Faculty 4: Computer Science

Institute for Web Science and Technologies

Data Analysis In The Cloud
Time Series Forecasting

Research Lab Report

submitted by

Dhruv Singh Chandel - 222100671,
Balram Tiwari - 222100583,
Ravi Singh - 222100546,
Ankush Arora - 221202855,
Unmesh Mhatre - 222100576

Supervisor: Prof. Dr. Frank Hopfgartner
Institute for Web Science and Technologies

Assistant Supervisor: Tania Sennikova
Databricks Team

Koblenz, September 2023

Abstract

This report presents a comprehensive examination of Time Series Forecasting through a comparative analysis undertaken by a team of five researchers. The study focused on three diverse domains: Stock, Sales, and Sensor data. We applied four distinct forecasting algorithms—LSTM, ARIMA/SARIMA, XGBoost, and Prophet to gain insights into their respective performance.

Our investigation looks into the world of time series data, aiming to compare performance of the algorithms within these domains. Through rigorous analysis and evaluation, we review the capabilities and limitations of each forecasting algorithm. Our findings provide valuable insights for decision-makers seeking to harness the forecasting power of time series data in diverse domains.

This report serves as a testament to our collective efforts to explore the nuances of time series forecasting, shedding light on the practical implications of employing various algorithms in real-world scenarios.

Contents

List of Figures

1 List of Abbreviations

AAPL Apple Inc.

ACF Autocorrelation Function

ANN Artificial Neural Network

ARIMA AutoRegressive Integrated Moving Average

AvP Actual vs Predicted

CO Carbon Monoxide

CPI Consumer Price Index

IoT Internet of Things

LPG Liquified Petroleum Gas

LSTM Long Short-Term Memory

MSE Mean Squared Error

PACF Partial Autocorrelation Function

PPM Parts Per Million

RMSE Root Mean Squared Error

SARIMA Seasonal AutoRegressive Integrated Moving Average

TSFRESH Time Series Feature extraction based on Scalable Hypothesis tests

UTC Coordinated Universal Time

XGBoost eXtreme Gradient Boosting

yf Yahoo Finance

2 Introduction

Time Series Forecasting is a vital aspect of data analytics, offering the means to forecast future values based on historical data trends. In our pursuit of enhancing our understanding of this complex field, we embarked on a journey that encompassed both theoretical insights and practical applications. This report encapsulates our endeavor to conduct a comprehensive "Time Series Forecasting: A Comparative Analysis."

Our exploration began with a thorough literature review, an essential foundation to comprehend the complexities of time series forecasting. We delved into research papers, seeking wisdom from the experts in this domain, and using their insights to guide our research.

For empirical validation and a deeper understanding of the subject, we worked upon a diverse set of five datasets. These datasets were Rossman Sales Data, Walmart Sales Data, Apple Stock Price, Google Stock Price, and Environmental Sensor Data. Each dataset represented unique challenges and characteristics, reflecting the real-world scenarios where time series forecasting finds its applications.

The key aspects of our research involved benchmarking the performance of these algorithms on the diverse datasets, shedding light on their forecasting capabilities.Root Mean Squared Error (RMSE) served as our trusted metric, quantifying the accuracy of our forecasts and guiding our comparative analysis.

In collaboration with the Databricks team, we harnessed cutting-edge infrastructure and environments to execute our experiments seamlessly. This partnership empowered us to leverage the full potential of our analysis, ensuring robustness and scalability.

Our aim is to share the valuable insights we've gained from this experience, showcasing the diverse applications and forecasting power of these techniques in the field of time series analysis. Through this report, we hope to contribute to the collective knowledge surrounding time series forecasting and provide valuable insights for practitioners and researchers alike. Join us in the following sections as we unravel the icomplex world of time series forecasting, unveiling its potential and limitations in real-world scenarios.

3 Research Methodology

In the initial phase of our research, we conducted an extensive literature review to gain a deeper understanding of the latest techniques and trends in time series forecasting. Subsequently, we carefully selected diverse datasets, covering domains like stock prices, sales data, and sensor readings. Building upon insights from the literature, we carefully handpicked four distinct forecasting algorithms. These algorithms were not merely applied but underwent rigorous optimization, utilizing techniques like windowing, feature extraction, and hyperparameter tuning. This multifaceted approach enabled us to harness the full forecasting potential of each algorithm, leading to valuable insights into their overall performance.

3.1 Literature Review

The foundation of our research was guided by understanding of the existing research done in the domain of time series forecasting. For this, the team had gone through 15 research papers. Some of the prominent papers which helped us the most in our research are listed below.

- Time-series forecasting of seasonal items sales using machine learning[1]
- Forecasting at Scale[2]
- Forecasting Techniques for Time Series from Sensor Data[3]
- Machine learning approach of detecting anomalies and forecasting time-series of IoT devices[4]
- Stock Closing Price Prediction using Machine Learning Techniques[5]
- Applying LSTM to Time Series Predictable through Time-Window Approaches[6]
- Sales Forecasting Using Deep Neural Network And SHAP techniques[7]
- Sales Demand Forecast in E-commerce Using a Long Short-Term Memory Neural Network Methodology[8]
- Time series forecasting using a hybrid ARIMA and neural network model[9]
- A comparison between Hybrid Approaches of ANN and ARIMA for Indian Stock Trend Forecasting[10]

As we concluded our journey through these research papers, we acknowledge the profound impact they have had on our research framework. These papers have been more than informative guides; they've been our mentors, equipping us with the knowledge and tools to navigate the complex landscape of time series forecasting. Their diverse perspectives, novel methodologies, and empirical insights have

undoubtedly helped our understanding of the field. After gaining the knowledge from these sources, we now set forth on our research path, focusing to contribute our insights and innovations to the ever-evolving domain of time series forecasting.

3.2 Dataset Selection

The choice of datasets for time series forecasting is a crucial step in ensuring the relevance and effectiveness of the analysis. In this study, a diverse set of datasets has been selected to provide a comprehensive understanding of forecasting dynamics across various domains. The Walmart sales data was chosen, as it is a big global store and this will help us understand the buying pattern of customers and how promotions affect the sales. The Google and Apple stock data is important because it helps in understanding financial market trends of IT giants. This data helps us see how different events and the overall economy impact stock prices. The Rossman sales data gives us another perspective on retail because it is a drugstore chain and the data helps us compare sales patterns between different types of stores. The IoT sensor data is from smart devices that collect information about the environment like humidity and temperature. Including this dataset helps us in identifying the trends and changes in telemetry systems.

3.3 Algorithm Selection

The process of choosing the four key forecasting algorithms was driven by keeping in mind the complexities of time series data. Drawing from the insights gained during our extensive literature review, we followed a strategic selection process to ensure that our chosen algorithms could effectively address the diverse dataset domains. These selections were not arbitrary but guided by the specific attributes and performance capabilities exhibited by each algorithm.

ARIMA/SARIMA: Our literature review unveiled the power of combining ARIMA and ANN algorithms to capture both linear and non-linear patterns in time series data[9]. However, for datasets exhibiting more complexity with seasonal patterns and yearly trends, we recognized the need for SARIMA. This choice was driven by the desire to equip our analysis with the capability to adeptly handle these intricate patterns.

LSTM: A fundamental motivation for including LSTM in our selection was to compare an algorithm rooted in neural networks. This choice allowed us to assess the performance of a deep learning approach alongside traditional methods.

XGBoost: Our decision to incorporate XGBoost stemmed from its demonstrated superior performance when compared to traditional linear regression algorithms. By including XGBoost, we aimed to explore the benefits of ensemble-based boosting techniques in the context of time series forecasting.

Prophet: The literature review highlighted the effectiveness of Prophet in capturing complex patterns within time series data[11]. Its inclusion in our selection was

informed by its demonstrated ability to handle intricate and multifaceted patterns.

By carefully curating these four algorithms, our methodology explores a diverse range of forecasting techniques, enabling us to find their respective strengths and adaptability across various datasets and real-world scenarios.

3.4 Algorithm Optimization

For performing the algorithm optimization, we started by focusing on key aspects to increase forecasting accuracy. We implemented an effective windowing strategy to split up the time series data[6]. This helped us to capture critical temporal patterns. Following this approach we were able to tailor our models more effectively in accordance with the unique characteristics of each dataset, improving forecasting precision.

Moving on further, we performed extensive feature extraction to create detailed features and reduce dimensionality, enhancing model efficiency. Moreover, hyperparameter tuning played a significant role in clarifying our algorithms. We tuned parameters like learning rate, epoch, verbose et.c. systematically making adjustments in the configuration settings of a model to optimize its performance on a specific task and to improve accuracy in forecasting. These optimizations of the critical model settings, helped our forecasting models achieve peak performance, ultimately giving more accurate and reliable forecasts. These efforts, centered on windowing, feature extraction, and hyperparameter tuning, inclusively contributed to the glory of our algorithm optimization in the context of time series forecasting.

4 Implementation

The implementation section of our study is where the theoretical groundwork transforms into practical insights. In this phase, we dive headfirst into the intricate process of data preprocessing, splitting, windowing, feature extraction, resampling, applying forecasting algorithms, and hyperparameter tuning. Each subsection within this section is dedicated to a specific domain, allowing us to meticulously document the journey through diverse domains.

Sales Domain: Our first stop on this data-driven journey takes us into the realm of sales data. Here, we examine the forecasting capabilities of our chosen algorithms within the context of Rossman Sales Data and Walmart Sales Data. These subsections provide a deep dive into the nuances of retail and sales forecasting, addressing issues like seasonality and promotions.

