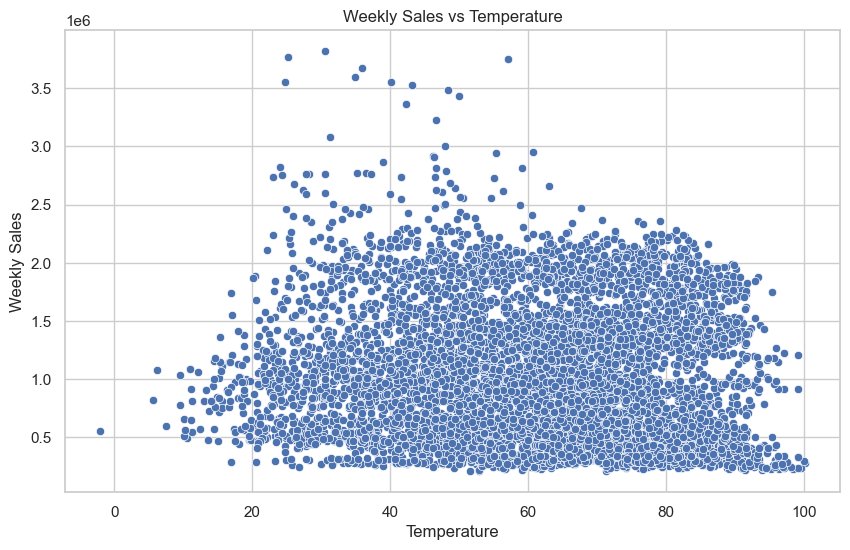


Fig 4: Scatter plot showing weekly sales with temperature

The third plot created is the scatter chart which represents weekly sales against temperature. On the x-axis we have the temperature and, on the y-axis, highlighting the figure of weekly sales. They are used to represent each observation in the dataset, and their position always depends on temperature and the corresponding weekly sales. In scatter plot it is possible to look for the relationships or trends between these two variables. The plot illustrates this idea, with the points widely dispersed and deviating from a straight line implies that there is no correlation between the weekly sales and temperature. However, relative density is highly focused on points of low values of sales, therefore besides temperature, other factors could have a stronger impact on these parameters.

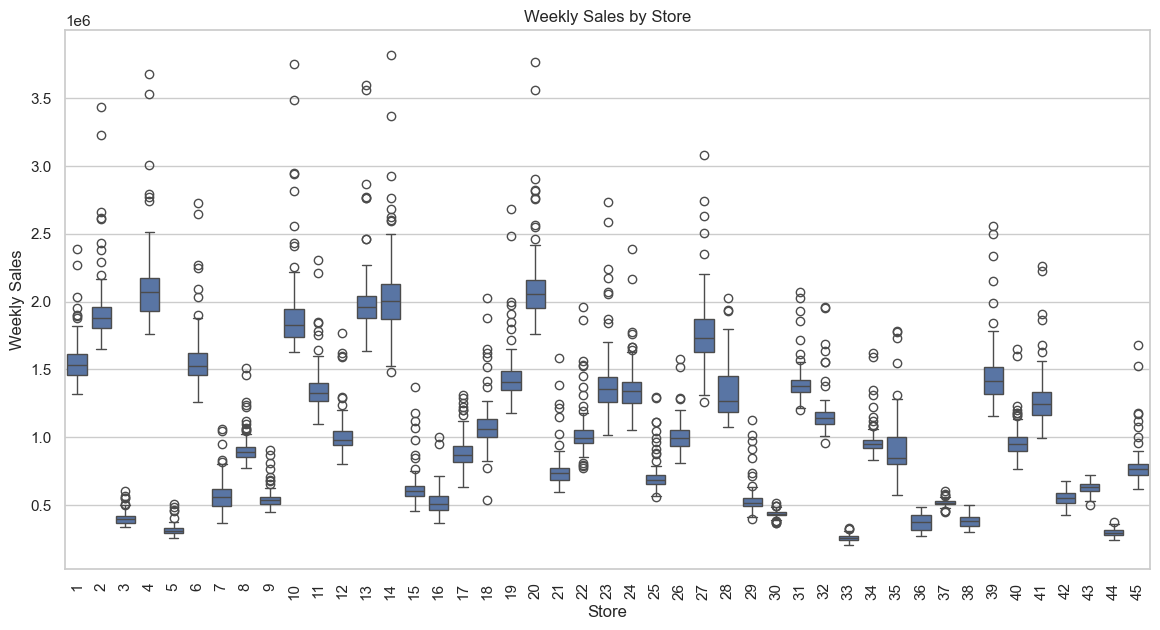
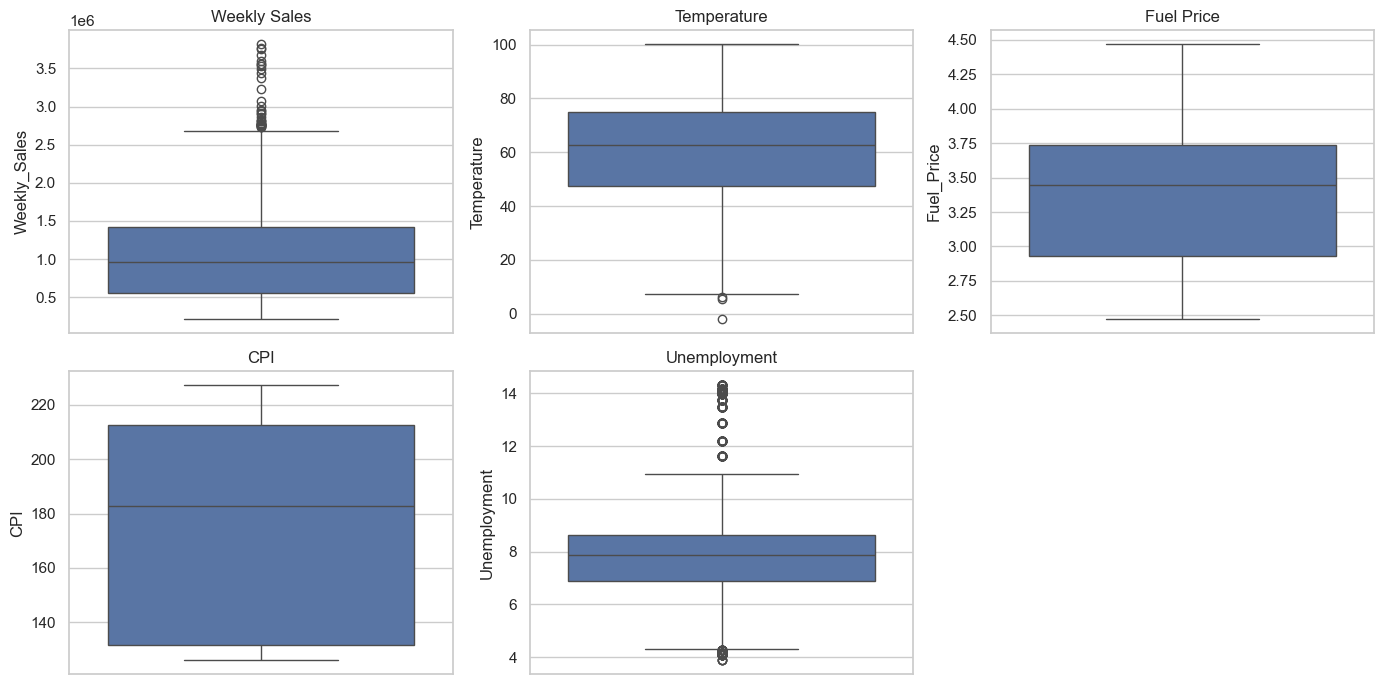


