

# Cloud based Smart Waste Management System Using Internet of Things (IoT) and Predictive Analysis.

MSc Research Project  
Cloud Computing

Prakash Shankar Mishra

Student ID: 21172684

School of Computing  
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Prakash Shankar Mishra
<b>Student ID:</b>	21172684
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2023
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Sean Heeney
<b>Submission Due Date:</b>	14/08/2023
<b>Project Title:</b>	Cloud based Smart Waste Management System Using Internet of Things (IoT) and Predictive Analysis.
<b>Word Count:</b>	7850
<b>Page Count:</b>	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	Prakash Shankar Mishra
<b>Date:</b>	14th August 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Cloud based Smart Waste Management System Using Internet of Things (IoT) and Predictive Analysis.

Prakash Shankar Mishra  
21172684

## Abstract

One of the key elements that determines the health of an urban or rural area is waste management. For the authorities, keeping the environment neat and orderly in terms of waste management becomes a difficult effort. Intelligent monitoring of solid waste dust bins is used right away to address this problem and create a safe and secure environment. I have suggested a smart waste management system in this paper. The proposed smart bin model forecasts the status of the waste bins using the cloud, IoT, and machine learning. In this study, five machine learning time series models have been used, and a comparison analysis has been conducted. The best outcomes were obtained using the generalised additive model, which had an MAE 0.2407 and MSE 1.9399 which shows the forecasting values have fewer deviations when compared to the actual values. In this research, I was able to Forecasting the waste bin fill level with a significant accuracy.

## 1 Introduction

The UN Sustainable Development Goal 11, "Make cities and human settlements inclusive, safe, and sustainable," identifies waste management as a crucial factor in the creation of sustainable cities as target 11.6.<sup>1</sup> The population of the world is projected to grow by 20% by 2025, peaking at 8 billion people, according to estimates from the UN. Due to the accelerated population expansion, there will be a corresponding increase in consumer demand, which will increase trash generation. Considering the present urban and specially third world countries waste management practises, the technology in place to handle this significant increase in waste are insufficient. The fast urbanisation of the growing world population, economic growth, and increased consumption of food and consumer goods are the main causes of the world's annual production of more than 2 billion tonnes of municipal solid waste (MSW). According to estimates, over 30% of all food and its packaging produced globally end up in landfills and municipal trash bins as a result of inadequate recycling and disjointed MSW operations. As per Leninpugalhanthi et al. (2021), by 2050, it is anticipated that there will be an astounding 3.4 billion metric tonnes of municipal waste generated worldwide, a 70% increase. In major institutions or a city administered by a municipal corporation, where many trash cans are set up and staff are recruited specifically for this duty, the conventional approach of physically checking for filled garbage bins does not fit with the technological era we are in and is wasteful. The regularly planned cleaning of the trash cans is rendered useless because a

---

<sup>1</sup><https://www.un.org/sustainabledevelopment/cities/>

trash can suddenly fill up or become tampered with or because frequent inspections may not be necessary for a long time. trash cans are kept at a number of sites by the city's administration. They are accountable for routinely checking and removing the garbage that is stored in the dustbins. The garbage can could not have enough trash in it, thus they frequently arrive late or empty-handed. There can be a chance that the rubbish will deteriorate if they are late. It would lead to the growth of germs and viruses. As a result of the waste that has been gathered, the resulting air pollution will cause respiratory illnesses like COPD, asthma, etc. According to estimates, 90% of instances of chronic obstructive pulmonary disease are brought on by the unpleasant smell that trash produces.

Using the Internet of Things (IoT) and machine learning, we can automate this procedure to save time and increase efficiency. The best approach to do this would be to build smart trash cans and place them in public areas where a lot of trash is produced. When the trash cans are filled enough for authorities to check and collect the trash, these smart bins will be able to predict and forecast when that will happen. By minimising the number of unnecessary trips the garbage truck makes to each trash bin, the overall waste management operation would become much more cost-effective and efficient. In this research paper, I have implemented and performed Comparative Analysis on various machine learning models for predicting when the waste bin will be filled using IoT sensor data retrieved from a smart bin.

## **1.1 Research Question**

How to improve waste management system with the use of IoT, Cloud, Machine Learning and Predictive Analysis?

## **1.2 Document Structure**

This research report is divided into several sections. The review of prior research on machine learning approaches, smart waste management systems, and their limitations is provided in Section 3. The report explores deeply into the technique used in Section 3, detailing the implementation of data processing of the data set. We have critically analysed the models we utilised in this research in section 5.

The method of implementation is thoroughly covered in Section 6. Section 7 discusses the results of the developed models and includes the methodology used to assess each model's accuracy and performance.

The paper's conclusion, Section 8, provides a thorough assessment of the study's findings and the conclusions taken from the model results.

## **2 Related Work**

This section will critically evaluate and discuss the significant previous research on smart waste management system implementation using cloud, machine learning and IoT. Understanding these issues and gaps in the earlier research efforts and how our method will fix them is helpful.

## 2.1 Cloud Computing Implementation

In the proposed smart waste bin named Recycle.io, Al-Masri et al. (2018) combined cloud technologies and an IOT enabled framework. IOT devices are used in each of these bins in the smart trash management system Recycle.io. For this purpose, Recycle.io attaches a Raspberry Pi to each of these containers. Each cutting-edge device is equipped with an ultrasonic sensor and an infrared camera. An ultrasonic sensor was used to find the location of waste dumping. The ultrasonic sensor informs the camera module to begin capturing pictures when a disposal is discovered. Following that, the edge device (in this case, a Raspberry Pi) processes these images locally for violation detection. In the event of a violation, a snapshot of the image is sent to our IoT-based cloud platform recycle.io. When a violation is discovered, the IoT edge device sends a photo of the infraction and additional information to the Microsoft Azure IoT platform. This information includes, among other things, the component that was the cause of the infringement, the time stamp, and the location of the bin. The data obtained by the camera module and sensors in the smart bins is sent to an analytic unit. In order to seek for any infractions, the analytic unit processes and evaluates the captured images of disposed materials. Although it is a scalable system, there is no emphasis on the effective collection of the waste detected in the bins, therefore it will require labour to examine the data given by the smart bins. My proposed system will predict the waste level in the bin by forecasting it in advance which will help in planning an effective collection of the waste in timely manner.

Aazam et al. (2016) has proposed the concept of a Cloud-based Smart Waste Management (CloudSWAM) system, where each bin is outfitted with sensors that can determine how much waste is there within. Each form of waste, including biological waste, objects made of plastic, and scrap metal, will have its own bin. This division of each sort of garbage allows for knowledge of the quantity and type of garbage that is gathered. A warning message will be sent to the cloud if the volume of trash reaches the threshold where garbage haulers need to schedule pickup. This informs users and waste collectors as to which bins still have space for trash. As a result, garbage collectors are able to schedule their trips according to the amount of rubbish present in distinct metro areas. This study article does not make any concrete recommendations for future work in any one topic. From there, we will examine prior waste data collected from bins, and will forecast the prediction of waste level in each bin, making our system more efficient

## 2.2 Machine Learning Implementation

For the recognition of garbage images and content, N. et al. (2019) used the machine learning algorithms KNN and SURF. In past studies, wastes were simply observed and notifications were provided, however, with this system, wastes are recorded and additionally classified into metallic, biodegradable, and non-biodegradable wastes. To make it easy to dispose of each type of garbage, the trash is categorised into categories. A cloud database is also used to transmit the waste facts in real-time.