Stock Domain: Next, we venture into the dynamic world of stock prices. Within this domain, we dissect the performance of forecasting algorithms using Apple Stock Price and Google Stock Price data. These subsections unravel the intricacies of handling financial time series data and the unique challenges it presents.

IoT Sensor Data: Our final destination is the realm of IoT Sensor Data. This subsection explores the application of forecasting algorithms in the context of environmental sensor readings. We navigate through the challenges of working with time series data generated by sensors, shedding light on the intricacies of data preprocessing and algorithmic adaptation to this unique domain.

Within each of these subsections, we'll detail how we tackled data preprocessing, handling missing values, and removing outliers to ensure data integrity. We'll discuss our strategies for splitting the data into training and testing sets, essential for model evaluation. We'll explore windowing techniques that allow us to frame the data effectively for time series forecasting. Feature extraction methods will be elucidated, highlighting how we distilled valuable information from the datasets. We'll address the significance of resampling techniques and how they contribute to model robustness.

As we traverse the terrain of each dataset, we will apply our chosen forecasting algorithms, carefully documenting their performance. Hyperparameter tuning, a critical aspect of fine-tuning algorithmic performance, will also be spotlighted.

4.1 Sales Data

In this section, we will discuss about the time series forecasting implementation for Rossman and Walmart Sales Data.

4.1.1 Rossman Sales Data

Dataset

In time series forecasting for sales domain, the most widely used data is of the Rossman Sales Dataset. In this dataset, there are various features such as store information, promotions, holidays, etc and also provides historical information about the sales of the drugstore. I have used the Rossman sales dataset for performing the time series forecasting on the sales data and forecasting the future prices. In order to build an efficient and accurate forecasting model, I need to understand the dataset and select all the relevant features.

The Rossman Dataset was downloaded from Kaggle Website. The dataset contains two CSV files named as train (1017209,9) and store (1115,10) from 2013 to 2015. Following is the overview of the dataset and what features have been considered for the forecasting:

Feature Name	Data type	Description	Example Values
Store	int	A unique store ID for each store	1,2,..,1115
Date	object	Helpful in analysing daily sales	2015-07-31
Sales	int	This represents the daily sales	5263,5020
Customers	int	Number of customers on each day	555,546
Open	int	The store is open or not	0 or 1
Promo	int	Promotional events at the store	0 or 1
StoreType	object	There are 4 categories of stores	a,b,c,d height

Table 1: Rossman Sales Data Table.

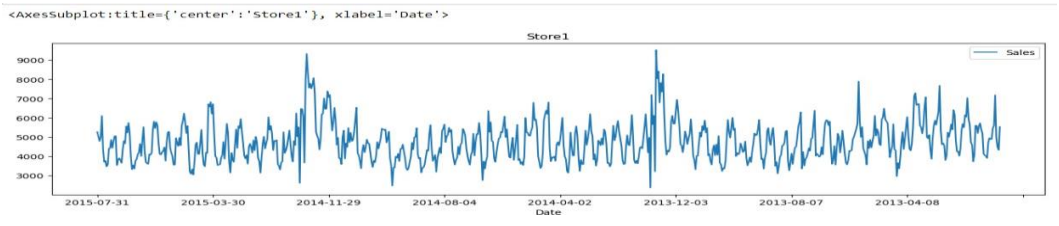

Figure 1: Rossman Raw Data Graph

- **Target Variable**

The Target variable for the Rossman sales forecasting is the Sales feature and it will be forecasted and tested for the future 7 months.

- **Data Pre-processing** During the process of data preparation, we took on several critical steps to make sure that our data is ready for time series forecasting.

  – **Handling Missing Values**

  The null values present in the store dataset were replaced with 0. This step was included because it gave better results as compared to replacing null with the median. Similarly, the null values present in the test dataset were replaced with 1. This replacement was done for the Open feature, considering that all stores are open.

  – **Stationarize the data**

  In linear regression, we know that observations are dependent on each other but for time series, observations are dependent on time. Some independent variables still hold their trends even if there is non-stationarity in the data. To apply time series more efficiently, we can make our data stationary. As we are also implementing the time series forecasting, we used the Dickey Fuller Test to check whether our data is stationary or not.[1]
  The Dickey Fuller Test generates Test Statistics value, p-value and Critical values. To check the data stationarity, we check whether our 'Test Statistic value' is less than the 'Critical Value' or not. In our case, we got the Test statistic value as -6.218 and the Critical values were (-3.437, -2.864, -2.568). Hence, our data is stationary and no need to apply any further process to stationarize the data.

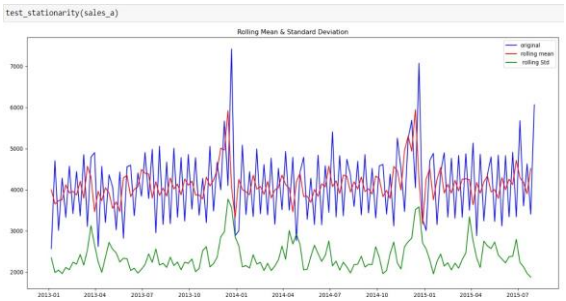

Figure 2: Rossman Rolling Statistics

```
Dickey Fuller test for store type A:
ADF Statistic: -6.218237
p-value: 0.000000
Critical Values:
1% -3.4374778690219956
5% -2.8646868684217556
10% -2.5684454926748583
Dickey Fuller test for store type B:
ADF Statistic: -5.589202
p-value: 0.000001
Critical Values:
1% -3.437485646962348
5% -2.8646901138095378
10% -2.568447319459459
Dickey Fuller test for store type C:
ADF Statistic: -4.374784
p-value: 0.000329
Critical Values:
1% -3.4374778690219956
```

Figure 3: Rossman Dickey Fuller Test Results

- **Data Preparation**
  - **Merging Datasets**

We merged the "train" and "store" datasets on the "Store" feature. This merging allowed for a more comprehensive understanding about the store's performance with the help of additional features about the store.

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	...	PromoInterval	Sales/Customer	Year	Mont
0	1	4	2015-07-31	5263	555	1	1	0	1	3	...		9.48	2015	
1	1	3	2015-07-30	5020	546	1	1	0	1	3	...		9.19	2015	
2	1	2	2015-07-29	4782	523	1	1	0	1	3	...		9.14	2015	
3	1	1	2015-07-28	5011	560	1	1	0	1	3	...		8.95	2015	
4	1	0	2015-07-27	6102	612	1	1	0	1	3	...		9.97	2015	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1017204	1115	5	2013-01-05	4771	339	1	0	0	1	4	...	Mar,Jun,Sept,Dec	14.07	2013	
1017205	1115	4	2013-01-04	4540	326	1	0	0	1	4	...	Mar,Jun,Sept,Dec	13.93	2013	
1017206	1115	3	2013-01-03	4297	300	1	0	0	1	4	...	Mar,Jun,Sept,Dec	14.32	2013	
1017207	1115	2	2013-01-02	3697	305	1	0	0	1	4	...	Mar,Jun,Sept,Dec	12.12	2013	
1017208	1115	1	2013-01-01	0	0	0	0	1	1	4	...	Mar,Jun,Sept,Dec	NaN	2013	

Figure 4: Rossman Merged Dataset (train and store)

  - **Feature Engineering**

Feature engineering steps were included to enhance the dataset's utility. In this we induced some new features with the help of other features and converted the datatype of some features as required.

The following is overview of the steps performed in feature engineering :

1. **Sales/Customers:** The average sales value per customer was calculated for each store, providing insights into customer behavior.
2. **Year and Month:** These were extracted from "Date" column to learn more about the seasonal and annual trends.
3. **Datetime Conversion:** The "Date" column was of the Object data type. So we converted it to datetime for time-based analysis.
4. **Dropping Irrelevant Columns:** All the columns that were irrelevant were dropped for the forecasting task, guided by a heatmap analysis.
5. **Encoding Categorical Variables:** a standard practise was followed for encoding all the categorical features into numerical format.

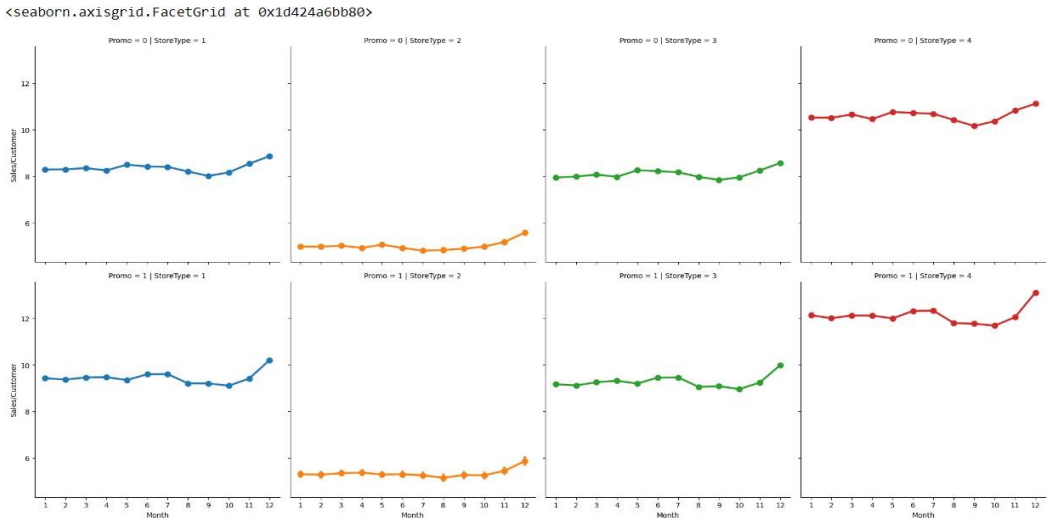

Figure 5: Rossman Average Sales Per Customer

- **Resampling**

To take into consideration the dataset's size and granularity, data resampling was carried out at both weekly and monthly intervals. The volume of data was managed using this technique while maintaining temporal data that was essential for forecasting.

