CSCI 6444

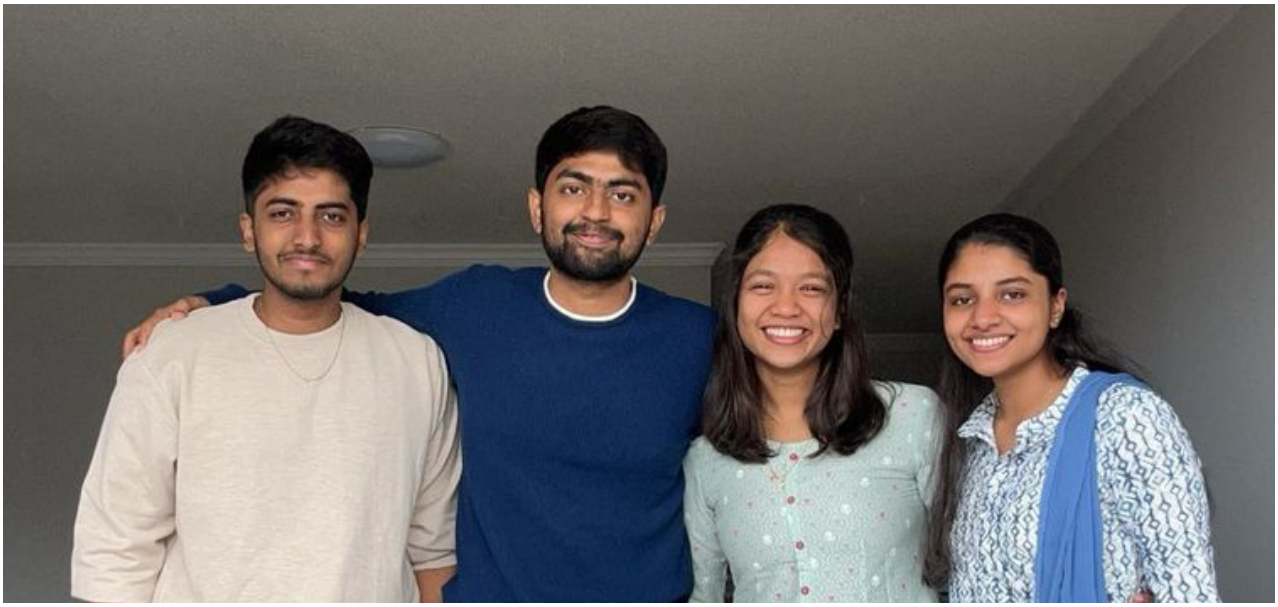
Project Report

Smart Waste Management

December 15, 2023

TABLE OF CONTENTS

1. Team Members



Kishan Ramesh - G44387483

Anantha Narayanan Sampath Varadharajan - G46252520

Anbu Ezhilmathi Nambi - G33350186

Aparna Shankar - G49628466

2. Introduction

Waste management is a significant concern for urban environments, necessitating innovative solutions to optimize collection processes and promote responsible disposal practices. This report introduces a Smart Waste Management and Classification System, aiming to minimize costs and enhance waste segregation efficiency.

The project focuses on creating a comprehensive system using two different technology approaches to compare their performance. The goal is to address the challenges of irregular waste disposal and the resulting hygiene issues from garbage accumulation around bins. The system utilizes sensor data for analytics, providing insights through visualization and trend analysis to aid decision-making in waste management practices.

A key component of the project is the implementation of automated waste classification. This involves using image data to categorize waste into two primary classes. The first classifier organizes general waste into different categories for efficient disposal, while the second classifier focuses on medical waste classification, ensuring compliance with specific handling and disposal procedures related to healthcare refuse.

Acknowledging the unique challenges posed by medical waste, the project not only contributes to broader waste management goals but also addresses the specific needs of healthcare-related refuse. By emphasizing proper waste handling and disposal procedures, the Smart Waste Management and Classification System aims to minimize waste overflow, promote recycling efforts, and enhance overall waste collection efficiency within the community.

3. Problem Statement

The United States generates approximately 811 tons of municipal garbage waste annually, as observed in the year 2018. This project aims to enhance waste management and sorting mechanisms for waste pick-up services, with a specific focus on efficiently collecting and segregating waste to reduce associated costs. A key challenge in waste collection is the inefficient handling of bins with low fill levels, specifically those below 50%, leading to elevated operational expenses. To address this issue, the project seeks to implement a system that monitors the fill levels of trash containers, allowing drivers to prioritize fully filled bins. The optimization strategy involves monitoring fill levels and measuring bin weights to predict fill levels over time. Additionally, the project aims to automate waste classification to enhance overall waste management efficiency.