Likotiko et al. (2021) use all the features mentioned in the aforementioned researches and use various machine learning algorithms to predict the waste level in the bin, but it is based on a small residential area data set. The previous researches proposed solutions for real-time waste monitoring, informing the concerned authorities about the bin status and segregation among the wastes. In our proposed research, we will implement it on a larger level by predicting the waste bins behavior in advance and forecast it

Bharathiraja et al. (2022) developed an efficient trash bin management framework in their paper to constantly monitor solid waste. IoT is the foundation of the architecture, with the goal of improvising waste management systems. Their suggested methodology used the SARIMAX model for forecasting and predictive modelling and communicated through cloud implementation. In my research, I have implemented five different time series machine learning models to for forecasting and predictive analysis and after evaluation through different methods the best models are proposed.

### 3 Methodology

After carefully reviewing the study conducted by the researchers in the preceding section, the methodology and analysis offered in this section were selected. The methodology is divided into four steps: data gathering, pre-processing, transformation, analysis, and model selection. Data collection is the first step. Data from IoT sensors is gathered and given a thorough analysis like mentioned in Medehal et al. (2020). After the data has been thoroughly analysed, it is pre-processed. Next, we analyse how long it takes to fill the dustbin before choosing the model. Different models are applied, evaluated using the proper metrics, and after deploying different machine learning models, we performed comparative Analysis on the basis of matrices and result.

#### 3.1 Data Collection:

The COMPOSITION EU Project served as the source of the sensor data data set. The sensor readings from the deployed bin fill level bins are included in the deployed data sets. The public can access these data sets. Data has been made available on the internet with the declaration of public access and use.<sup>2</sup> Monitoring of indoor bin fill levels is the subject of the first data set. A bin's fill level being monitored outside is the subject of the second data set. In doing our research, we utilised the second data set.

#### 3.2 Data Understanding

The data set contains 16,801 rows, and 5 columns. The column 'id' display the id of the sensor deployed on the dustbin for the reading, the column 'Fill Percentage' is showing how much percentage the dustbin is filled from bottom, the column 'battery' shows the battery level of the sensor deployed. The column 'eventDate' is the timestamp of when the sensor took the reading of the bin and 'Distance' is the distance between the sensor which is placed on the lid of the bin and the waste.

The fill percentage is being calculated by comparing the distance between the sensor which is mounted on the top of the dustbin and the waste level to the full distance of the empty bin. Hence, the distance and the fill percentage columns are inversely proportional to each other.

#### 3.3 Data Preparation

For predictive analysis, the timestamp data is very crucial. We converted the eventDate column data to date time format, by this the data is made more suitable for time series

---

<sup>2</sup><https://zenodo.org/record/3375560>

machine learning algorithms. This transformation makes it simple to easily compute the time gaps between various events and sorting events in chronological order. We re sample the data at 5-minute intervals and identify any missing timestamps and we found 133 missing values.

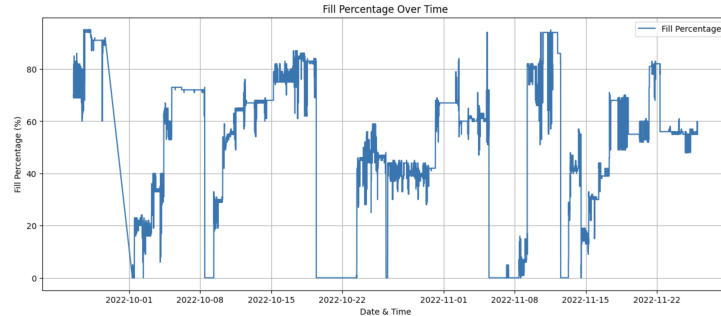


Figure 1: Fill Percentage Over Time

Our data was divided into training and test sets; the test set would span 5 consecutive days. To evaluate the model’s performance over a specified time period, this duration was selected. To distinguish between our training and test data, we must choose a specific time interval. For the test period to cover the specified 5 days, this timestamp must be exactly 5 days before the end of our data set. Once we have the timestamp we wanted, we split our data set in half. Data that came before the established date makes up the first component, called the training set. The test set, which comprises the second component, spans the five days after the timestamp. We used a method known as interpolation to preserve the integrity of our data set. By guessing missing values based on the data already collected, a continuous and coherent data set is created for analysis. Our training set has 14,477 data points after the split and filling in the missing values, whereas the test set has 1,430 data points. These sizes represent the variety of cases accessible for model testing and training. We extract more characteristics with a time component from the timestamp data. The weekday, hour of the day, month, and other elements are among these properties. These temporal aspects provide perceptions of temporal patterns that can improve the precision of our analysis and modelling.

### 3.4 Data Pre-processing

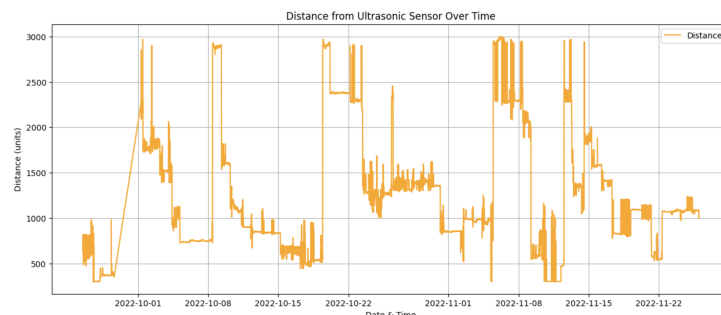


Figure 2: Distance from Ultrasonic Sensor Over Time

Our data is split into two primary categories as we move forward. Every attribute that we utilise to create predictions are included in one section, which is referred to as the input features ( $X$ ). The value we're attempting to forecast or analyse is represented by the second component, which is referred to as the target variable ( $y$ ). Both the training and test data sets are separated in this way.

### 3.5 Model Evaluation

In order to train and evaluate machine learning models, we have incorporated a number of them at this stage, including SARIMA, XGBoost, Prophet, Holt-Winters Exponential Smoothing, and Generalized Additive Model. In the stage of model evaluation, the models' performance was then assessed. Different graphs have been generated for the models provided in this study in order to do a comparative analysis.

## 4 Design Specification

The architecture, model deployment, and general functionality of the research project, which leverages Internet of Things (IoT) data and Predictive Analysis, are all significantly shaped by Amazon SageMaker. Amazon SageMaker is a crucial element that makes it easier to create and use machine learning models. SageMaker's features for data preparation, model training, and hosting are used by the cloud-based architecture. The system takes advantage of SageMaker's modular and scalable design to ensure flexibility in response to shifting needs and data volume.

I have used AWS SageMaker to develop and model the machine learning models. This project uses five machine learning algorithms in total for predictive analysis of waste level data. All of the models use the time series learning technique, which is widely used to analyse and forecast future values based on historical data collected over time from observations of a variable.