Algorithms

- **SARIMA**
  - **Random Parameter Selection**
    Initially, SARIMA model parameters (p, d, q) were randomly selected. The model was trained on the merged dataset, and sales forecasts were generated for the target period.
  - **Parameter Selection from ACF and PACF Plots**
    Subsequently, an alternative approach was employed.
    Auto Correlation Function (ACF) and Partial Auto Correlation Function (PACF) plots were created from the time series data.
    Visual analysis of these plots was used to determine suitable values for p, d, and q.
    The SARIMA model was trained and forecasts were generated using the parameters identified from the ACF and PACF plots.
  - **Results:**
    When SARIMA parameters were selected randomly, the resulting forecasts yielded RMSE value of 151.4.

These RMSE values were comparatively higher, suggesting that random parameter selection did not capture the underlying patterns in the data effectively.
In contrast, when parameters were chosen based on ACF and PACF plots, the SARIMA model exhibited superior forecasting accuracy.
The RMSE value of 85.28 obtained from this approach were consistently lower, indicating a better fit to the data.

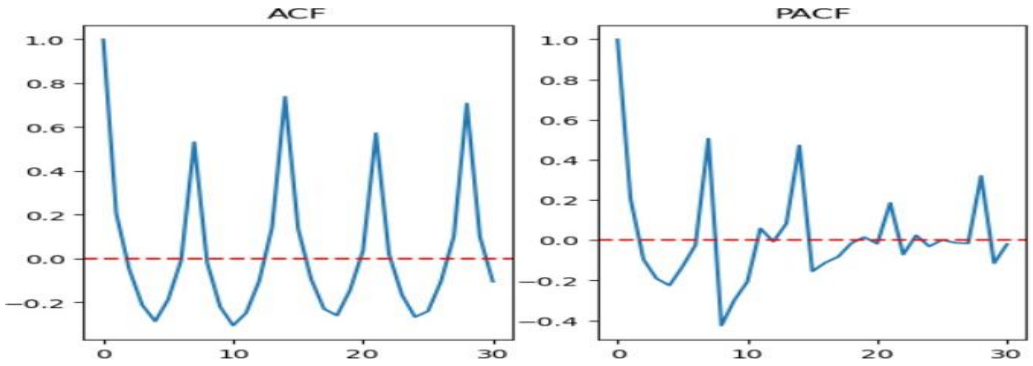

Figure 6: Rossman ACF-PACF values

```
Date
2015-01-01    4056.433014
2015-02-01    4072.163840
2015-03-01    4253.079097
2015-04-01    4033.754787
2015-05-01    3536.277847
2015-06-01    4430.132468
2015-07-01    4375.070051
Freq: MS, Name: predicted_mean, dtype: float64
ye hai org Date
2015-01-01    4112.838710
2015-02-01    4074.214286
2015-03-01    4341.096774
2015-04-01    4033.733333
2015-05-01    3726.935484
2015-06-01    4426.666667
2015-07-01    4315.000000
Freq: MS, Name: Sales, dtype: float64
```

Figure 7: Rossman SARIMA AvP Table

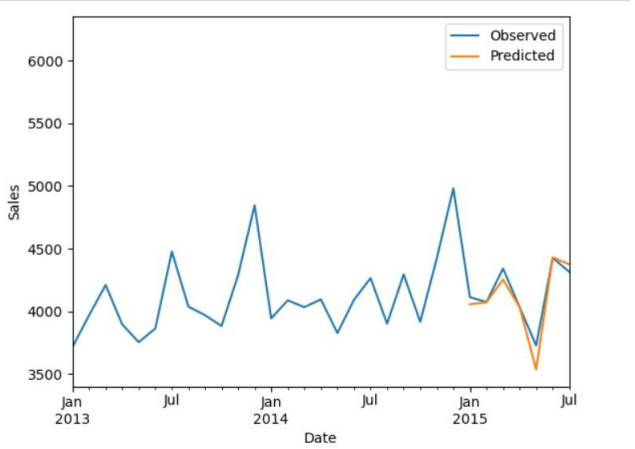

Figure 8: Rossman SARIMA AvP Graph

- **Prophet**
  - **Prophet Modeling - Sales Only**
    Initially, the Prophet model was trained using only the "Sales" feature as the target. Future sales values were forecasted, and the model's forecasting accuracy was assessed using RMSE.

This approach generated an RMSE value of more than 500, which seemed to be a bit higher and not much accurate.
These higher RMSE values suggested that using "Sales" alone did not capture the complexities and contributing factors affecting sales accurately.

- **Prophet Modeling - Sales with Additional Features**
  In a subsequent analysis, the Prophet model was trained with the inclusion of additional relevant features. (Open and Promo)
  These features were identified through a heatmap analysis, highlighting their potential impact on sales.
  Future sales values were forecasted, and RMSE was calculated for evaluation.
  The RMSE value generated was 391.601, which was lower than the initial approach. The model forecasted future sales value that were much closer to the ground truth.
  The RMSE values obtained in this scenario were consistently lower, indicating a more accurate representation of sales trends and patterns.

	ds	yhat	yactual
0	2015-01-01	6119.63	5752.75
1	2015-02-01	6149.57	5710.30
2	2015-03-01	6176.62	5949.13
3	2015-04-01	6206.56	5916.86
4	2015-05-01	6235.53	5472.12
5	2015-06-01	6265.48	6199.20
6	2015-07-01	6294.45	6142.71

Figure 9: Rossman Prophet AvP Table

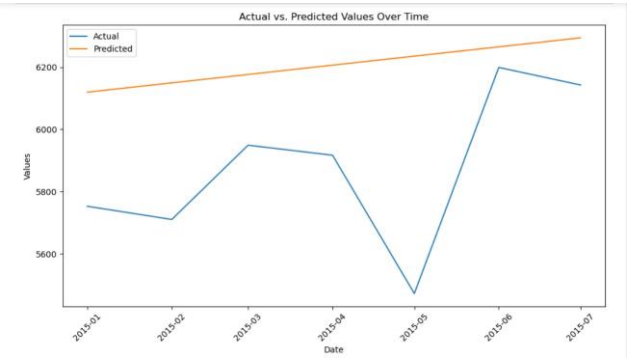

Figure 10: Rossman Prophet AvP Graph

- **LSTM**
  - **Data Reshaping**
    The Rossman sales dataset was reshaped to conform to the input requirements of the LSTM model.
  - **Normalization**
    Data normalization was performed to scale numerical features for better model convergence and training.

- **Windowing** Windowing methodology was applied to format the data into sequences suitable for LSTM input.
- **LSTM Model Building**
  An LSTM model was constructed using Keras with four layers.
  The model architecture was designed to capture temporal dependencies in the data.
- **Training and Initial Predictions**
  The LSTM model was trained on the normalized data in window format.
  Initial predictions were made for future sales values, resulting in an higher RMSE value as a measure of forecasting accuracy.
- **Model Optimization**
  To enhance forecasting performance, model parameters such as epochs, batch size, and verbosity were optimized.
  Iterative adjustments were made to these parameters, leading to more accurate forecasts and lower RMSE value of 157.796.

	Predictions	Actual
0	[3911.14990234375]	[4972.0]
1	[3908.041748046875]	[7006.0]
2	[3915.8388671875]	[6282.0]
3	[3927.39111328125]	[7368.0]
4	[3950.64013671875]	[0.0]
..	...	...
278	[3898.94677734375]	[2342.0]
279	[3907.64990234375]	[4484.0]
280	[3924.9306640625]	[4159.0]
281	[3940.16015625]	[4422.0]
282	[3953.526611328125]	[0.0]

[283 rows x 2 columns]

Figure 11: Rossman LSTM AvP Table

- **XG-BOOST**
  - **Feature Selection**
    Irrelevant columns in the dataset were identified and dropped, streamlining it for the XGBoost model's input.
  - **XGBoost Model Building**
    The XGBoost model was constructed and trained on the processed dataset, with the initial set of parameters.

– **Initial Predictions**
The initial XGBoost model predictions resulted in an RMSE value of 1388.523, providing an initial assessment of forecasting accuracy.

– **Model Parameter Optimization**
To enhance forecasting performance, parameters of the XGBoost model were optimized systematically.
Adjustments to hyperparameters were made to improve model accuracy.

– **Revised Predictions**
After parameter optimization, the XGBoost model generated revised predictions for future sales values.
The RMSE value was recalculated, revealing improved forecasting accuracy with a reduced RMSE value of 1354.448.

Date	Actual Sales	Predicted Sales
2015-01-01	5752.747866	6281.594727
2015-02-01	5710.296541	5846.567383
2015-03-01	5949.130132	5947.620117
2015-04-01	5916.857578	5854.105469
2015-05-01	5472.122002	5175.850586
2015-06-01	6199.203976	6032.577148
2015-07-01	6142.705511	6057.812500

Figure 12: Rossman XG-Boost AvP Table

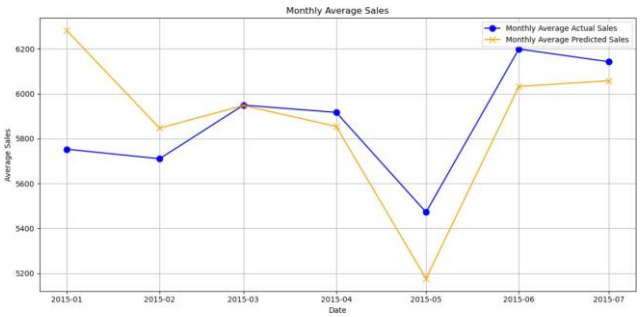

Figure 13: Rossman XG-Boost AvP Graph

Algorithms	SARIMA	LSTM	Prophet	XG-Boost
RMSE	85.28	157.796	391.60	1354.448

Table 2: Rossman Sales Data Algorithm Comparison

4.1.2 Walmart Sales Data

Dataset

The multinational retailer Walmart operates both supermarkets and department stores. For this part of the research, the Walmart dataset was downloaded from the Kaggle platform. The dataset contained two files, namely train and feature. In order to predict the weekly sales, 45 stores were chosen and contains historical sales data from 2010 to 2012.