Fig 5: Box plot showing weekly sales by store

The fourth representation is the box plot that represents weekly sales and stores. The x-axis is the stores whereas the y-axis shows the weekly sales rate. The box plot representing the sales of each store where the box in the middle represents the interquartile range and the horizontal line in the box represents the median while the perpendicular lines through the box are the whiskers are going up to 1. 5 times the IQR. In all of the plots, any data points beyond the whiskers are considered outliers and are plotted separately. Hence, the box plot has helped to compare the stores in terms of median sales performance, dispersion of the performance and outliers. For example, some of the stores like 4 and 14 having the data that is much spread out with extreme values have high sale compared to other stores and store number 7 and 21 is showing low sales while others present more moderate data with few extreme values. This plot is useful in determining those stores that have high or low sales performances and may also suggest the level of dispersion of sales within a store and across them.

  
Fig 6: Box plots representing distribution of variables

The fifth visualization consists of multiple box plots, each representing the distribution of different variables: Number of units sold per week, temperature, fuel price, CPI and unemployment rate on our products. The box in each box plot in turn presents the extent of variability in the data of these variables, so that their distribution can be easily compared. Looking at the box plot of weekly sales the graph show’s that it is positively skewed and most of the observation are much more larger than the others hence some weeks experience high sales whereas most weeks showing moderate sales. Looking at the temperature box plot, one is able to see that the plot range of the temperature data is somewhat small but the temperature box has long tails suggesting that although the temperature range as a central tendency, there does exist extreme temperatures at either end of the scale. According to the comparison of the two box plots, it can be identified that there is relatively higher dispersion of fuel prices during the period under the consideration of the dataset. From the CPI box plot, the distribution seems more or less steady with a broad spread and is an illustration of the macro-economic trends in a given period. The box plot of the unemployment rate is significantly wider can be attributable to some outliers thus showing that while most countries have kept moderate unemployment rates, there have been higher rates of unemployment from time to time.

All the above graphs, help in achieving the goal of explaining the structure of dataset and the relations or dependencies between variables. These visualizations plays a crucial role in visual pattern analysis, problem finding, choosing the best actions for developing a predictive or diagnostic model, and even developing business strategies based on the data.

**3.3. Data Pre-processing**

* Datetime Conversion and Sorting: To ensure that all date related operations that can be applicable on the dates present in the ‘Date’, the ‘Date’ column of the given dataset is converted to the ‘datetime’ format.
* Index Setting: The ‘Date’ column is set as the index of the Data Frame this allows for resampling and time series analysis.
* Resampling: This is done through re-sampling the weekly total sale in a week so as to aggregate the daily data in a week. This step aids in maintaining the appropriate interval for the data points this is vital in forecasting the data points.
* Train-Test Split: Partition of data the dataset is divided into the training set and the test set with a training size of 0.8. So that 80% of data is used for training and 20% of data is used for testing the model. As the dataset is smaller, a larger train size is more effective. This split allows a degree of verification on new data, which is very crucial since it tests the model’s true capability of effectively working on new data, data which was not used in the training of the model.