The machine learning models I have used in my research's are:

- SARIMA (Seasonal Auto Regressive Integrated Moving Average)
- XGBoost
- Prophet
- Holt-Winters Exponential Smoothing
- Generalized Additive Model

### 4.1 SARIMA (Seasonal Auto Regressive Integrated Moving Average)

The powerful time series forecasting technique called the Seasonal Auto Regressive Integrated Moving Average (SARIMA) model was created to recognise and forecast patterns in data that exhibit both trend and seasonality. SARIMA expands the capabilities of the Auto Regressive Integrated Moving Average (ARIMA) model to take into account the intricacies of data that is depending on the season.



SARIMA combines a number of essential elements to acquire its forecasting prowess. Historical trends can be incorporated due to the Auto Regressive (AR) component, which predicts the impact of earlier observations on the current value. The Integrated (I) component manages differencing, which entails deducting earlier values from the present value to ensure stationarity, where the statistical characteristics of the data remain constant throughout time. The Moving Average (MA) component, which aids in identifying random shifts and patterns, takes into account the impact of previous forecasting errors on the present number.

Permanasari et al. (2013) have discussed that when a time series shows a seasonal variation, seasonal ARIMA (SARIMA) is used. The multiplicative process of SARIMA will be written as  $(p, d, q)(P, D, Q)_m$ , where  $(P, Q)$  is a seasonal moving average notation and  $(P)$  is a seasonal auto regressive notation. The length of the seasonal period is indicated by the sub scripted letter.

The hyper parameters of the SARIMA model are:

- $(p)$ : Trend auto regression order.
- $(d)$ : Trend difference order.
- $(q)$ : Trend moving average order.
- $(P)$ : Seasonal auto regressive order.
- $(D)$ : Seasonal difference order.
- $(Q)$ : Seasonal moving average order.
- $(m)$ : The number of time steps for a single seasonal period.

Given that we have daily seasonality with 5-minute intervals,  $(m)$  will be 288 (as there are 288 intervals in a day).

SARIMA is particularly good for forecasting time series data with complicated patterns because it can incorporate temporal and seasonal components. Despite its benefits, SARIMA has drawbacks, such as the requirement to choose the right placements for each of its parts, which frequently necessitates testing and validation. Additionally, with larger datasets or sophisticated configurations, the computational complexity of the model may rise.

## 4.2 XGBoost

XGBoost is a powerful gradient boosting algorithm that can be used for time series forecasting. It's well-known for its efficiency, flexibility, and strong predictive performance.

Since XGBoost is a tree-based algorithm, it doesn't inherently capture time dependencies. I need to create lagged features that provide information about previous time steps. Due to its regularisation strategies, parallel processing, and gradient boosting ideas, XGBoost performs exceptionally well. While parallel processing speeds up tree construction for scalability, regularisation reduces over fitting. It is adept at handling problems involving classification and regression involving a variety of data formats.

The capacity of XGBoost to handle imbalanced data sets and provide precise predictions for unusual events is one of its prominent characteristics. Additionally, it aids interpret ability by providing insights into the significance of the features.

Ndayishimiyepas et al. (2022) used Xgboost to compute the total of the predictions from each tree as the final prediction. Traditional Euclidean-space optimisation techniques cannot be applied to the tree ensemble model in Figure 4 since it has functions as parameters. Instead, the model undergoes additive training.

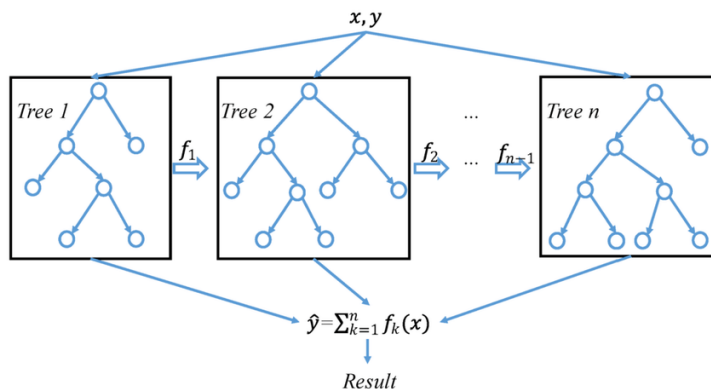


Figure 3: XGBoost Architecture

### 4.3 Prophet

The Prophet model is a robust forecasting tool developed by Facebook that is designed to handle missing data, outliers, and multiple seasonalities. It is based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. Prophet uses an additive model that takes trend, seasonality, and holiday effects into account. Because it can detect sudden changes and outliers in the data, it is resistant to anomalies. Additionally, it enables users to include unique seasonality and events that can have an impact on the data. Prophet stands out for its intuitive

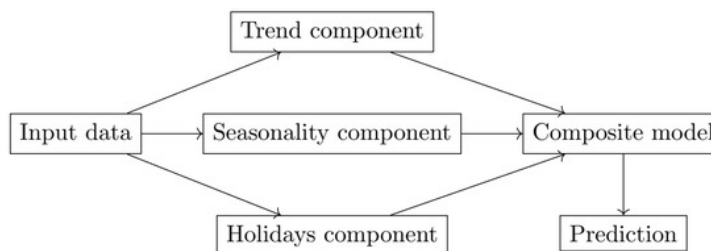


Figure 4: Prophet Architecture

user interface and automatic handling of numerous modelling decisions, such as outlier detection and handling missing data. Because of this, it may be used by people with different degrees of competence, from novices to seasoned data scientists. Prophet offers simplicity, but it may not be as adaptable as other advanced time series models. Prophet, on the other hand, offers a useful option that requires little technical know-how and parameter modification for situations where quick and precise forecasts are necessary. Stefenon et al. (2023) stated that it's a useful tool for companies and researchers looking for effective time series forecasting without becoming entangled down in the details of complex modelling methodologies.

## 4.4 Holt-Winters Exponential Smoothing

Holt-Winters Exponential smoothing is an advanced time series forecasting technique created to identify and forecast patterns in data that exhibit trend, seasonality, and level components. This approach, which includes the names of its creators Charles Holt, Peter Winters, and Frank Brown, works especially well for data sets that exhibit recurring patterns that are persistent over time.

Holt-Winters Exponential Smoothing, at its core, smoothes data by emphasising recent data points while compensating for historical values by distributing exponentially decreasing weights to previous observations. The three primary parts are as follows:

- **Level Component:** The level component depicts the time series' underlying baseline and captures its overall trend through time. It takes into account how the series has evolved during recent observations.
- **Trend component:** The trend component depicts the direction and rate of the time series' long-term changes. It records the data regular upward or downward movement.
- **Seasonal Component:** The seasonal component takes into account recurring patterns that happen at regular intervals, such as daily, weekly, or monthly cycles. It captures recurring fluctuations brought on by a variety of outside causes, including seasons, holidays, and economic cycles.

Singh et al. (2020) discussed the two primary variants of Holt-Winters Exponential Smoothing: additive and multiplicative. When the seasonal fluctuations are constant, regardless of the level of the time series, the additive technique is appropriate. When the level of the time series changes along with the amount of seasonal variations, the multiplicative technique should be used.

## 4.5 Generalized Additive Model

By enhancing Generalised Linear Models' (GLMs') capabilities, Generalised Additive Models (GAMs) constitute an important development in the field of statistical modelling. Despite being quite good at modelling relationships between predictors and response variables, GLMs are constrained by the linearity assumption. Li et al. (2019) mentioned that GAMs come into play to overcome this restriction, allowing for the flexible modelling of nonlinear connections.