The dataset contains sales values from multiple stores and their departments. Based on the numerous patterns and trends seen in the prior data, we can forecast store sales.

train.csv: This training data spans from 05/02/2010 to 26/10/2012. It also includes information on dates, weekly sales, and holidays. It has 421571 rows and 5 columns in total.

features.csv: The feature dataset includes extra features that can be helpful in forecasting store sales, including temperature, fuel price, CPI, unemployment, and holiday. There are 12 columns and 8191 rows in it.

Feature Name	Data Type	Description	Example Values
Temperature	Numeric	Temperature for a particular date	42.31, 38.51
Fuel Price	Numeric	Fuel cost on the specified date	2.572
CPI	Numeric	Average price paid by the consumer	211.09
Unemployment	Numeric	Represents the unemployment rate	8.106, 8.106
IsHoliday	Boolean	Holiday falls on the specified date	FALSE, TRUE

Table 3: Walmart Sales Feature Data Table.

Feature Name	Data Type	Description	Example Values
Store	Numeric	Distinct store number	1,2
Dept	Numeric	Department number of store	29,49
Date	Object	Date corresponds to the sales	2010-02-12
Weekly Sales	Numeric	Weekly sales of stores	2714.42, 24924.5
IsHoliday	Boolean	Holiday falls on the specified date.	FALSE, TRUE

Table 4: Walmart Sales Train Data Table.

- **Target Variable**
  Here the target variable is **Sales** feature.

- **Data Pre-Processing**

  In order to make sure that the data quality is enhanced and is ready to forecast, some pre-processing steps were performed.

  – **Preparing Datasets**

  The dataset preparation was done by merging the train and feature datasets for more comprehensive understanding about the store's performance along With some additional features.

	Store	Dept	Date	Weekly_Sales	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI
421540	45	41	2012-10-26	1954.67	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421541	45	42	2012-10-26	4894.78	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421542	45	44	2012-10-26	3144.09	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421543	45	46	2012-10-26	13329.25	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421544	45	52	2012-10-26	1104.16	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421545	45	54	2012-10-26	23.92	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421546	45	55	2012-10-26	3458.17	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421547	45	56	2012-10-26	937.72	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421548	45	58	2012-10-26	275.00	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421549	45	59	2012-10-26	191.98	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899
421550	45	60	2012-10-26	117.00	False	58.85	3.882	4018.91	58.08	100.0	211.94	858.33	192.308899

Figure 14: Walmart Merged Dataset (train and feature)

  – **Handling Missing Values**

The merged dataset contained null values in the "MarkDown1", "MarkDown2", "MarkDown3", "CPI" and "Unemployment" feature columns. Instead of replacing these null values, these rows were eliminated for the better forecasting accuracy.

  – **Handle Duplicate Value**

As part of a crucial preprocessing step, to have good data quality and integrity, duplicate values were eliminated from the dataset to reduce redundancy.

– **Outlier Removals**

Outliers were identified and removed from the dataset to ensure that the analysis will be accurate. This preprocessing step made sure that the underlying patterns and trends are accurately shown, supporting the validity of the future conclusions.

– **Resampling**

Due to the hugeness of the dataset, resampling was done on the data using a weekly interval.

– **Data Splitting**

On the basis of an 80:20 ratio, we divided the resampled data into the Training and Testing sets. 20 percent of the data are assigned to the testing set, and the remaining 80 percent to the training set.

– **Windowing**

With the help of windowing, the time series data was extracted and constructed into data windows of size 30. This captures the historical pattern of the data.

Algorithms

- **SARIMA**
  - **Parameter Selection**
    The model was trained using the training dataset, by randomly selected the SARIMA parameters (p, d, and q), and forecasted the weekly sales for the desired time period.
  - **ACF and PACF parameter selection**
    As an alternative for better results, we have created the ACF and PACF plots for examining the proper p, d, and q parameter values of SARIMA. In the end, the model is evaluated by calculating the RMSE value, which is calculated to be 400.215.

- **PROPHET**
  - **Prophet Modelling and Evaluation**
    In order to forecast future weekly sales for Walmart stores, we trained a prophet model utilizing train data. The model was evaluated using this approach [11] which produced RMSE values of more than 2000.

ds	Actual Sales	Predicted Sales
2012-04-15	10386.491028	10638.590405
2012-04-22	10269.392024	10622.419042
2012-04-29	10116.420124	10385.269387
2012-05-06	10337.295664	10687.983198
2012-05-13	10685.687038	10454.025758
2012-05-20	10871.964559	10360.468344
2012-05-27	10833.467060	10336.156163

Figure 15: Walmart SARIMA AvP Table

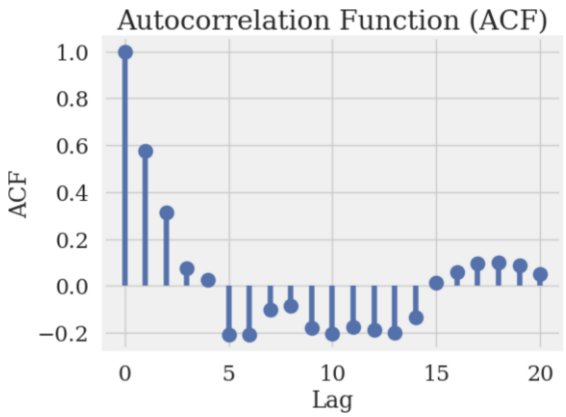

Figure 16: Walmart ACF values

	Actual Sales	Predicted Sales
7	10833.009012	10549.814329
8	10865.805102	10586.368065
9	10936.950215	10644.294677
10	10982.104187	10683.794183
11	10767.328875	10663.077901
12	11004.578985	10616.750323

Figure 17: Walmart Prophet AvP Table

- **Windowing**
  Windowing was used to produce higher accuracy. From the time series, consecutive data windows with a fixed size of 30 was created. This helped in capturing the historical pattern data. The RMSE value came down to 312.41 after using this approach.

- **LSTM**
  To improve training for forecasting weekly sales, the data was first resampled and then normalization was performed on the train and test sets of data.
  The windowing approach was then employed with a window size of 30. Total 50 LSTM neurons were used in this model, which was built using 4 layers(two LSTM layers and two Dense LSTM layers).
  The RMSE is calculated at the end to evaluate the model. The LSTM model

forecasting generated an RMSE value of 108.88.

	Actual Sales	Predicted Sales
0	9683.870761	10394.502930
1	9108.582694	10408.833984
2	9246.933477	10393.782227
3	9178.073087	10362.829102
4	9685.453986	10319.399414
5	10362.511024	10278.865234
6	10671.778309	10250.562500

Figure 18: Walmart LSTM AvP Table

- **XGBOOST**
  Initially, the model yielded a high RMSE value. To improve its performance, I implemented hyperparameter optimization using the Hyperopt tuning technique. This strategic tuning process substantially decreased the RMSE value to 327.614.

The RMSE values representing the performance of each algorithm on the Walmart dataset can be found in the table below.

Algorithms	Prophet	XGBoost	LSTM	SARIMA
RMSE	312.41	327.614	108.88	400.215

Table 5: Walmart Sales Data Algorithm Comparison

4.2 Stock Price Data

In this study, we analyze stock price data from two tech giants, Apple and Google, for forecasting purposes. Apple, renowned for its innovative products, and Google, a leading technology company, represent significant players in the stock market. Examining their stock data allows us to gain insights into the dynamic factors influencing their market performance.

4.2.1 Apple Stock Price

Dataset

This dataset was downloaded from Yahoo Finance via the Python library yfinance. It offers a historical record of stock market data for the stock symbol 'AAPL' (representing Apple Inc.) over the period from January 1, 2017, to December 1, 2022. There is a total of 1,489 data points(Number of Rows). It consists of 6 key features: Open, High, Low, Close, Adj Close, and Volume. Date is acting as the index, facilitating a time series analysis of Apple Inc.'s stock performance.

Feature Name	Data Type	Description	Example Values
Open	Numeric	Opening stock price for the day	28.95, 28.96
High	Numeric	Highest stock price during the day	29.08, 29.12
Low	Numeric	Lowest stock price during the day	28.69, 28.93
Close	Numeric	Closing stock price for the day	29.03, 29.00
Adj Close	Numeric	Adjusted closing stock price	27.05, 27.02
Volume	Numeric	Number of shares traded	115127600, 84472400

Table 6: Apple Stock Data Table.

- **Target Variable**
  The primary focus is on the **Closing Price** of Apple Inc.'s stock. This represents the last traded price at the end of a trading day. The closing price is pivotal for the following reasons:
  Reflects Market Sentiment: It summarizes all market sentiment, trading activities, and daily news, offering insight into how the market collectively perceives the stock's value by day's end.
  Decision-Making Reference: The closing price serves as a vital reference for traders and investors in:
  - Assessing Performance over different time-frames.
  - Establishing stop-loss and take-profit levels.
  - Applying technical analysis tools due to its stability and reliability.