**3.4. Classification Models**

**3.4.1. LSTM Model**

* Data Scaling: In order to make the values on the sales data common, it is scaled using StandardScaler. This step is required as scaler object is used for normalizing and scaling the input data on which LSTM models are sensitive.
* Data Preparation for LSTM: This is done by feeding the data to create sequences of a particular time step and each sequence is utilized to forecast the subsequent value in the sequence. They reshape the input data into the required format of LSTM which is a three-dimensional matrix.
* Model Definition and Training: Sequential LSTM model is created with two LSTM components which is followed by dense component. The model uses ‘Adam’ optimizer to adjust weights with a batch size of 1 and using 5 epochs. I have used ‘sgd ‘initially but it gives a low accuracy and slow compared to Adam. Adam is used frequently because it involves the use of a learning rate which is perfect for training of LSTMs due to the fact that it applies the learning rate for each individual parameter. The model is trained is by using the training data and adjusting for the loss function (mean squared error) to minimize on the prediction errors which is used for regression.
* Prediction and Evaluation: The trained model in the case predicts the outcomes on the training and the testing data. The predict values are inverted to the initial formats, based on which RMSE is computed in order to assess the model.

**3.4.2. ARIMA Model**

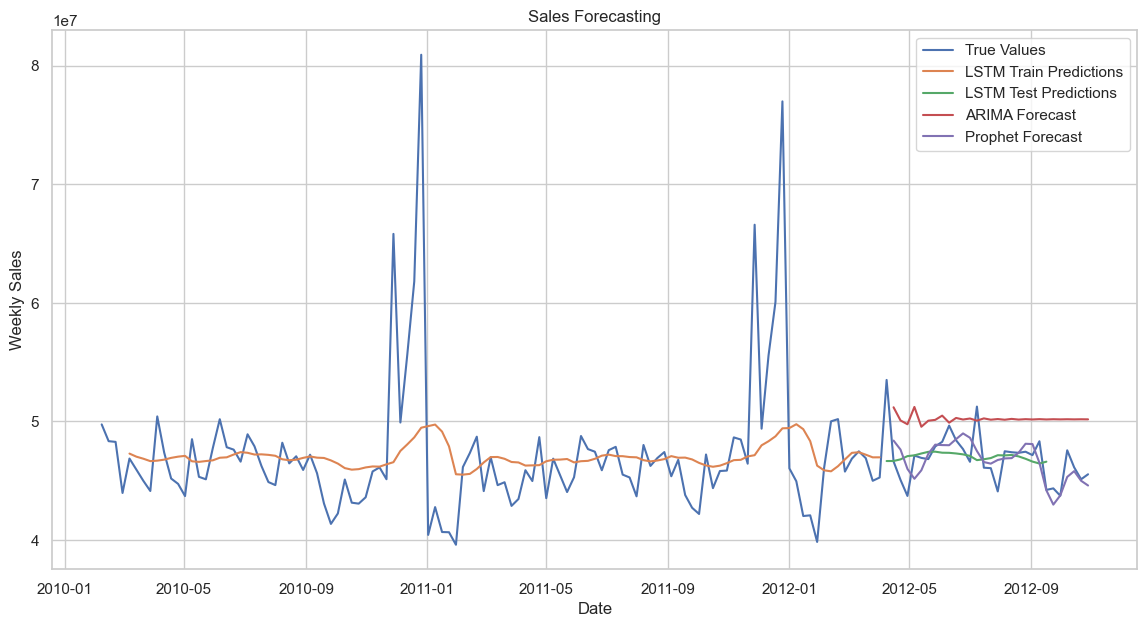
* Model Fitting: The ARIMA model is fit on the training data with three parameters (5, 1, 0) where the first figure represents the order of autoregressive term, second the order of differencing term, which is the measure of varying observations to make the time series stable. The last one is the order of moving average term.
* Forecasting: During forecasting, the model generates number of future steps to forecast as the same length of the test data.
* Evaluation: The model predicts the values into the future and RMSE is then computed from the predicted values with the test data.

**3.4.3. Prophet Model**

* Data Preparation: Initially we reset the index of weekly data and also the Prophet wants to have specific column names thus we change date as ’ds’ and weekly sales as ‘y’. Then we fit the model to the training data.
* Model Training and Forecasting: Whenever future values are required; they are predicted using the Prophet model which has been estimated using the historical data in conjunction with the additional regressors. The forecasted values are matched to the actual values using RMSE and then the performance of the model is examined.

# **4. Results and Conclusion**

## **4.1. Results**

  
Fig 6: Sales forecasting graph by various models

### **4.1.1. LSTM**

From the graph we can identify that the Long Short-Term Memory (LSTM) was evaluated based on the weekly sales data of the products. Here, the LSTM based model resulted in training RMSE nearly about 47026541 and testing RMSE of nearly about 47016045. This results mean that the model rarely over fits and has a good balance of training. However, the high RMSE values will show that the model is very inaccurate compared to the actual values.

From the sales forecasting graph regarding the LSTM model the fit and predictability shows a little similarity with the true values as the green shadow line on the graph denote the training set and the red dotted line the testing set hence, we can clearly estimate there is some deviation at certain period, particularly when the true sales values are either high or low. The first particularity is that the model tends to fail in reflecting the abrupt increases in sales that periodically occur, hence the large absolute errors during these periods. This was demonstrated by the comparatively high RMSE values, which show that it is difficult to forecast really large fluctuations in the data.