GAMs introduce the idea of smooth functions for predictors while retaining the fundamental concepts of GLMs. Consequently, GAMs can handle a variety of response types, including binary, count, and continuous variables, just like GLMs can. GAMs permit these interactions to take on more complex shapes as opposed to assuming a linear relationship between the predictors and the outcome.

Another advantage of GAMs is their interoperability. It is possible to investigate the structure of the smooth functions to learn more about how predictors and responses interact. This makes it easier to get important insights and comprehend the variables influencing the result.

## 5 Implementation

### 5.1 Configuring AWS SageMaker

AWS SageMaker modelling configuration includes a number of significant stages. I launched the AWS Management Console and went to SageMaker first. When building up the notebook environment, I used PyTorch 1.10 Python 3.8 CPU Optimised image, which is included in deep learning containers, along with python3 Kernel. I then built a notebook instance of 48 vCPU + 96 GIB and ml.c5.12xlarge instance type. The Jupyter Notebook interface is then opened, and I upload and process the data set there. I loaded datasets through the Jupyter interface and used code cells to install the required libraries.

### 5.2 Dustbin's Fill Cycle Analysis

Through an in-depth analysis of the time it takes for the dustbin to fill to 90% capacity, we identified distinct fill cycles by considering readings below 20% as the starting point for each cycle. This approach ensures that we capture the time required to fill the dustbin from a nearly empty state to 90% full.

The analysis revealed the following insights:

- Average Time to Fill to 90% Approximately 5 days, 13 hours, 22 minutes
- Minimum Time to Fill to 90% Approximately 1 day, 8 hours, 3 minutes
- Maximum Time to Fill to 90% Approximately 9 days, 18 hours, 41 minutes

Given this information, we can define the test set length as 5 days, representing a typical fill cycle. This choice balances the need to capture the inherent variability in fill times with the desire to have a test set that is representative of common scenarios.

By using a 5-day test set, we ensure that the model evaluation is grounded in the actual operational dynamics of the waste collection system, providing more robust and meaningful performance metrics.

### 5.3 Conclusion of Data Exploration & Pre processing

#### 5.3.1 Data set Structure

This data set contains an abundance of information that was methodically obtained using a sensor that was placed atop a garbage can. The "Distance" data is carefully sourced from an ultrasonic sensor that measures the distance between the sensor and the contents of the bin. This sensor expertly captures key metrics, including the "Fill Percentage," which denotes the extent of the bin's occupancy, the "Battery Level," which is essential for maintaining uninterrupted sensor functionality. These readings provide a thorough insight of the condition of the garbage can as a whole, laying a vital foundation for wise resource allocation, waste management, and operational optimisation decisions. Rijah and Abeygunawardhana (2023)

#### 5.3.2 Pre processing

The timestamps were rounded to the nearest 5-minute interval, and irrelevant columns were removed. We treated missing values with interpolation.

### 5.3.3 Summary Statistics

- Fill Percentage: Varied between 0% and 95% with an average fill level of 49.32%:
- Battery: Mostly stable around the value of 56
- Distance: Ranged from 300 to 3001 units, with an average distance of 1255.91 units.

### 5.3.4 Visualizations

- Fill Percentage: The figure 2 showed fluctuations and patterns, reflecting the usage and emptying cycles of the dustbin.
- Distance: The distance from the sensor varied inversely with the fill percentage, decreasing as the bin filled as shown in figure 3.

## 5.4 Seasonal Decomposition & Time Series Analysis

Damrongkulkamjorn and Churueang (2005) discussed that seasonal decomposition and time series analysis are important resources for examining and comprehending temporal data trends. Time series data are observations that are collected over time periods of days, months, or years. These data frequently display a variety of patterns, including trends, seasonal fluctuations, and abnormalities.

The process of seasonal decomposition entails dividing a time series into its three basic elements, trend, seasonality, and residuals. The trend is an indicator of the data's underlying long-term movement, which may be increasing, decreasing, or steady over time. Seasonality refers to repeated patterns that happen on a regular basis, such as daily, weekly, or annual cycles. After trend and seasonality are taken into consideration, residuals include the unexplained variability and arbitrary fluctuations that are still there.

The seasonal decomposition plot provides insights into different components of the "Fill Percentage" time series:

### 5.4.1 Observed

percentage over the whole temporal spectrum. This unaltered analysis depicts the variations in fill levels as they change over time, capturing the true essence of the temporal dynamics of the data. The flow of fill percentages are clearly shown through this novel time series, providing the framework for additional analysis, trend identification, and predictive modelling.

### 5.4.2 Trend

The underlying and persistent pattern found in the data is what is meant by the term "trend." It acts as a conduit through which the subtle changes in the data over a long period of time are visible. The trend provides a panoramic orientation by encompassing the broad trends that endure across time and providing a clear sense of the data's main trajectory.