  Noise Reduction: Compared to intraday data, the closing price is less prone to noise, providing valuable data for analysis due to its summary of the entire trading day.

- **Data Pre-processing**

  During the process of data preparation, we took on several critical steps to make sure that our data is ready for time series forecasting.

  - **Handling Missing Values**

  There were no missing values or null entries in the downloaded data. This absence of data gaps simplifies the data preprocessing phase, allowing us to focus on tasks like scaling, encoding, and addressing outliers to enhance the datasets for analysis.

  - **Scaling**

  To ensure that all features are on a consistent scale, a Min-Max scaling technique with a feature range between 0 and 1 was applied to normalize the numerical columns of the datasets.

  - **Data Splitting**

  The dataset was divided into training and testing sets. Approximately 80% of the data was assigned to the training set, while the remaining 20% constituted the testing set.

  - **Windowing**

  In the process of windowing, we created sequential data windows consisting of 60 time steps from the time series data. These windows were constructed to capture historical patterns in the data. Each window, representing a specific segment of the time series, was utilized as input to the model. This technique allowed the model to learn from the sequential patterns within each window, enhancing its ability to make accurate forecasts based on the historical data trends.

Algorithms

- **LSTM**
  - **Model Architecture**
    The LSTM model architecture was meticulously crafted, comprising three stacked LSTM layers. These layers were structured in a sequential manner. The model started with an LSTM layer containing 100 units and configured to return sequences. This was followed by another LSTM layer with 100 units, but this time configured to return a single sequence. After these LSTM layers, the model incorporated two dense layers, the first

```
def build_lstm_model(self, input_shape):
    """Build an LSTM model."""
    model = keras.Sequential()
    model.add(layers.LSTM(100, return_sequences=True, input_shape=input_shape))
    model.add(layers.LSTM(100, return_sequences=False))
    model.add(layers.Dense(25))
    model.add(layers.Dense(1))
    model.summary()
    return model
```

Figure 19: Apple LSTM Model Architecture

one with 25 units and the final output layer with 1 unit. This specific architecture was designed to effectively capture intricate temporal patterns within the data.

- **Model Compilation:**
  Model compilation involved specifying the 'adam' optimizer and Mean Squared Error (MSE) as the loss function. The optimizer determines how the model parameters are updated during training.
- **Model Training**
  The model was trained using the training data with a batch size of 1 over three epochs. This process updated the model's parameters to improve its forecasting capabilities.
- **Model Evaluation**
  After training, the model was evaluated on unseen testing data. Forecasts were scaled back to the original range using Min-Max scaling.
- **Feature Extraction Comparison**
  Initially, without windowing or feature extraction, the model yielded an RMSE of 158.96. Subsequently, feature extraction using the tsfresh library was employed to create diverse features, leading to a substantial RMSE reduction to 3.96. RMSE quantified forecasting accuracy, enabling comparison with actual stock prices.
- **LSTM Model Performance Analysis**
  In Figure 20, we can see that the LSTM model did a really good job. The forecasted values closely match the actual values for each date. There's only a small difference between them, showing that the model is accurate and understands the data well.

Predictions	Actual
[167.796630859375]	[172.99000549316406]
[168.26927185058594]	[175.63999938964844]
[170.25863647460938]	[176.27999877929688]
[172.1123504638672]	[180.3300018310547]
[175.03245544433594]	[179.2899932861328]
...	...
[149.96435546875]	[151.07000732421875]
[150.31414794921875]	[148.11000061035156]
[149.38479614257812]	[144.22000122070312]
[146.97413635253906]	[141.1699981689453]
[143.9905242919922]	[148.02999877929688]

Figure 20: Apple LSTM AvP Closing Stock Prices Table

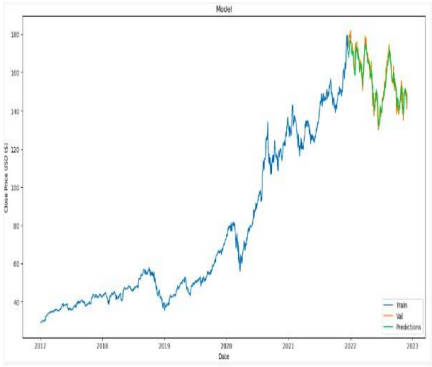

Figure 21: Apple LSTM AvP Closing Stock Prices Graph

- **XGBoost**
  - **Model architecture and Training**
    XGBoost, a powerful gradient boosting algorithm, with hyperparameters including 100 estimators, a learning rate of 0.1, and 'reg:squarederror' as the regression objective, was selected for regression. The model was trained on historical data to make forecasts.
  - **Model Evaluation**
    The forecasted value of the model were scaled back to the original price range. Model accuracy was measured by comparing these forecasts to actual stock prices, with RMSE used to evaluate forecasting quality.
  - **XGBoost Model Performance Analysis**

Predictions	Actual
[141.95843505859375]	[140.91000366210938]
[141.16693115234375]	[143.75999450683594]
[141.81002807617188]	[144.83999633789062]
[143.7239532470703]	[146.5500030517578]
[145.05050659179688]	[148.75999450683594]
...	...
[147.84803771972656]	[151.07000732421875]
[148.93673706054688]	[148.11000061035156]
[147.2613983154297]	[144.22000122070312]
[145.55654907226562]	[141.1699981689453]
[144.71607971191406]	[148.02999877929688]

Figure 22: Apple XG-Boost AvP Closing Stock Prices Table

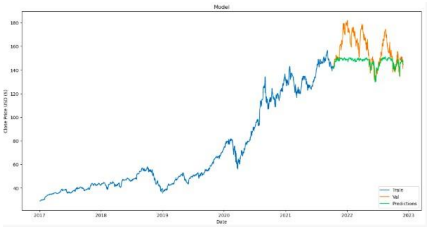

Figure 23: Apple XG-Boost AvP Closing Stock Prices Graph

- **ARIMA**
  - **Model architecture and Training**
    The ARIMA model's order denoted as (7, 2, 4), encompasses autoregressive, differencing, and moving average components respectively. I feed the model with same training data that was used earlier for above two models(LSTM and XGBOOST).
  - **Straight-Line Forecasts**
    My ARIMA model yielded forecast values that formed a seemingly straight line when plotted. This result was puzzling and indicated that our model was facing some challenges.
  - **Non-Stationary Data**
    Upon investigation, we discovered a crucial issue—our data was non-stationary, confirmed by the Dickey-Fuller test with a p-value exceeding the significance level of 0.05. Non-stationary data can hinder the performance of time series models like ARIMA.
  - **ARIMA Model Performance Analysis**

Predictions	Actual
146.731988	145.369995
146.879583	141.910004
147.091373	142.830002
147.207439	141.500000
147.188277	142.649994
...	...
175.858963	151.070007
175.960650	148.110001
176.063795	144.220001
176.156013	141.169998
176.257425	148.029999

Figure 24: Apple ARIMA AvP Closing Stock Prices Table

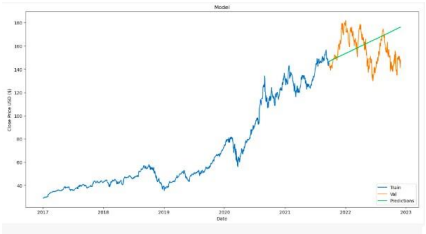

Figure 25: Apple ARIMA AvP Closing Stock Prices Graph

- **Prophet**
  - **Model architecture and Training**
    In the Prophet model, the 'Close' price data served as the target variable labeled 'y.' Additionally, the 'Date' column was transformed into 'ds' to represent the date feature. The labeling was performed to meet the specific requirements of the Prophet model.[12] I fed the model with same

Figure 26: Apple Prophet Closing Stock Price Trend

training data that was used earlier for above three models(LSTM, XG-BOOST and ARIMA).

- **Apple data trend pattern**
  We can clearly see that my Apple stock data shows yearly pattern in the trend, which can be observed using visualizations generated by the Prophet model.
- **Model Evaluation**
  In the evaluation of the Prophet model, we obtained an RMSE value of 23.09.
- **Model Performance Analysis**

Predictions	Actual
149.010174	145.369995
149.395337	141.910004
149.643559	142.830002
149.858901	141.500000
149.994940	142.649994
...	...
189.980507	151.070007
189.738537	148.110001
189.809706	144.220001
189.978985	141.169998
190.009637	148.029999

Figure 27: Apple Prophet AvP Closing Stock Prices Table

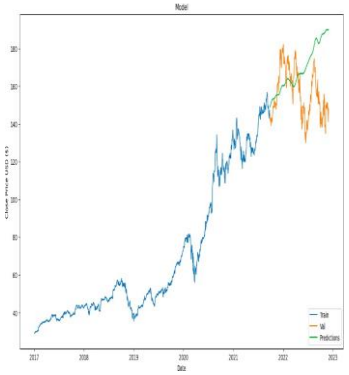

Figure 28: Apple Prophet AvP Closing Stock Prices Graph

Algorithms	LSTM	XG-Boost	ARIMA	Prophet
RMSE	3.96	14.34	17.85	23.09

Table 7: Apple Stock Data Algorithm Comparisons

4.2.2 Google Stock Price

Dataset

The dataset used in this analysis was downloaded from Kaggle and offers the daily record of stock market data for Google over the period from April, 2013 to March, 2023. The dataset contains 2510 data points(Number of Rows), each representing a specific day. The dataset has 7 columns namely : Date, Price, high, Low, Close, Volume and Adj Close.