### **4.1.2. ARIMA**

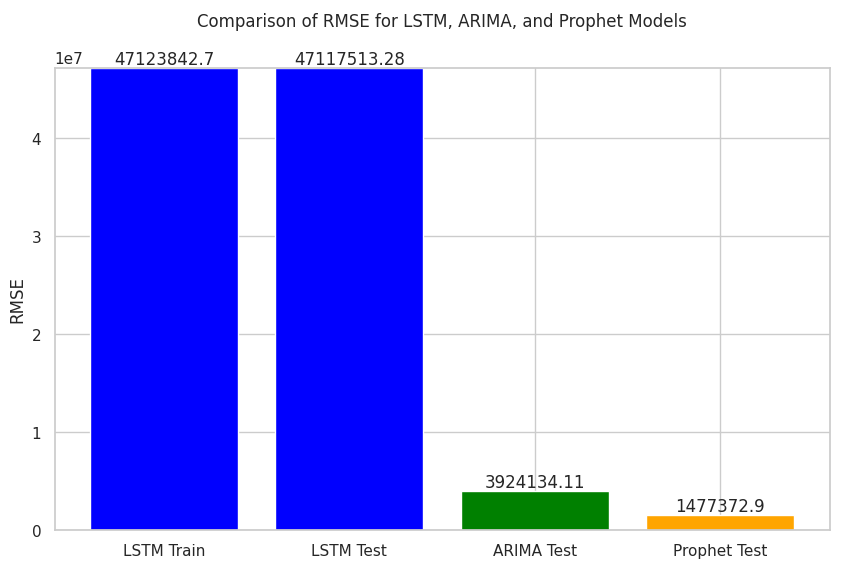
For comparison, the same data was processed using the ARIMA (Autoregressive Integrated Moving Average) model aimed at detection of changes of patterns of the sales data. The average RMSE value of the ARIMA model is about 3,924,134. This lower RMSE means that the ARIMA model had higher level of accuracy in terms of prediction as compared to the LSTM model.

The forecasted line from the ARIMA model shown in the sales forecasting graph try to fit the general patterns of the sales data but in a much smoother manner. As a result, the model can accurately represent the overall flow of the sales data, but it results in the loss of some smaller features, particularly the spiky ones. This suggests that ARIMA, which is less volatile and actually reduces variations to some amount, may be more suited if the trend and seasonality components were more steady.

### **4.1.3. Prophet**

The Prophet model also tested on the sales data, with no extra regressors in the model, the initial RMSE is around 1,477,380 and is lower than the LSTM and ARIMA forecasting models. This result shows that Prophet Model does well in identifying patterns of the sales data as it is designed to address trend, seasonality, and holidays.

ARIMA model is slightly inferior to the Prophet model in terms of the depiction of the forecast line, as the latter even captures some of the sales bumps far better. This is clearly shown by the low RMSE number, which indicates that the model type is able to very accurately predict future sales values based on the given historical data.

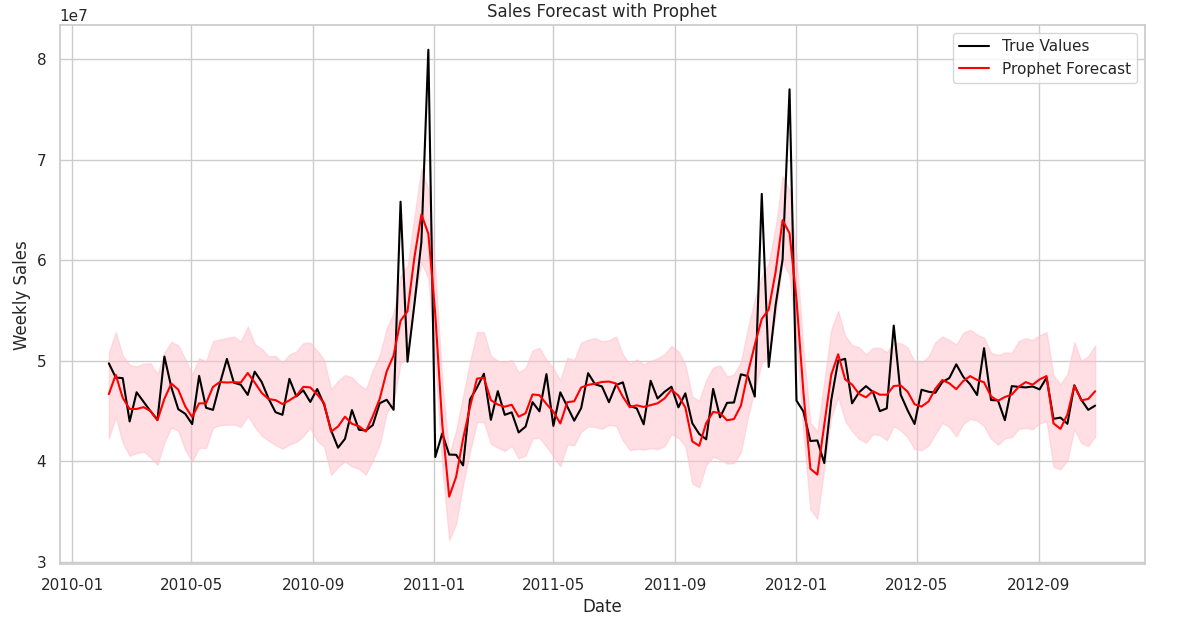
  
Fig 6.Comparision of LSTM, ARIMA, Prophet using bar chart

The bar chart of the models of all the RMSE helps to understand the extent of the difference in the approach. The trained LSTM model has a certain degree of efficiency between training and testing, with the highest RMSE values, however, suggesting relatively low accuracy in terms of forecasts. Still, in comparison to LSTM, ARIMA is more accurate in its predictions but still outperforms by Prophet. Prophet is noticeably more effective than either LSTM or ARIMA when used alone, and this is especially true when it is trained with only the most basic features.