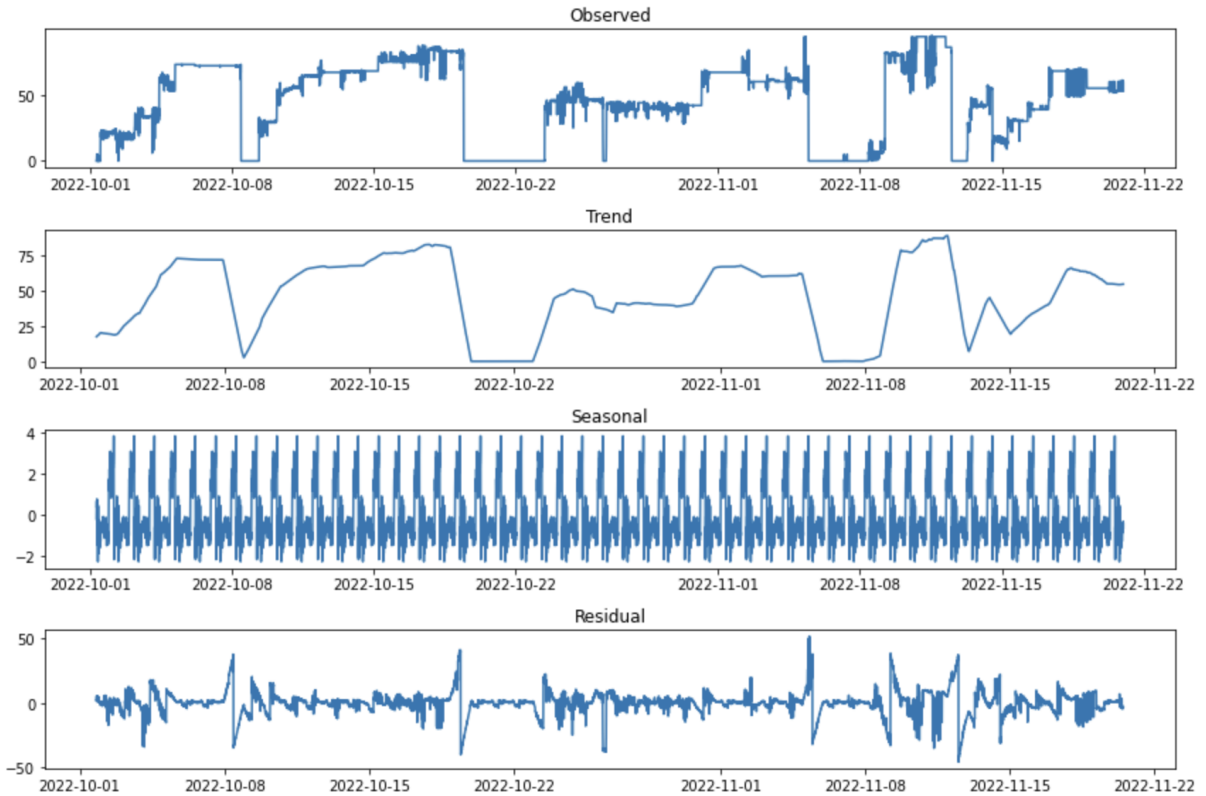


Figure 5: Seasonal Decomposition Plot

### 5.4.3 Seasonal

Seasonal: The term "seasonal" refers to reoccurring patterns that appear at regular periods, in this case repeating everyday. The graphical display highlights the fill percentage's cyclical changes and reveals a distinct daily pattern in the data. This finding emphasises the data's propensity to display regular patterns that correspond to the passing of each day, providing insight into the systematic oscillations that occur with a predictable rhythm.

### 5.4.4 Residual

The term "residual element" refers to the remaining noise or random oscillations in the data after the trend and seasonal components have been methodically removed. It basically captures the unexplained variance that is left behind after taking into consideration the structured patterns.

The decomposition confirms the presence of daily seasonality in the data and provides a clearer view of the trend and residual components.

## 5.5 Model Selection & Training

### 5.5.1 SARIMA

SARIMA is an extension of ARIMA that explicitly supports uni variate time series data with a seasonal component. It adds three new hyper parameters to specify the auto

regression (AR), difference (I), and moving average (MA) for the seasonal component of the series. Once the required libraries have been loaded, the IoT sensor data is first placed into a data set by parsing the Date column and converting it to a Date Time format in Jupyter Notebook in AWS SageMaker. Now, the model only accepts the Date and fill percentage columns as input. The primary libraries used to generate ARIMA and SARIMA models are Pandas, NumPy, Matplotlib, Sklearn, and statsmodel.tsa.

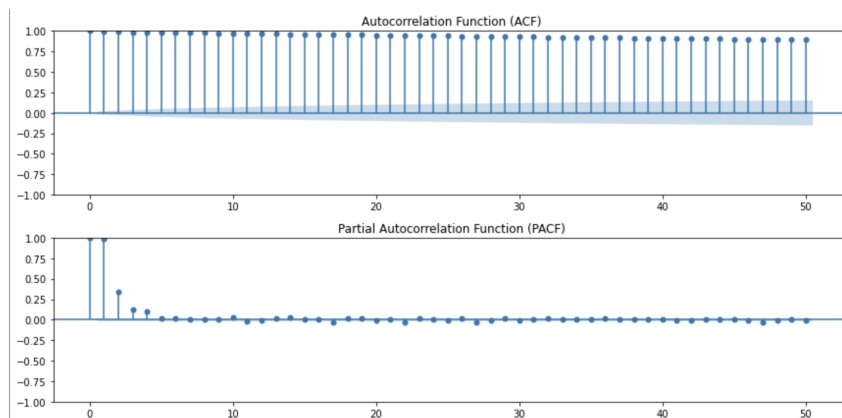


Figure 6: ACF and PACF Plots

### 5.5.2 Hyper Parameter Analysis

- Correlogram Function (CF) and Partial Auto correlation Function (PACF) Plots: The use of CF and PACF plots is essential for revealing the complex auto correlation structure ingrained in the data. We learn a lot about the possible ordering of the seasonal auto regressive (P) and moving average (Q) components by visualising these charts. These graphical representations make it easier to spot lags that have noticeable auto correlation, which helps us choose the best parameters for our investigation.
- Application of the Augmented Dickey-Fuller (ADF) test on seasonally differentiating data emerges as a critical determinant as we begin the investigation of seasonal patterns. This statistical test determines whether unit roots exist, which would suggest non-stationarity. We determine the order of seasonal differencing (D) necessary to establish stationarity by putting the seasonally differentiated data to this test. This crucial stage directs our comprehension of the stability of the data, permitting knowledgeable choices regarding the temporal properties and the suitable metrics to get significant insights from the time series data.

Seasonal Lags: We can observe spikes at regular intervals, suggesting the presence of daily seasonality. P and Q Values: Based on the PACF and ACF plots, we may consider trying different values for the seasonal auto regressive (P) and moving average (Q) terms during the model fitting process.

The ADF test is used to check for stationarity in a time series. Here are the results:  
ADF Statistic: -5.56

Given the low p-value, we can reject the null hypothesis that the series has a unit root, and we can conclude that the seasonally differenced series is stationary.

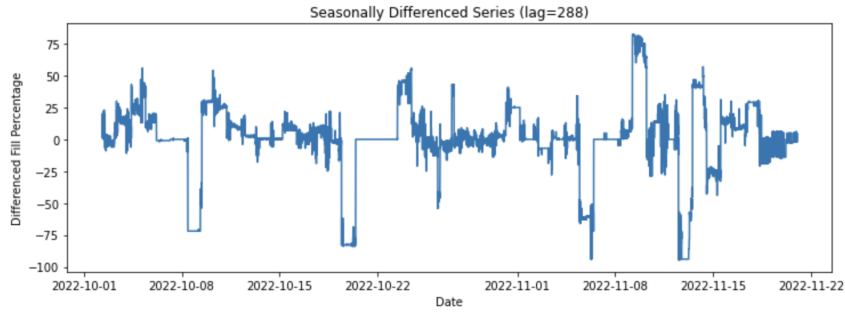


Figure 7: Seasonally differenced series

SARIMA Model works better since the data follows a seasonal trend. The SARIMA model is fitted using similar procedures. It was also given to check the auto correlation function (ACF) and partial auto correlation function (PACF) plots in order to ascertain the values of  $p$  and  $q$ . The SARIMA model is then trained on the train data using the variables  $p$ ,  $d$ , and  $q$ . The data has now been divided into train and test groups. Order by Season (3, 2, 1) The `fit()` method is then used to fit the model. Once the model has been trained, the values for the test data are forecast using the function `forecast()`. Both predicting values and data frame values are examined using the evaluation metrics MAPE, and MAE.

### 5.5.3 XGBoost

Extreme gradient boosting (XG Boost) is one of the best ways to apply the gradient boosting machine learning approach. XGBoost is created for classification and regression problems in addition to time series forecasting. The initial step involves setting up the XGBoost library. The cleaned IoT sensor data set is now used as input for the XG Boost model. In the features engineering section, we created lagged features which is a common approach for time series forecasting is to create lagged features, which are values of the target variable (in this case, fill percentage) at previous time steps. By using these lagged values as features, we allow the model to learn from past observations to make future predictions. After that we did model preparation and model training performed evaluation on the results.

### 5.5.4 Prophet

The Prophet model is a robust forecasting tool developed by Facebook that is designed to handle missing data, outliers, and multiple seasonalities. It is based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. In data Preparation stage the column `eventDate` was changed to "ds" and the column `Fill Percentage` was updated to "y" in accordance with the prophet model's specifications. Prophet expects the input data to have two columns: `ds` for the timestamps and `y` for the values we want to forecast. In the model initialization stage we configured the Prophet model to include daily seasonality with a frequency of 288 intervals, which corresponds to the 5-minute data intervals we have in our data set. This accounts for the repetitive patterns that occur within a day. We also Tuned different hyper parameters. The `[.fit()]` function is used to fit the prophet model to training data,



and prediction values are then forecasted. To compare, MSE and MAE measurements are used.