Feature Name	Data Type	Description	Example Values
Date	Date	Date of Stock Market Operations	2013-04-15, 2023-03-31
Price	Numeric	Price of stock for the day	19.67, 101.30
High	Numeric	Highest stock price during the day	19.94, 103.89
Low	Numeric	Lowest stock price during the day	19.44, 101.04
Close	Numeric	Closing stock price for the day	19.57, 103.73
Adj Close	Numeric	Adjusted closing stock price	19.57, 103.73
Volume	Numeric	Number of shares traded	98025876, 36823200

Table 8: Google Stock Daily Data Table.

- **Target Variable**
  For this part of the research, the primary focus is on the "Adj Close" of Google stock price daily data. This represents the adjustments made after last traded price at the end of a trading day accounting for the bonds and after market settlements.

- **Data Pre-processing**

  During the process of data preparation, the main focus was on preparing the data in such a way that would help in getting the best forecasting.

  – **Handling Missing Values**

  The Dataset was downloaded from kaggle and was already preprocessed as there was no null value or outliers.

  – **Scaling**

  To ensure that all features are on a consistent scale, a Min-Max scaling technique with a feature range between 0 and 1 was applied to normalize the columns like volume as the trade volume of the stock had very high values(in millions) that could have a affect on the model's accuracy.

– **Data Splitting**

The dataset was splitted into train-test split of 80%-20% which is an industry standard approach for training and validation of majority of the Machine Learning Algorithms.

– **Windowing**

I applied a windowing technique with a seven-day window size. This approach segmented the data into consecutive seven-day intervals, allowing me to capture short-term patterns and trends within the stock price movements.

Algorithms Performance

- **LSTM**

  In the context of our analysis, the LSTM algorithm, while exhibiting notable capabilities, presented certain limitations. It was not as effective as XGBoost in capturing the underlying patterns within the Google Stock Price dataset (based on the RMSE values). The LSTM model's forecasting displayed a degree of variability when compared to the actual stock prices, resulting in an RMSE value of 26.740. The "AvP Price" table(Figure 29) and accompanying graph(Figure 30) offer a visual representation of these forecasts, providing insight into the model's performance. While LSTM demonstrated its potential in capturing temporal dependencies[6] in time series data, our findings suggest that, in this particular context, it fell short in comparison to the forecasting accuracy achieved by XGBoost.

	Date	Actual Price	Predicted Price
0	2021-04-06	110.46	103.813240
1	2021-04-07	111.95	103.989853
2	2021-04-08	112.52	104.190727
3	2021-04-09	113.53	104.389084
4	2021-04-12	112.23	104.591309
..	...	...	...
497	2023-03-27	102.46	140.025833
498	2023-03-28	101.03	140.026611
499	2023-03-29	101.39	140.027374
500	2023-03-30	100.89	140.028137
501	2023-03-31	103.73	140.028885

Figure 29: Google LSTM AvP Adj Close Table

Figure 30: Google LSTM AvP Adj Close Graph

- **XGBoost**

  Within the analysis of Google Stock Price daily dataset, the XGBoost algorithm emerges as a standout performer in the forecasting of "Adj Close" Price. It notably outperforms all other algorithms, demonstrating unparalleled forecasting accuracy. The remarkable efficacy of XGBoost is evident in its RMSE

	Date	Actual Price	Predicted Price
0	2021-04-07	110.65	106.969437
1	2021-04-08	113.20	106.101112
2	2021-04-09	112.27	105.453400
3	2021-04-12	112.71	105.673531
4	2021-04-13	112.55	105.600540
...	...	...	...
496	2023-03-27	104.62	105.973297
497	2023-03-28	102.44	104.409950
498	2023-03-29	102.28	101.999985
499	2023-03-30	100.91	101.913017
500	2023-03-31	101.30	101.726753

Figure 31: Google XG-Boost AvP Adj Close Table

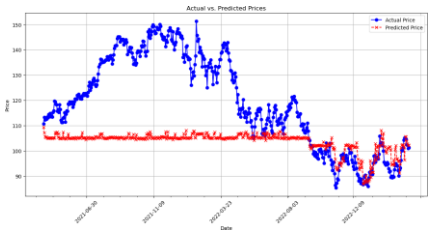

Figure 32: Google XG-Boost AvP Adj Close Graph

value, which impressively stands at 22.417. The "AvP Price" table(Figure 31) and accompanying graph(Figure 32) in this subsection vividly illustrate the precision of XGBoost's forecasts. This algorithm's ability to identify complex patterns and non-linearities within the time series data sets it apart as a powerful tool for time series forecasting, particularly in capturing the complexities of daily stock price fluctuations.

- **ARIMA**

	Date	Actual Price	Predicted Price
0	2021-04-06	110.46	110.250175
1	2021-04-07	111.95	110.660183
2	2021-04-08	112.52	110.536216
3	2021-04-09	113.53	110.456911
4	2021-04-12	112.23	109.977068
..	...	...	...
497	2023-03-27	102.46	132.320483
498	2023-03-28	101.03	132.365745
499	2023-03-29	101.39	132.411008
500	2023-03-30	100.89	132.456271
501	2023-03-31	103.73	132.501534

Figure 33: Google ARIMA AvP Adj Close Table

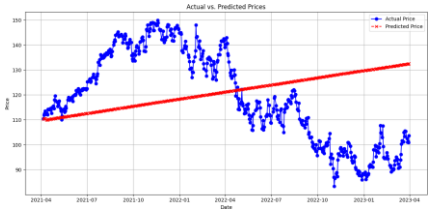

Figure 34: Google ARIMA AvP Adj Close Graph

In our examination of the ARIMA (AutoRegressive Integrated Moving Average) algorithm within the context of the Google Stock Price dataset, we observed some distinct characteristics. ARIMA, while a robust tool for time series forecasting, showcased certain limitations in capturing the complexities present in daily stock price dynamics[13]. This algorithm may excel when applied to datasets with clear linear or seasonal patterns. However, it encounters challenges when dealing with datasets having more complex and non-

linear relationships. The RMSE value, which provides a quantitative measure of forecasting accuracy, arrived at 23.442. The "AvP Price" table(Figure 33) and accompanying graph(Figure 34) in this subsection visually encapsulate the algorithm's performance, highlighting its strengths and limitations in finding the patterns of daily stock price movements. These findings highlight the importance of algorithm selection tailored to the specific attributes of the dataset, recognizing that while ARIMA can be a valuable tool, it may not be the ideal choice for every forecasting scenario.

- **PROPHET**
  Within our analysis, the PROPHET algorithm stands out as a notably user-friendly forecasting tool. PROPHET's ease of use and intuitive application made it a valuable addition to our arsenal of forecasting methods. While its simplicity in implementation is a commendable aspect, it's important to note that PROPHET, in our study, demonstrated an RMSE value of 27.148. This metric, which quantifies forecasting accuracy, provides valuable insights into the algorithm's performance. The "AvP Price" table(Figure 35) and accompanying graph(Figure 36) in this subsection visually convey the algorithm's forecasting capabilities. While PROPHET's RMSE value suggests room for improvement in terms of forecasting accuracy, its user-friendly nature makes it an attractive choice for practitioners seeking a straightforward forecasting solution. This balance between ease of use and forecasting performance positions PROPHET as a valuable tool, particularly for applications where simplicity and quick deployment are paramount.

	Date	Actual Price	Predicted Price
0	2021-04-06	110.46	93.324397
1	2021-04-07	111.95	93.385392
2	2021-04-08	112.52	93.420315
3	2021-04-09	113.53	93.416800
4	2021-04-12	112.23	92.533140
...	...	...	...
497	2023-03-27	102.46	127.222553
498	2023-03-28	101.03	127.332245
499	2023-03-29	101.39	127.416820
500	2023-03-30	100.89	127.462796
501	2023-03-31	103.73	126.627246

Figure 35: Google Prophet AvP Adj Close Table

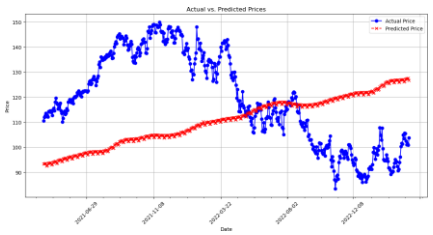

Figure 36: Google Prophet AvP Adj Close Graph

Algorithm Optimization

To fine-tune the performance of the forecasting algorithms on Google Stock Data, various optimization techniques were implemented. A pivotal aspect of this optimization process was the utilization of parallel computing techniques, specifically leveraging SparkTrials within the Databricks environment. This approach enabled us to run hyperparameter optimization tasks in parallel across multiple Spark nodes, significantly expediting the parameter search process(Figure 37).

One of the key techniques for optimization was the fmin() function, which seamlessly capitalizes on Spark's distributed computing capabilities. The parallelization of hyperparameter tuning tasks allowed us to explore a vast search space efficiently.

Notably, the optimization techniques were most beneficial in the case of the XGBoost algorithm, where the implementation of optimized parameters led to a substantial 8% reduction in the RMSE.

```
In [ ]: spark_trials = SparkTrials(4)
max_evals_spark = 100

with mlflow.start_run():
  argmin = fmin(
    fn=XGBoostFunc,
    space=space,
    algo=rand.suggest,
    max_evals=max_evals_spark,
    trials=spark_trials)
```

Figure 37: Google Code Snippet Spark Trials for Hyperparameter tuning

The RMSE values representing the performance of each algorithm can be found in the table below.