**4.2. Prophet with additional regressors**

When comparing the RMSE scores of LSTM, ARIMA and Prophet, the Prophet performed best for this specific sales forecast task. But as we can observe, LSTM never degrades with training and testing data, while a high error rate justifies that maybe LSTM need tuning up to this dataset or could be, this dataset needs a different topology. For smoother, less volatile data, ARIMA would have been a decent middle ground, but Prophet can handle rightly-complex seasonality and trends much better than the rest of the models. Adding more variables is more beneficial, so we add more regressors that is temperature, fuel price, CPI, holidays and unemployment to the Prophet model.

After pre-processing and modelling, I have removed the entire row with missing values. This is done to reduce inaccurate predictions. Then we calculate the RMSE value of Prophet with additional variables. While the addition of other variables leads to higher RMSE value means accuracy is reduced. Thus the experience of variation with adding regressors into Prophet shows the fact that the introduction of new variables can sometimes has a negative impact on the model and deviate the results.

  
Fig 7: Forecasting with additional regressors using Prophet

The above graph is obtained by plotting sales forecast with prophet against the actual data. The x-axis showing the date from 2010 to 2012 and the y-axis showing the sales figures in the range of 10 million. The black line represents the actual values of the sales and the red line is the Prophet Forecast line. The shaded area is the confidence interval where the uncertainty is high is the widely shaded portion. From the graph what I have observed is that the Prophet model tries to follow the peaks and some fluctuations but it deviates from actual data at some points. This suggests that the model shows good results even though it needs more improvement.

**4.3. Limitations**

One of the challenges while doing the project is that, the dataset contain seasonality trends and holiday effect. The sales during holidays may not be the same, it may vary according to the location and store. This is evident from the inability of Prophet model to follow the actual sales especially the peaks and fluctuations even though it tries to captures the tends and seasonality. Another major challenge is that when adding additional regressors it led to reduced accuracy. As the Prophet model shows good accuracy initially but after adding more features with them the RMSE score increased. Finally tuning the LSTM model was a bit challenging as the improper tuning was the major factor that led to high RMSE value.

## **4.4. Future Work**

In the future work, there could be an improvement of the various model performances through the application of better hyper parameter tuning approaches such as Grid Search as well as Bayesian Optimization for the LSTM and Prophet models. However, possible further research shows that combining different functioning ARIMA, LSTM, and Prophet models can further increase accuracy. Future research could also examine the effects, for instance, when more random variables have been included and how different feature selection techniques can help reduce over fitting. The investigation would next be extended to additional retail datasets and cross-validations in order to confirm the overall effectiveness of each model.

**4.5. Conclusion**

In conclusion, this project offers comparison of performance between different forecasting models using the Walmart dataset for sales prediction. The forecasting models used here are LSTM, ARIMA and Prophet. Among them traditional models like ARIMA are significant but in this paper Prophet shows highest accuracy with least RMSE value implies that modern tools have the capacity to handle complexities of retail sales data .Despite the challenges faced, future work and further hyper parameter tuning can help LSTM to improve the forecast accuracy.

# **5. Legal, Ethical and Professional Issues**

Accurate sales forecasting is necessary to avoid providing false information that influences business decisions that negatively impact all stakeholders and market competition. This is one of the legal, ethical, and professional challenges. Ignorance of the models can occasionally result in inaccurate projections, which can be expensive when it comes to inventory-related problems or possibly cause unjust pricing. From a legal point of view, it might be necessary to inform decision-makers to the shortcomings of the forecasting models. Furthermore, these models must be employed carefully and without accounting for market fluctuations and other circumstances that might cause them ineffective or hazardous to the company.

# **6. References**

Abdolrasol, M.G., Hussain, S.S., Ustun, T.S., Sarker, M.R., Hannan, M.A., Mohamed, R., Ali, J.A., Mekhilef, S. and Milad, A., 2021. Artificial neural networks based optimization techniques: A review. *Electronics*, *10*(21), p.2689.

Anand, A. (2023) *Hyperparameters of an artificial neural network*, *Linkedin.com*. Available at: <https://www.linkedin.com/pulse/hyperparameters-artificial-neural-network-aditya-anand> (Accessed: August, 2024).

Artley, B. (2022) *Time series forecasting with ARIMA , SARIMA and SARIMAX*, *Towards Data Science*. Available at: [https://towardsdatascience.com/time-series-forecasting-with-arima-sarima-and-sarimax-ee61099e78f6](https://towardsdatascience.com/time-series-forecasting-with-arima-sarima-and-sarimax-ee61099e78f6%20) (Accessed: August, 2024).

Carlsson, F., 2024. Regression models for heat pump performance: Exploring statistical and machine learning techniques for estimating heat pump COP and Carnot efficiency.

Choudhary, A. (2018) *Generate Quick and Accurate Time Series Forecasts using Facebook’s Prophet (with Python & R codes)*, *Analytics Vidhya*. Available at: [https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/](https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/%20) (Accessed: June 14, 2024).

Dieckmann, J. (2024) *Getting started predicting time series data with Facebook prophet*, *Towards Data Science*. Available at: [https://towardsdatascience.com/getting-started-predicting-time-series-data-with-facebook-prophet-c74ad3040525](https://towardsdatascience.com/getting-started-predicting-time-series-data-with-facebook-prophet-c74ad3040525%20) (Accessed: August, 2024).

Fuqua School of Business (no date) *Introduction to ARIMA models*, *Duke.edu*. Available at: <https://people.duke.edu/~rnau/411arim.htm> (Accessed: August, 2024).