### 5.5.5 Holt-Winters Exponential Smoothing

The Holt-Winters Exponential Smoothing model is a powerful time series forecasting technique that generalizes and extends the exponential smoothing method. It is well-suited for data with trends and seasonal patterns. While initializing the model, we imported libraries like Exponential Smoothing and we initialized the Holt-Winters Exponential Smoothing model with the specified parameters like trend Component which specifies the type of trend component ('additive' or 'multiplicative'), seasonal component which specifies the type of seasonal component ('additive' or 'multiplicative') and seasonal periods which Specifies the number of observations per seasonal cycle (e.g., 288 for daily seasonality with 5-minute intervals). Then we fitted the model into the training data and initialized the forecasting. Once the forecasting is successfully done, we calculated the MAE and MSE and plotted the graphs.

### 5.5.6 Generalized Additive Model

GAMs are an extension of Generalized Linear Models (GLMs), allowing for nonlinear relationships between the predictors and the response variable. First we created a copy of the training data for feature engineering and then we created N lagged features, after that we dropped the null values and separated the features other than Fill Percentage as X and target as Fill Percentage. Then we imported LinearGAM library and define the GAM model with smoothing terms for each feature. We proceeded to make predictions on the test set and calculate the MAE and MSE and plotted the Actual vs Predicted Fill Percentage graphs.

## 6 Evaluation

All of the models that have been deployed will be thoroughly examined in this section. We will thoroughly evaluate each model's performance to determine its efficacy and forecasting ability.

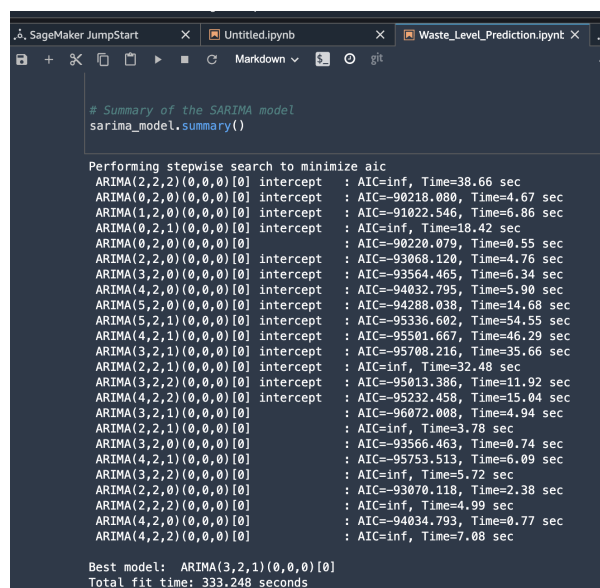
Mean Absolute Error (MAE) and Mean Squared Error (MSE), two well-known performance metrics, will be used to assess the precision and competence of these models in predicting the amount of waste that will be present in the bins.

- The Mean Absolute Error (MAE): It measures the average magnitude of errors between actual and predicted values, is a significant benchmark. It captures the absolute difference between these numbers and provides insight into the average accuracy with which the models estimate waste quantities. A lower MAE indicates that the model's predictions are closer to the actual values, implying higher accuracy.
- Mean Squared Error (MSE): By averaging the squared differences between the expected and actual values, MSE determines prediction accuracy. Bigger errors are highlighted, assisting in the evaluation of model performance across different sectors.

## 6.1 SARIMA (Seasonal AutoRegressive Integrated Moving Average)

I plotted ACF and PACF graphs (Figure 6) which helped me understand the auto correlation structure of the data and provide insights into the seasonal auto regressive (P) and moving average (Q) orders, we can observe spikes at regular intervals, suggesting the presence of daily seasonality.

I performed the Augmented Dickey-Fuller (ADF) test shown in Figure 7 as well on the data to determine the order of seasonal difference (D) and the p value came as  $1.5287170699873637e-06$ . Given the low p-value, we can reject the null hypothesis that the series has a unit root, and it can be concluded that the seasonally differences series is stationary. In order to build the SARIMA model, we have utilised auto arima to discover



```
# Summary of the SARIMA model
sarima_model.summary()

Performing stepwise search to minimize aic
ARIMA(2,2,2)(0,0,0)[0] intercept : AIC=inf, Time=38.66 sec
ARIMA(0,2,0)(0,0,0)[0] intercept : AIC=-98218.080, Time=4.67 sec
ARIMA(1,2,0)(0,0,0)[0] intercept : AIC=-91022.546, Time=6.86 sec
ARIMA(0,2,1)(0,0,0)[0] intercept : AIC=inf, Times=18.42 sec
ARIMA(0,2,0)(0,0,0)[0] intercept : AIC=-98220.079, Time=0.55 sec
ARIMA(2,2,0)(0,0,0)[0] intercept : AIC=-93068.120, Time=4.76 sec
ARIMA(3,2,0)(0,0,0)[0] intercept : AIC=-93564.465, Time=6.34 sec
ARIMA(4,2,0)(0,0,0)[0] intercept : AIC=-94032.795, Time=5.90 sec
ARIMA(5,2,0)(0,0,0)[0] intercept : AIC=-94288.038, Time=14.68 sec
ARIMA(5,2,1)(0,0,0)[0] intercept : AIC=-95336.602, Time=54.55 sec
ARIMA(4,2,1)(0,0,0)[0] intercept : AIC=-95501.667, Time=46.29 sec
ARIMA(3,2,1)(0,0,0)[0] intercept : AIC=-95708.216, Time=35.66 sec
ARIMA(2,2,1)(0,0,0)[0] intercept : AIC=inf, Time=32.48 sec
ARIMA(3,2,2)(0,0,0)[0] intercept : AIC=-95013.386, Time=11.92 sec
ARIMA(4,2,2)(0,0,0)[0] intercept : AIC=-95232.458, Time=15.04 sec
ARIMA(3,2,1)(0,0,0)[0] intercept : AIC=-96072.008, Time=4.94 sec
ARIMA(2,2,1)(0,0,0)[0] intercept : AIC=inf, Time=3.78 sec
ARIMA(3,2,0)(0,0,0)[0] intercept : AIC=-93566.463, Time=0.74 sec
ARIMA(4,2,1)(0,0,0)[0] intercept : AIC=-95753.513, Time=6.09 sec
ARIMA(3,2,2)(0,0,0)[0] intercept : AIC=inf, Time=5.72 sec
ARIMA(2,2,0)(0,0,0)[0] intercept : AIC=-93070.118, Time=2.38 sec
ARIMA(2,2,2)(0,0,0)[0] intercept : AIC=inf, Time=4.99 sec
ARIMA(4,2,0)(0,0,0)[0] intercept : AIC=-94034.793, Time=0.77 sec
ARIMA(4,2,2)(0,0,0)[0] intercept : AIC=inf, Time=7.08 sec

Best model: ARIMA(3,2,1)(0,0,0)[0]
Total fit time: 333.248 seconds
```

Figure 8: Auto Arima Function

parameter values (p,q,d,P,Q,D) by iterating through the loop. to enable the identification of the most suitable model. Figure 8 depicts the loop's result as of the moment. The best result is determined using the Akaike information criterion (AIC) value, a metric of a statistical model's relative quality for a certain collection of data. The AIC is a metric that measures how well a model fits the data while accounting for the complexity of the entire model. The model fits similarly with less features the smaller the AIC value, hence for this study the AIC value used is 960.51, which was obtained for ARIMA(3, 2, 1), meaning it includes 3 auto regressive (AR) terms, 2 differencing term, and 1 moving average (MA) terms. Following the application of the SARIMA model, the result was initially assessed using diagnostic plots, as seen in the Figure 9. A good model should have a normal distribution of residuals. It is expected that the KDE line will roughly resemble the mean line in the histogram display.