Algorithms	Prophet	XGBoost	LSTM	ARIMA
RMSE	27.148	22.417	26.740	23.442

Table 9: Google Stock Data Algorithm Comparison

4.3 IoT Sensor Data

Dataset

The dataset was downloaded from Kaggle, providing insights into environmental parameters such as carbon monoxide levels(co), humidity(humidity), light intensity(light), LPG concentration(lpg), motion detection(motion), smoke presence(smoke), and temperature(temp) at a certain timestamp(ts). The data was generated from a series of three identical, custom-built, breadboard-based sensor arrays(device). Each array was connected to Raspberry Pi devices. Each of the three IoT devices was placed in a physical location with varied environmental conditions. Each IoT device collected a total of seven different readings on a regular interval. The data spans the period from 07/12/2020 00:00:00 UTC – 07/19/2020 23:59:59 UTC. There is a total of 405,184 rows of data. This wealth of information holds significant potential for time series forecasting, allowing for the forecasting of temperature trends with precision and accuracy.

Column	Description	Units	Example Values
ts	timestamp of event	epoch	1.594e+09,1.595e+09
device	unique device name	string	b8:27:eb:bf:9d:51,00:0f:00:70:91:0a
co	carbon monoxide	ppm(%)	0.00495,0.00591
humidity	humidity	percentage	51,75.3
light	light detected?	boolean	TRUE,FALSE
lpg	liquid petroleum gas	ppm(%)	0.00765,0.00624
motion	motion detected?	boolean	FALSE
smoke	smoke	ppm(%)	0.0234,0.016
temp	temperature	fahrenheit	22.7,19.2

Table 10: IoT Sensor Data

- **Pre-Processing**

1. **Removal of Outliers**
   Outliers, which can introduce noise and skew predictions, were systematically identified and removed from the dataset. This crucial step ensures that the forecasting model is trained on reliable and representative data.

2. **Removal of 'motion' Column**
   The 'motion' column, containing solely false values, is omitted from the dataset to eliminate noise and enhance the forecasting model's performance. And I dropped the device column because it's not crucial for forecasting.

3. **Correlation-based Feature Selection**
   Highly correlated variables, namely 'LPG', 'smoke', and 'CO', are excluded

from the dataset to mitigate multicollinearity and prevent redundancy in the forecasting process.

4. **Timestamp Conversion**
   The 'timestamp' column is converted to datetime format to facilitate temporal analysis.

5. **Resampling to Per-Minute Data**
   To align the data with the desired forecasting model, the remaining data is resampled to per-minute intervals. This adjustment ensures a harmonious fit between the data and the forecasting algorithm.

	time	humidity	temp
0	2020-07-12 00:01:00	64.071429	23.764286
1	2020-07-12 00:02:00	64.750000	23.113889
2	2020-07-12 00:03:00	65.816216	22.956757
3	2020-07-12 00:04:00	66.028947	22.881579
4	2020-07-12 00:05:00	64.430303	22.924242
...	...	...	...
11518	2020-07-19 23:59:00	62.268571	21.997143
11519	2020-07-20 00:00:00	61.687500	22.928125
11520	2020-07-20 00:01:00	63.405883	22.561765
11521	2020-07-20 00:02:00	64.347369	22.539474
11522	2020-07-20 00:03:00	63.137501	22.458334

11523 rows × 3 columns

Figure 38: Sensor Data Processed

- **Selected Features**

The features identified for forecasting temperature values are **humidity** and **temp**. And the **target variable** is **temp**. These variables exhibit a significant influence on temperature fluctuations and are crucial in modeling accurate forecasts. By judiciously selecting and processing features, we aim to optimize the accuracy and efficacy of the forecasting model. The transformed dataset, equipped with pertinent environmental variables, is poised for further analysis and forecasting of temperature trends with heightened precision.

Algorithms

The accuracy of temperature forecasting is crucial for various applications, ranging from climate control systems to environmental monitoring. LSTM, with its ability to capture intricate temporal dependencies is a powerful tool for handling IoT sensor data[14]. Feature engineering was carried out using the TSFRESH library and Random Forest Regressor to select the most influential features. The dataset was partitioned into training and testing sets utilizing a rolling window approach. The models were subsequently trained, evaluated, and compared based on RMSE metric. The results provide valuable insights into the strengths and weaknesses of each algorithm in the context of temperature forecasting using sensor data.

- **Methodology**

1. **Feature Engineering**
   -The TSFRESH library was employed for automated feature extraction, capturing the intrinsic characteristics of the sensor data.
   -The Random Forest Regressor was utilized to select the top 20 features based on their importance for temperature.

2. **Data Splitting**
   The dataset was partitioned into training and testing sets utilizing a rolling window approach, ensuring that the models were evaluated on dynamically changing data. And the size of the window is 60.

3. **Hyperparameter Tuning**
   Hyperparameter optimization was employed to fine-tune the parameters of each algorithm.

```
# Define Hyperopt optimization algorithm
trials = Trials()
best = fmin(fn=objective, space=space, algo=tpe.suggest, max_evals=20, trials=trials)

# Retrieve best hyperparameters
best_n_estimators = int(best['n_estimators'])
best_max_depth = int(best['max_depth'])
best_learning_rate = best['learning_rate']
best_min_child_weight = int(best['min_child_weight'])
best_subsample = best['subsample']
best_colsample_bytree = best['colsample_bytree']

# Build and train the final model with the best hyperparameters
final_model = xgb.XGBRegressor(
    n_estimators=best_n_estimators,
    max_depth=best_max_depth,
    learning_rate=best_learning_rate,
    min_child_weight=best_min_child_weight,
    subsample=best_subsample,
    colsample_bytree=best_colsample_bytree,
    objective='reg:squarederror'
)
```

Figure 39: Sensor Data Hyperopt Algorithm

- **Models**

1. **Prophet**
   This model is forecasting 2305 minutes in the future which is around 2 days of data. Here "ds" is the timestamp, "yhat" is the forecast and "yhatupper" and "yhatlower" is the range of the forecast. Prophet distinguishes itself as an accessible and intuitive forecasting tool. While its simplicity offers a user-friendly experience, it achieved an RMSE value of 0.582 in our study, indicating its forecasting accuracy(fig. 40). The visual representation of actual versus predicted values shows Prophet's forecasting capabilities(fig. 41). Its balance between ease of use and forecasting performance positions Prophet as a compelling choice for practitioners seeking straightforward forecasting solutions.

```
predictions = forecast.iloc[-2305:]['yhat']
print("RMSE:", rmse(predictions, test['y']))
print("mean:", test['y'].mean())
result_df = pd.DataFrame({'Actual Values': test['y'], 'Predicted Values': predictions})
print(result_df)

RMSE: 0.5824673004543026
mean: 22.838777259724854
      Actual Values  Predicted Values
9218       22.057143         22.195610
9219       21.915152         22.195248
9220       21.821212         22.194899
9221       21.977778         22.194563
9222       21.903030         22.194241
...              ...               ...
11518      21.997143         23.922420
11519      22.928125         23.921271
11520      22.561765         23.920124
11521      22.539474         23.918979
11522      22.458334         23.917835

[2305 rows x 2 columns]
```

Figure 40: Sensor Data Prophet AvP

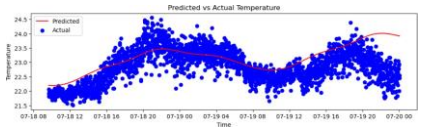

Figure 41: Sensor Data Prophet Visualization

2. **XGBoost**
   XGBoost showcases robustness and adaptability in IoT sensor data forecasting, leveraging an ensemble learning approach[15]. Although in our study, it yielded an RMSE value of 2.859(fig. 42), indicating room for improvement in forecasting accuracy. Nevertheless, its flexibility and proficiency in handling diverse data types make XGBoost a valuable model for IoT forecasting applications.

```
100%|████████| 20/20 [08:37<00:00, 25.88s/trial, best loss: 2.85912120976228]
Root Mean Square Error (RMSE): 2.85912120976228
       Actual Values  Predicted Values
0          22.100000         22.386740
1          19.799999         21.381012
2          23.600000         22.760618
3          22.100000         22.022015
4          22.100000         20.910036
...              ...               ...
80977      19.200001         20.563622
80978      22.200000         22.524492
80979      26.600000         23.911810
80980      19.200001         21.107914
80981      22.200000         22.270702

[80982 rows x 2 columns]
```

Figure 42: Sensor Data XGBoost AvP

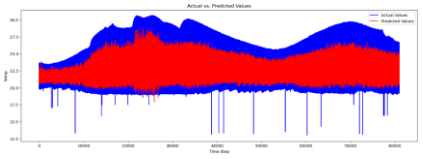

Figure 43: Sensor Data XGBoost Visualization

3. **LSTM**

   LSTM stands out as strong contender in forecasting IoT sensor data, harnessing deep learning capabilities. Impressively, it demonstrated an RMSE value of 0.1138 in this study, attesting to its high level of forecasting accuracy. The graphical depiction of actual versus forecasted values vividly showcase LSTM's forecasting prowess(fig. 45). With such precision, LSTM is an excellent option for practitioners prioritizing accurate forecasts, especially in scenarios involving intricate temporal dependencies.

```
# Build the LSTM model with best hyperparameters
best_model = build_lstm_model(input_shape)
optimizer = best['optimizer']
if optimizer == 0:
    opt = keras.optimizers.Adam()
elif optimizer == 1:
    opt = keras.optimizers.SGD()
else:
    opt = keras.optimizers.RMSprop()

best_model.compile(optimizer=opt, loss='mean_squared_error')

# Fit the best model to the training data
best_model.fit(X_train_lstm, y_train_lstm, batch_size=best['batch_size'], epochs=best['epochs'], verbose=2)

# Make predictions using the best model
predictions = best_model.predict(X_test_lstm)
```