Latha, S.B., Dastagiraiah, C., Kiran, A., Asif, S., Elangovan, D. and Reddy, P.C.S., 2023, August. An Adaptive Machine Learning model for Walmart sales prediction. In *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)* (pp. 988-992). IEEE.

Melanie (2024) *Facebook Prophet : All you need to know*, *Data Science Courses | DataScientest*. DataScientest. Available at: https://datascientest.com/en/facebook-prophet-all-you-need-to-know (Accessed: August, 2024).

Mittal, A. (2019) *Understanding RNN and LSTM - Aditi Mittal*, *Medium*. Available at: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e (Accessed: August, 2024).

Neba, C., Nsuh, G., Neba, A., Webnda, F., Ikpe, V., Orelaja, A. and Sylla, N.A., 2024. A Comprehensive Study of Walmart Sales Predictions Using Time Series Analysis. *Asian Research Journal of Mathematics*, *20*(7), pp.9-30.

Niu, Y., 2020, October. Walmart sales forecasting using xgboost algorithm and feature engineering. In *2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)* (pp. 458-461). IEEE.

Pathak, P. (2020) *Building an ARIMA model for time series forecasting in python*, *Analytics Vidhya*. Available at: [https://www.analyticsvidhya.com/blog/2020/10/how-to-create-an-arima-model-for-time-series-forecasting-in-python/ (Accessed: June 14, 2024).](https://www.analyticsvidhya.com/blog/2020/10/how-to-create-an-arima-model-for-time-series-forecasting-in-python/%20(Accessed:%20June%2014,%202024).)

Phi, M. (2018) *Illustrated Guide to LSTM’s and GRU’s: A step by step explanation*, *Towards Data Science*. Available at: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> (Accessed: June 14, 2024).

Qiao, Z., 2020, October. Walmart Sale Forecasting Model Based On LightGBM. In *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)* (pp. 76-79). IEEE.

Taylor, S.J. and Letham, B., 2018. Forecasting at scale. *The American Statistician*, *72*(1), pp.37-45.

*Understanding LSTM networks* (no date) *Github.io*. Available at: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (Accessed: August, 2024).

Vyas, R. and As, R., 2022, March. Seasonal Sales Prediction and Visualization for Walmart Retail Chain Using Time Series and Regression Analysis: A Comparative Study. In *2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN)* (pp. 1-6). IEEE.

# 

# **7. Appendices**

Below given the colab link:

<https://colab.research.google.com/drive/1VWLcFpyareYP5WN5dvTX9J3Am8K6K4Zu?usp=sharing>

Below given the code of the project:

*#import essential libraries*

import pandas as pd

import matplotlib.pyplot as plt

*#Loading the data*

from google.colab import drive

drive.mount('/content/drive')

df = pd.read\_csv('/content/drive/MyDrive/Walmart.csv')

*#Display first 5 rows*

df.head()

*#Plotting histogram of weekly sales*

df['Weekly\_Sales'].plot(kind='hist');

*# Convert the 'Date' column to datetime format*

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

*# Check for missing values and duplicates*

missing\_values = df.isnull().sum()

duplicates = df.duplicated().sum()

print("Missing values per column:")

print(missing\_values)

print("\nNumber of duplicate rows:")

print(duplicates)

import seaborn as sns

*# Set the style of the visualizations*

sns.set\_theme(style="whitegrid")

*# Sales Distribution*

plt.figure(figsize=(10, 6))

sns.histplot(df['Weekly\_Sales'], bins=30, kde=True)

plt.title('Distribution of Weekly Sales')

plt.xlabel('Weekly Sales')

plt.ylabel('Frequency')

plt.show()

holiday\_sales = df[df['Holiday\_Flag'] == 1].groupby('Date').mean().reset\_index()

non\_holiday\_sales = df[df['Holiday\_Flag'] == 0].groupby('Date').mean().reset\_index()

*# Plot the mean Weekly Sales for Holiday weeks and Non-Holiday weeks*

plt.figure(figsize=(14, 7))

plt.plot(holiday\_sales['Date'], holiday\_sales['Weekly\_Sales'], label='Holiday Weeks', color='blue', linewidth=2)

plt.plot(non\_holiday\_sales['Date'], non\_holiday\_sales['Weekly\_Sales'], label='Non-Holiday Weeks', color='red', linewidth=2)

plt.title('Mean Weekly Sales Over Time')

plt.xlabel('Date')

plt.ylabel('Mean Weekly Sales')

plt.xticks(rotation=45)

plt.legend()

plt.grid(True)

plt.show()

*# Plotting weekly sales Trends Over Time*

plt.figure(figsize=(14, 7))

sns.lineplot(data=df, x='Date', y='Weekly\_Sales')

plt.title('Weekly Sales Over Time')

plt.xlabel('Date')

plt.ylabel('Weekly Sales')

plt.show()

*# Sales vs Temperature graph*

plt.figure(figsize=(10, 6))

sns.scatterplot(data=df, x='Temperature', y='Weekly\_Sales')

plt.title('Weekly Sales vs Temperature')

plt.xlabel('Temperature')

plt.ylabel('Weekly Sales')

plt.show()

*# Plotting weekly sales by Store*

plt.figure(figsize=(14, 7))

sns.boxplot(data=df, x='Store', y='Weekly\_Sales')

plt.title('Weekly Sales by Store')

plt.xlabel('Store')

plt.ylabel('Weekly Sales')

plt.xticks(rotation=90)

plt.show()