The linear trend of the samples in the qq plot is matched by the ordered distribution of residuals. The time series residuals have a poor correlation with its own logged form, as shown by the correlogram (i.e. auto correlation) diagram.

After implementing all above into execution, the model was eventually assessed using mean square error values. For this model, the Mean Absolute Error (MAE) and Mean

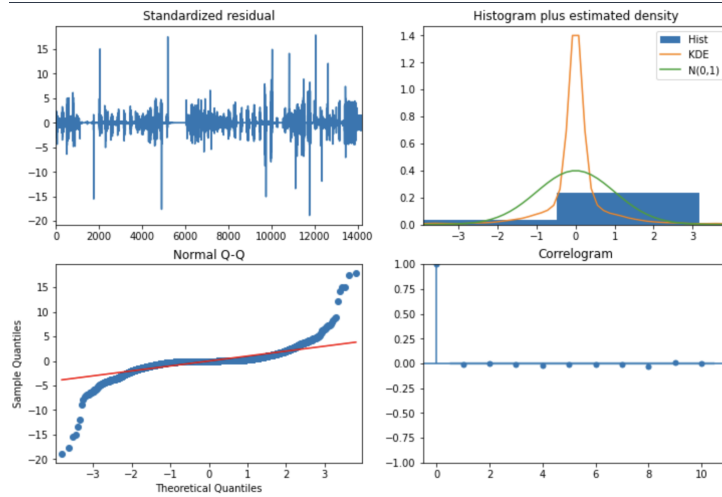


Figure 9: Model Diagnostics

Squared Error (MSE) are 6.939 and 147.450 respectively. In the graph, The discrepancies between the actual and forecasted values suggest potential opportunities for improvement. We might consider exploring different model specifications, feature engineering, or data transformations. Here are visible differences between the actual and forecasted values, especially in certain regions of the plot. Residuals and deviations contribute to the MAE and MSE values we observed.

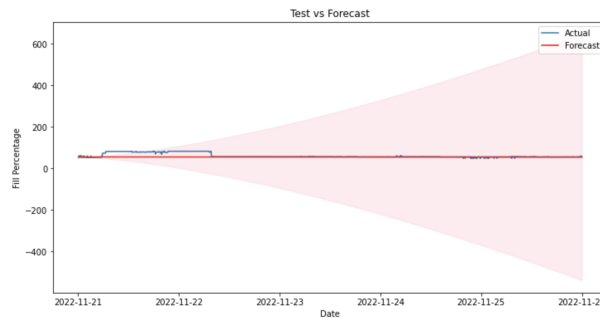


Figure 10: SARIMA Actual vs Forecast Graph

## 6.2 XGBoost

The initial step involves setting up the XGBoost library. The IoT sensor data set is now used as input for the XG Boost model. The training data set and testing data set must first be separated. A typical approach for time series forecasting uses lagged features, which are values of the target variable (in this case, fill percentage) at earlier time steps. I execute the model to use these lagged values as features to learn from past observations and produce predictions for the future. The target value and the selected features have now been used to partition the training data set. The model is then fitted using the training data set and the testing data set. The XGBoost model's performance on the test set was Mean Absolute Error (MAE): 1.467 and Mean Squared Error (MSE): 6.719 The XGBoost model has demonstrated reasonable predictive accuracy, both quantitatively

(MAE and MSE) and visually (plot). Figure 11 shows the actual vs. predicted fill percentage values for the test set.

In the actual vs forecast graph, we can see that its almost overlapping the actual data but around 22nd November to 23rd November, we can see the difference in the predicted data in compare to actual data. For that day the actual waste level was constant in the bin but XGBoost predicted some variation in that time period.

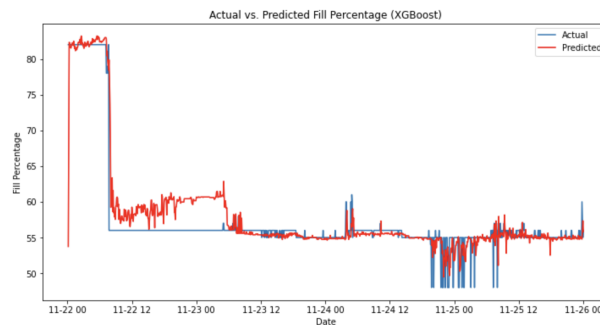


Figure 11: XGBoost Actual vs Forecast Graph

### 6.3 Prophet

To implement the time series forecasting using the Prophet model, I started by defining the parameter grid, including 'seasonality mode', 'daily seasonality', 'yearly seasonality', and 'weekly seasonality' options. Then I created a list comprising all potential combinations of these hyper parameters. After then I Initialized variables to track the best hyper parameters found during the process. Then it was proceeded to iterate through each combination, initializing a Prophet model with the selected hyper parameters. Fitting of this model to the training data was done for allowing it to learn patterns. Then, it utilized the trained model to make forecasts on the test data and provide the best hyper parameter, which was weekly seasonality and the Mean Absolute Error (MAE) and Mean Squared Error (MSE) are 12.564 and 283.669 respectively. The actual vs forecasted graph plotted by Prophet model is not accurate, as its going in a straight line between the days 21st November to 26th November while in same period we can see the fluctuations in the actual data.

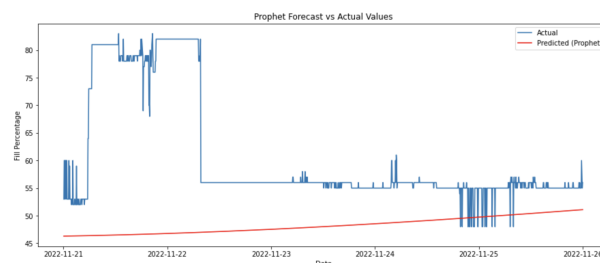


Figure 12: Prophet Actual vs Forecast Graph

Figure 12 shows the Prophet model's predicted fill percentage values for the test set.

## 6.4 Holt-Winters Exponential Smoothing

At the start I imported libraries and initialized the Holt-Winters Exponential Smoothing model with the specified parameters like `y train,trend=None, seasonal=None, seasonal periods=288`). Then I fitted the model to the training data and forecast the bin level. After that I modified the parameters and Re run the hyper parameter tuning with the updated options and through a loop iterating through the hyper parameters and fit the model. As a result we got a sorted table with hyper parameters and MAE.

The graph is not able to showcase the forecast and its predicting a constant level of waste inside the bin. This may have occurred because the data is being updated at every 5 minutes and the model is not able to show the overlapping over the actual data.