Figure 44: Sensor Data LSTM Training

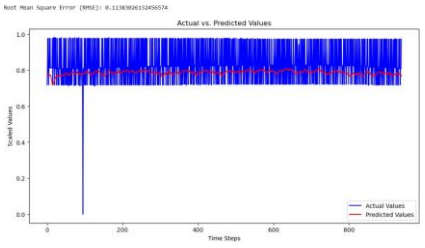

Figure 45: Sensor Data LSTM Visualization

4. **ARIMA**

   ARIMA maintains its status as a steadfast and well-established forecasting method for IoT sensor data. ARIMA demonstrated an RMSE value of 1.140(fig. 46), denoting a respectable level of forecasting accuracy. While there is potential for enhancement, ARIMA's enduring methodology and effectiveness in capturing linear trends affirm its value for IoT forecasting. Its reliability and straightforward application positions it as a compelling choice for practitioners seeking a trusted forecasting solution.

```
Root Mean Square Error (RMSE): 1.1409989601523622
      Actual Values  Predicted Values
0         21.885714         21.861122
1         21.844118         21.864766
2         21.805556         21.862862
3         21.800000         21.858994
4         22.005714         21.857578
...             ...               ...
2288      21.997143         21.859984
2289      22.928125         21.859984
2290      22.561765         21.859984
2291      22.539474         21.859984
2292      22.458334         21.859984

[2293 rows x 2 columns]
```

Figure 46: Sensor Data ARIMA AvP

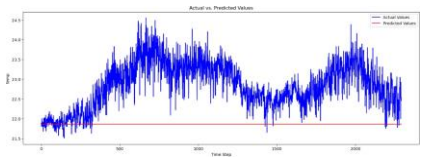

Figure 47: Sensor Data ARIMA Visualization

Comparison

After comparing the RMSE values for different algorithms, we can draw the follow-

ing conclusions:

1. LSTM demonstrated the lowest RMSE of 0.113, indicating that it provided the most accurate forecasts among the algorithms tested.
2. Prophet also showed strong performance with an RMSE of 0.582, suggesting its effectiveness in capturing trends and seasonal patterns.
3. ARIMA exhibited moderate performance with an RMSE of 1.140, indicating that it performed adequately, though there may be room for improvement.
4. XGBoost produced the highest RMSE of 2.859, suggesting it may not be the most suitable algorithm for this particular dataset.

Based on these results, we recommend considering LSTM or Prophet for future forecasts, given their lower RMSE values and thus, higher accuracy.

Algorithms	Prophet	XGBoost	LSTM	ARIMA
RMSE	0.582	2.859	0.113	1.140

Table 11: Sensor Data Algorithm Comparison

5 Collective Results

This section serves as the culmination of our meticulous exploration into time series forecasting. In this section, we bring together the performance metrics, specifically the RMSE, of all aforementioned forecasting algorithms across the 5 datasets. These RMSE values encapsulate the forecasting accuracy of each algorithm, shedding light on their strengths and weaknesses in real-world scenarios. As we dive into the results, we aim to get meaningful insights that resonate not only with the data we analyzed but also with the broader landscape of time series forecasting.

Dataset	ARIMA/SARIMA	LSTM	Prophet	XGBoost
Apple Stock Data	17.85	3.96	23.09	14.34
Google Stock Data	23.442	26.740	27.148	22.417
Rossman Sales Data	85.28	157.79	391.601	1354.44
Wallmart Sales Data	400.21	108.88	312.41	327.61
IoT Sensor Data	1.140	0.113	0.582	2.859

Table 12: Performance Comparison(RMSE Values) of Forecasting Algorithms

For Google Stock Data, we observed that XGBoost exhibited remarkable forecasting capabilities, delivering low RMSE value. Whereas for apple stock data, LSTM performed well as compared to other algorithms with a lower RMSE value. In the realm of Sales Domain, ARIMA/SARIMA emerged as a robust performer, offering precise forecasts with minimal error, specially for Rossman Sales Dataset. In contrast, LSTM showcased its prowess in handling the complexities of IoT Sensor Data, giving accurate forecasts.

Turning our attention to the retail domain, the Rossman Sales Data posed unique challenges, where Prophet demonstrated its versatility by yielding competitive RMSE scores. Wallmart Sales Data presented a scenario where XGBoost excelled, showcasing its adaptability in handling diverse datasets.

Collectively, these results underscore the importance of algorithm selection in the context of specific datasets and applications. Furthermore, they highlight the significance of iterative optimization, such as hyperparameter tuning, in enhancing forecasting accuracy.

As we conclude this section, it becomes evident that there is no one-size-fits-all solution in the realm of time series forecasting. Instead, our journey through diverse datasets and algorithms has illuminated the value of a tailored approach, where algorithm selection and fine-tuning align with the unique characteristics of the data at hand. These insights not only enrich our understanding of forecasting but also provide a pragmatic roadmap for practitioners seeking to harness the power of time

series analysis in real-world scenarios.

5.1 Key Takeaways

Our in-depth exploration into time series forecasting has yielded several key takeaways that encapsulate the essence of our research:

Algorithmic Excellence: In the realm of stock data forecasting, LSTM and XGBoost emerged as standout performers, delivering precise forecasts. These algorithms showcased their ability to capture the complexities of financial markets, providing valuable tools for investors and analysts.

Sales Data Mastery: SARIMA, a go to industry standard in time series forecasting, stood out as the top performer for sales data. Its adeptness in capturing seasonality and trends within retail data makes it an invaluable asset for demand forecasting and inventory management.

Sensor Data Precision: In the complex domain of IoT sensor data, LSTM once again demonstrated its prowess. It exhibited the capacity to handle intricate and time-varying patterns, delivering accurate forecasts. This makes LSTM an indispensable tool for industries relying on sensor data for decision-making.

Accessible Resources: We've made our research accessible to the community. Our code and documentation can be found on our GitHub repository, allowing fellow researchers and practitioners to replicate our experiments and build upon our findings. You can access the repository at: GitHub - Time Series Forecasting. These takeaways encapsulate the practical insights gained from our comparative analysis. They emphasizes the importance of algorithm selection tailored to specific data domains and underscore the significance of optimization in achieving accurate time series forecasts.

6 Future Directions

While our current research has provided valuable insights into time series forecasting, there remain exciting avenues for future exploration and advancement in this field. Here are some promising directions for future work:

Ensemble Methods: One promising direction is the exploration of ensemble methods. Combining the strengths of multiple forecasting algorithms can lead to enhanced forecasting accuracy and robustness, especially in complex, noisy datasets. Investigating ensemble techniques, such as model stacking or weighted averaging, can potentially yield superior forecasting results and mitigate the limitations of individual algorithms.

Interpretability: Incorporating interpretability into forecasting models is crucial for gaining insights into model decisions. Future research in this area can bridge the gap between model forecasts and actionable insights. Developing techniques that provide explanations for forecasting outcomes can empower decision-makers to understand the factors driving forecasts, leading to more informed and effective decision-making processes.

These future directions underscore the evolving nature of time series forecasting research. As the field continues to advance, embracing ensemble methods and enhancing model interpretability can contribute to more accurate and actionable forecasting models, ultimately benefiting a wide range of applications in finance, retail, and IoT, among others.

References

[1] Yasaman Ensafi, Saman Hassanzadeh Amin, Guoqing Zhang, and Bharat Shah. Time-series forecasting of seasonal items sales using machine learning - a comparative analysis. *Int. J. Inf. Manag. Data Insights*, 2:100058, 2022.

[2] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72:37 – 45, 2018.

[3] Adriana Horelu, Catalin Adrian Leordeanu, Elena Simona Apostol, Dan Huru, Mariana Mocanu, and Valentin Cristea. Forecasting techniques for time series from sensor data. *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 261–264, 2015.

[4] Amer Malki, Elsayed Atlam, and Ibrahim Gad. Machine learning approach of detecting anomalies and forecasting time-series of iot devices. *Alexandria Engineering Journal*, 2022.

[5] Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. Stock closing price prediction using machine learning techniques. *Procedia Computer Science*, 167:599–606, 2020.

[6] Felix Alexander Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In *International Conference on Artificial Neural Networks*, 2000.

[7] Junhang Chen, Wang Koju, Shurui Xu, and Zhe Liu. Sales forecasting using deep neural network and shap techniques. *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pages 135–138, 2021.

[8] Kasun Bandara, Peibei Shi, C. Bergmeir, Hansika Hewamalage, Quoc-Huy Tran, and Brian Seaman. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. *ArXiv*, abs/1901.04028, 2019.

[9] Guoqiang Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[10] Nitin Merh, Vinod Prakash Saxena, and Kamal Raj Pardasani. A comparison between hybrid approaches of ann and arima for indian stock trend forecasting. 2010.

[11] Bineet Kumar Jha and Shilpa Mangesh Pande. Time series forecasting model for supermarket sales using fb-prophet. *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 547–554, 2021.

[12] Mohammed Ali Alshara. Stock forecasting using prophet vs. lstm model applying time-series prediction. 2022.

[13] Xue bo Jin, Wenlong Gong, Jianlei Kong, Yu ting Bai, and Tingli Su. A variational bayesian deep network with data self-screening layer for massive time-series data forecasting. *Entropy*, 24, 2022.

[14] Xueqi Zhang, Meng Zhao, and Rencai Dong. Time-series prediction of environmental noise for urban iot based on long short-term memory recurrent neural network. *Applied Sciences*, 2020.

[15] Iliana Paliari, Aikaterini Karanikola, and Sotiris Kotsiantis. A comparison of the optimized lstm, xgboost and arima in time series forecasting. In *2021 12th International Conference on Information, Intelligence, Systems Applications (IISA)*, pages 1–7, 2021.