*# Checking for outliers using box plots for relevant columns*

plt.figure(figsize=(14, 7))

plt.subplot(2, 3, 1)

sns.boxplot(y=df['Weekly\_Sales'])

plt.title('Weekly Sales')

plt.subplot(2, 3, 2)

sns.boxplot(y=df['Temperature'])

plt.title('Temperature')

plt.subplot(2, 3, 3)

sns.boxplot(y=df['Fuel\_Price'])

plt.title('Fuel Price')

plt.subplot(2, 3, 4)

sns.boxplot(y=df['CPI'])

plt.title('CPI')

plt.subplot(2, 3, 5)

sns.boxplot(y=df['Unemployment'])

plt.title('Unemployment')

plt.tight\_layout()

plt.show()

*# Function to calculate outliers using IQR method*

def calculate\_outliers(data):

Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = data[(data < lower\_bound) | (data > upper\_bound)]

return outliers

*# Calculate outliers for each relevant column*

weekly\_sales\_outliers = calculate\_outliers(df['Weekly\_Sales'])

temperature\_outliers = calculate\_outliers(df['Temperature'])

fuel\_price\_outliers = calculate\_outliers(df['Fuel\_Price'])

cpi\_outliers = calculate\_outliers(df['CPI'])

unemployment\_outliers = calculate\_outliers(df['Unemployment'])

*# Summary of outliers*

outliers\_summary = {

'Weekly\_Sales\_Outliers': weekly\_sales\_outliers,

'Temperature\_Outliers': temperature\_outliers,

'Fuel\_Price\_Outliers': fuel\_price\_outliers,

'CPI\_Outliers': cpi\_outliers,

'Unemployment\_Outliers': unemployment\_outliers

}

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from statsmodels.tsa.arima.model import ARIMA

from prophet import Prophet

from keras.models import Sequential

from keras.layers import LSTM, Dense

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

import seaborn as sns

from datetime import datetime

*# Ensure the 'Date' column is in datetime format*

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

*# Sort the dataframe by date*

df.sort\_values('Date', inplace=True)

*# Set the date as the index*

df.set\_index('Date', inplace=True)

*# Resampling the data by week*

weekly\_data = df['Weekly\_Sales'].resample('W').sum()

*# Function to create training and test datasets*

def create\_train\_test\_split(data, train\_size=0.8):

train\_len = int(len(data) \* train\_size)

train\_data = data[:train\_len]

test\_data = data[train\_len:]

return train\_data, test\_data

*# Train-test split*

train\_data, test\_data = create\_train\_test\_split(weekly\_data)

*# Preprocessing for LSTM*

def preprocess\_lstm(data, time\_step=1):

X, y = [], []

for i in range(len(data) - time\_step - 1):

X.append(data[i:(i + time\_step)])

y.append(data[i + time\_step])

return np.array(X), np.array(y)

*# Scale the data*

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(weekly\_data.values.reshape(-1, 1))

*# Prepare the data for LSTM*

time\_step = 4

X\_train, y\_train = preprocess\_lstm(scaled\_data[:len(train\_data)], time\_step)

X\_test, y\_test = preprocess\_lstm(scaled\_data[len(train\_data):], time\_step)

*# Reshape input to be [samples, time steps, features]*

X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], 1)

X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], 1)

*# LSTM Model*

model = Sequential()

model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)))

model.add(LSTM(50, return\_sequences=False))

model.add(Dense(25))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(X\_train, y\_train, batch\_size=1, epochs=1)

*# Predicting*

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

*# Transform back to original form*

train\_predict = scaler.inverse\_transform(train\_predict)

test\_predict = scaler.inverse\_transform(test\_predict)

*# Calculate RMSE*

train\_score = np.sqrt(mean\_squared\_error(y\_train, train\_predict[:,0]))

test\_score = np.sqrt(mean\_squared\_error(y\_test, test\_predict[:,0]))

print(f'LSTM Train Score: {train\_score:.2f} RMSE')

print(f'LSTM Test Score: {test\_score:.2f} RMSE')

*# Forecasting with ARIMA*

arima\_model = ARIMA(train\_data, order=(5, 1, 0))

arima\_result = arima\_model.fit()

arima\_forecast = arima\_result.forecast(steps=len(test\_data))

*# Calculate RMSE for ARIMA*

arima\_rmse = np.sqrt(mean\_squared\_error(test\_data, arima\_forecast))

print(f'ARIMA Test Score: {arima\_rmse:.2f} RMSE')

*# Forecasting with Prophet*

prophet\_df = weekly\_data.reset\_index().rename(columns={'Date': 'ds', 'Weekly\_Sales': 'y'})

prophet\_model = Prophet()

prophet\_model.fit(prophet\_df.iloc[:len(train\_data)])

future = prophet\_model.make\_future\_dataframe(periods=len(test\_data), freq='W')

prophet\_forecast = prophet\_model.predict(future)

*# Determine the length of test predictions and corresponding x values*

len\_test\_predict = len(test\_predict)

len\_test\_index = len(weekly\_data.index[len(train\_predict) + time\_step:])

*# Adjust the index if necessary*

if len\_test\_index > len\_test\_predict:

x\_values = weekly\_data.index[len(train\_predict) + time\_step:len(train\_predict) + time\_step + len\_test\_predict]

else:

x\_values = weekly\_data.index[len(train\_predict) + time\_step:]

*# Plotting the results*

plt.figure(figsize=(14, 7))

*# LSTM Predictions*

plt.plot(weekly\_data.index, weekly\_data.values, label='True Values')

plt.plot(weekly\_data.index[time\_step:len(train\_predict) + time\_step], train\_predict, label='LSTM Train Predictions')

plt.plot(x\_values, test\_predict, label='LSTM Test Predictions')