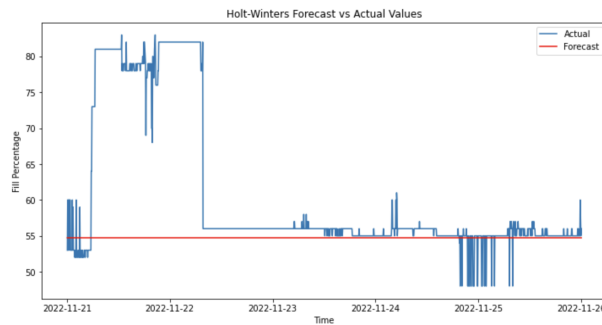


Figure 13: Holt-Winters Exponential Smoothing Actual vs Forecast Graph

Table 1: Holt-Winters model MAE relation with parameters

Trend	Seasonal	Seasonal Periods	MAE (\$)
None	None	288	6.399
None	add	288	8.354
add	None	288	9.0377
add	add	288	12.017

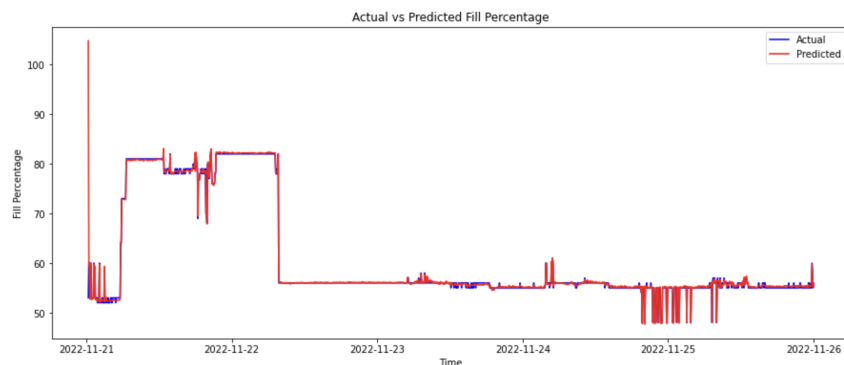


Figure 14: GAM Model Actual vs Forecast Graph

## 6.5 Generalized Additive Model

At the start I imported libraries and Created a copy of the training data for feature engineering. Then I created a lag of 3 that means the model will utilized past 3 result for the forecast. Then I separated features (X) and target (y) and made predictions on the test set. For this model, the Mean Absolute Error (MAE) and Mean Squared Error (MSE) are 0.2407 and 1.9399 respectively.

In the graph, it can be seen that the actual and predicted lines are nearly overlapping each other throughout, its show that Generalised Additive Model was able to predicted and forecast the dustbin waste level accurately.

## 6.6 Discussion

The Generalised Additive Model excelled among the variety of models evaluated on the waste bin data set. With favourable results for the MAE and MSE metrics, it proved to have higher predictive abilities. The Generalised Additive Model repeatedly demonstrated its effectiveness in this thorough assessment of all five models.

Table 2: Comparison of All Models

Model	MAE	MSE
SARIMA	6.939	147.450
XGBoost	1.467	6.719
Prophet	12.564	283.669
Holt-Winters Exponential Smoothing	6.399	143.970
Generalized Additive Model	0.2407	1.9399

The results of the XGBoost model were not as remarkable as those of the Generalised Additive Model, despite achieving a good MAE value. When compared to the other four models, the Prophet model performed considerably worse in terms of performance indicators, demonstrating a comparably poorer predictive performance.

## 7 Conclusion and Future Work

In order to predict and forecast waste bin occupancy, a smart waste management system was developed in this research. IoT, cloud computing, and machine learning are the foundations of the architecture, which aims to improve waste management systems. The improvement of the waste bin's intelligence was suggested in order to accelerate the waste collection, preserve the environment, and reduce waste collection time. Five machine learning models for forecasting and predictive modelling have been deployed in this study, and a comparative analysis has been done. The Generalised Additive Model produced the best results, with an MAE and MSE that were less deviated from the actual values (0.2407 and 1.9399, respectively).

Any future work in this area should concentrate on analysing information from several waste bins put in various locations and develop an application or end user dashboard over the cloud infrastructure where the concerned authorities may examine the prediction data.

## References

- Aazam, M., St-Hilaire, M., Lung, C.-H. and Lambadaris, I. (2016). Cloud-based smart waste management for smart cities, *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, pp. 188–193.
- Al-Masri, E., Diabate, I., Jain, R., Lam, M. H. and Reddy Nathala, S. (2018). Recycle.io: An iot-enabled framework for urban waste management, *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5285–5287.
- Bharathiraja, N., Deepa, T., Hariprasad, S. and Chokkalingam, A. (2022). Design and development of giot based intelligent smart waste management and predictive modelling, *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 7–12.
- Damrongkulkamjorn, P. and Churueang, P. (2005). Monthly energy forecasting using decomposition method with application of seasonal arima, *2005 International Power Engineering Conference*, pp. 1–229.
- Leninpugalhanthi, P., Bharanidaran, G., Bahiradhan, T., Abirami, E., Anandh, R. and Kumar, R. (2021). Enhanced smart waste management system with incinerator compartment, *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1, pp. 1313–1317.
- Li, J., Li, W., Guo, Z., Zhong, Z. and Wu, X. (2019). Lateral spread prediction based on generalized additive model for hot strip finishing mill, *2019 Chinese Control And Decision Conference (CCDC)*, pp. 1726–1730.
- Likotiko, E., Misaki, S., Matsuda, Y. and Yasumoto, K. (2021). Sgbs: A novel smart garbage bin system for understanding household garbage disposal behaviour, *2021 Thirteenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pp. 1–8.
- Medehal, A., Annaluru, A., Bandyopadhyay, S. and Chandar, T. (2020). Automated smart garbage monitoring system with optimal route generation for collection, *2020 IEEE International Smart Cities Conference (ISC2)*, pp. 1–7.
- N., S., Fathimal, P. M., R., R. and Prakash, K. (2019). Smart garbage segregation management system using internet of things(iot) machine learning(ml), *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pp. 1–6.
- Ndayishimiyepas, P., Wilson, C. and Kimwele, M. (2022). A hybrid model for predicting missing records in data using xgboost, *2022 IEEE International Symposium on Product Compliance Engineering - Asia (ISPCE-ASIA)*, pp. 1–5.
- Permanasari, A. E., Hidayah, I. and Bustoni, I. A. (2013). Sarima (seasonal arima) implementation on time series to forecast the number of malaria incidence, *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 203–207.

- Rijah, U. L. M. and Abeygunawardhana, P. K. W. (2023). Smart waste segregation for home environment, *2023 3rd International Conference on Advanced Research in Computing (ICARC)*, pp. 184–189.
- Singh, B., Kumar, P., Sharma, N. and Sharma, K. P. (2020). Sales forecast for amazon sales with time series modeling, *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*, pp. 38–43.
- Stefenon, S. F., Seman, L. O., Mariani, V. C. and Coelho, L. d. S. (2023). Aggregating prophet and seasonal trend decomposition for time series forecasting of italian electricity spot prices, *Energies* **16**(3).  
**URL:** <https://www.mdpi.com/1996-1073/16/3/1371>