*# ARIMA Predictions*

plt.plot(test\_data.index, arima\_forecast, label='ARIMA Forecast')

*# Prophet Predictions*

plt.plot(test\_data.index, prophet\_forecast['yhat'].iloc[-len(test\_data):], label='Prophet Forecast',color='yellow')

plt.title('Sales Forecasting')

plt.xlabel('Date')

plt.ylabel('Weekly Sales')

plt.legend()

plt.show()

*# RMSE for LSTM*

train\_score = np.sqrt(mean\_squared\_error(y\_train, train\_predict[:,0]))

test\_score = np.sqrt(mean\_squared\_error(y\_test, test\_predict[:,0]))

print(f'LSTM Train Score: {train\_score:.2f} RMSE')

print(f'LSTM Test Score: {test\_score:.2f} RMSE')

*# RMSE for ARIMA*

arima\_rmse = np.sqrt(mean\_squared\_error(test\_data, arima\_forecast))

print(f'ARIMA Test Score: {arima\_rmse:.2f} RMSE')

*# RMSE for Prophet*

prophet\_rmse = np.sqrt(mean\_squared\_error(test\_data, prophet\_forecast['yhat'].iloc[-len(test\_data):]))

print(f'Prophet Test Score: {prophet\_rmse:.2f} RMSE')

import matplotlib.pyplot as plt

import numpy as np

*# Labels for the x-axis*

labels = ['LSTM Train', 'LSTM Test', 'ARIMA Test', 'Prophet Test']

*# RMSE values*

rmse\_values = [train\_score, test\_score, arima\_rmse, prophet\_rmse]

*# Plotting the RMSE comparison*

plt.figure(figsize=(10, 6))

bars = plt.bar(labels, rmse\_values, color=['blue', 'blue', 'green', 'orange'])

*# Adding the RMSE values on top of the bars*

for bar in bars:

yval = bar.get\_height()

plt.text(bar.get\_x() + bar.get\_width()/2, yval, round(yval, 2), ha='center', va='bottom')

*#Add padding to avoid overlap*

plt.title('Comparison of RMSE for LSTM, ARIMA, and Prophet Models', pad=30)

plt.ylabel('RMSE')

plt.ylim(0, max(rmse\_values) + 500)

plt.show()

df.head()

import pandas as pd

import numpy as np

from prophet import Prophet

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

*# Reset index to get 'Date' as a column*

df.reset\_index(inplace=True)

*# Ensure the 'Date' column is in datetime format*

df['Date'] = pd.to\_datetime(df['Date'])

*# Sort the dataframe by date*

df.sort\_values('Date', inplace=True)

*# Set the date as the index*

df.set\_index('Date', inplace=True)

*# Resampling the data by week*

weekly\_data = df['Weekly\_Sales'].resample('W').sum()

*# Reset index for Prophet*

prophet\_df = df.resample('W').sum().reset\_index().rename(columns={'Date': 'ds', 'Weekly\_Sales': 'y'})

*# Adding additional regressors and handling NaN values*

prophet\_df['Temperature'] = df['Temperature'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')

prophet\_df['Fuel\_Price'] = df['Fuel\_Price'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')

prophet\_df['Holiday\_Flag'] = df['Holiday\_Flag'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')

prophet\_df['CPI'] = df['CPI'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')

prophet\_df['Unemployment'] = df['Unemployment'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')

*# Create Prophet model*

prophet\_model = Prophet()

*# Add the additional regressors*

prophet\_model.add\_regressor('Temperature')

prophet\_model.add\_regressor('Fuel\_Price')

prophet\_model.add\_regressor('Holiday\_Flag')

prophet\_model.add\_regressor('CPI')

prophet\_model.add\_regressor('Unemployment')

*# Fit the model*

prophet\_model.fit(prophet\_df)

future = prophet\_model.make\_future\_dataframe(periods=len(weekly\_data), freq='W')

*# Add the same additional regressors to the future dataframe and handle NaN values*

future['Temperature'] = df['Temperature'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')[:len(future)]

future['Fuel\_Price'] = df['Fuel\_Price'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')[:len(future)]

future['Holiday\_Flag'] = df['Holiday\_Flag'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')[:len(future)]

future['CPI'] = df['CPI'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')[:len(future)]

future['Unemployment'] = df['Unemployment'].resample('W').mean().reset\_index(drop=True).fillna(method='ffill')[:len(future)]

*#Removing rows with missing values from future dataframe*

future.dropna(inplace=True)

*# Prophet forecast*

forecast = prophet\_model.predict(future)

*# Calculate RMSE for the forecasted period*

prophet\_rmse = np.sqrt(mean\_squared\_error(weekly\_data[-len(forecast):], forecast['yhat'].iloc[-len(weekly\_data):]))

print(f'Prophet Test Score with Regressors: {prophet\_rmse:.2f} RMSE')

*#Plotting the Prohet forecast with additional regressors*

plt.figure(figsize=(14, 7))

plt.plot(weekly\_data.index, weekly\_data.values, label='True Values', color='black')

plt.plot(forecast['ds'], forecast['yhat'], label='Prophet Forecast', color='red')

plt.fill\_between(forecast['ds'], forecast['yhat\_lower'], forecast['yhat\_upper'], color='pink', alpha=0.5)

plt.title('Sales Forecast with Prophet')

plt.xlabel('Date')

plt.ylabel('Weekly Sales')

plt.legend()

plt.show()