4. Objectives

1. To motivate the early removal of garbage collected in bins to avoid cluttering around the vicinity and to avoid unnecessary commute of garbage collectors.
2. Integrate an automated waste classification system to categorize diverse waste types, improving the garbage sorting processes and alerting when higher fill level bins to reduce unnecessary transportation and costs.
3. Utilize advanced analytics for sensor data, providing actionable insights through visualizations and trend analyses to inform decision-making in waste management practices.
4. Create a user-friendly interface for waste management personnel to access real-time information on fill levels, trends, and classifications, facilitating informed decision-making.
5. Implement sustainable practices within the waste management system to contribute to environmental conservation and promote responsible waste disposal.
6. Collaborate with local municipalities and waste management agencies to ensure seamless integration and adoption of the proposed system, considering regulatory and logistical considerations.

5. Related Work

The project's inspiration stems from insights derived from the examination of research papers. The paper[9] on "Waste Management and Prediction of Air Pollutants Using IoT and Machine Learning Approach" has provided valuable insights, to explore the possibilities of forecasting the air quality in the vicinity. The paper's objective is to construct an efficient model for classifying bin status (filled, half-filled, unfilled), facilitating real-time garbage level monitoring and notifications through an alert mechanism. Utilizing our collected data, a similar approach to categorizing bin status has been implemented.

Addressing limitations identified in the paper [7], titled "Waste Object Detection and Classification," which were attributed to a constrained dataset, the project strives to enhance model generalization by acquiring a more comprehensive set of real-world images. Furthermore, drawing insights from the paper [11] titled "A deep learning approach for medical waste classification," the project introduces the concept of sub-classifying medical waste from broader waste categories. The referenced paper proposes a deep-learning approach using ResNeXt for the identification and classification of eight types of medical waste, achieving an accuracy of 97.2%. This addition significantly influences the project's strategy to augment the overall waste classification system.

Additionally, the project incorporates concepts from the paper [8] entitled "Intelligent waste management system using deep learning with IoT." This paper introduces a smart waste management system that integrates IoT and deep learning for optimal garbage management. The methodology involves employing convolutional neural networks for waste classification (digestible and indigestible) with an accuracy of 95.31%. It also implements a smart trash box architecture equipped with multiple sensors for real-time data collection and transmission. Certain elements from the workflow and pipeline design references outlined in this paper have been integrated into the development of the project's own pipeline.

6. Project Design

In the project's workflow, two distinct phases are evident. During the initial phase, sensors are strategically placed to collect data from waste bins, enabling an analysis of the bin fill levels, weight, and the surrounding air quality. Once the sensors identify that a bin has reached its capacity, the waste collection process is triggered. The subsequent phase of the project is dedicated to the image classification of garbage images, categorizing them into different groups.

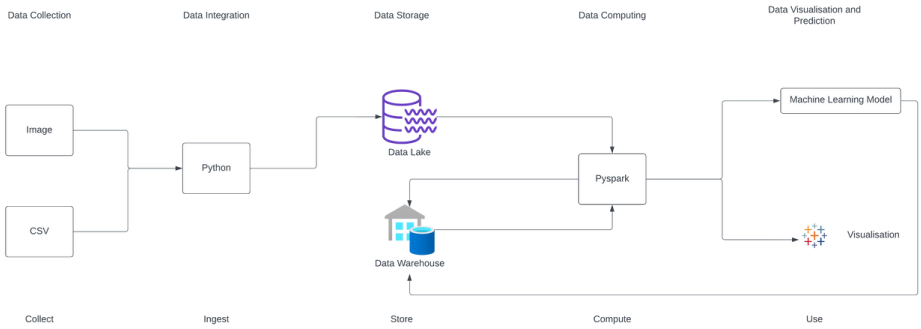


Figure 6.1 : Architecture design

The figure 6.1 above illustrates the system architecture design of the project. The Python script is employed to ingest both image and CSV data collected from bin sensors into the Data Lake, serving as a repository for the unprocessed raw data. This approach not only accommodates the expanding data volume (scalability) but also ensures data durability, guaranteeing the integrity of the stored information. Subsequently, the data preprocessing stage is initiated using Pyspark, where standard methods of data imputation are applied to handle missing or incomplete data points. Data imputation is a crucial technique to ensure the dataset's completeness and suitability for analysis.

The preprocessed data is then loaded into the data warehouse. From the data warehouse, the processed image data is fed into the garbage classification model for prediction, while the sensor data is loaded into Tableau. Tableau functions as a data visualization platform, enabling the creation of interactive dashboards and reports. These visualizations play a crucial role in monitoring waste bin fill levels and air quality data, facilitating a data-driven decision-making based on the collected information.

Upon the authorities receiving alerts triggered by sensor detection, the waste classification objective is to automatically identify and segregate various types of waste, including recyclable materials (such as paper, metal, glass, plastics, etc.) and non-recyclable materials (organic, bio wastes, etc.). The predicted images are reinstated into the Data Warehouse for future reference.

Design Diagram

The figure 6.2 below shows the UML design diagram of the 2 different tech-stacks implemented in this project.

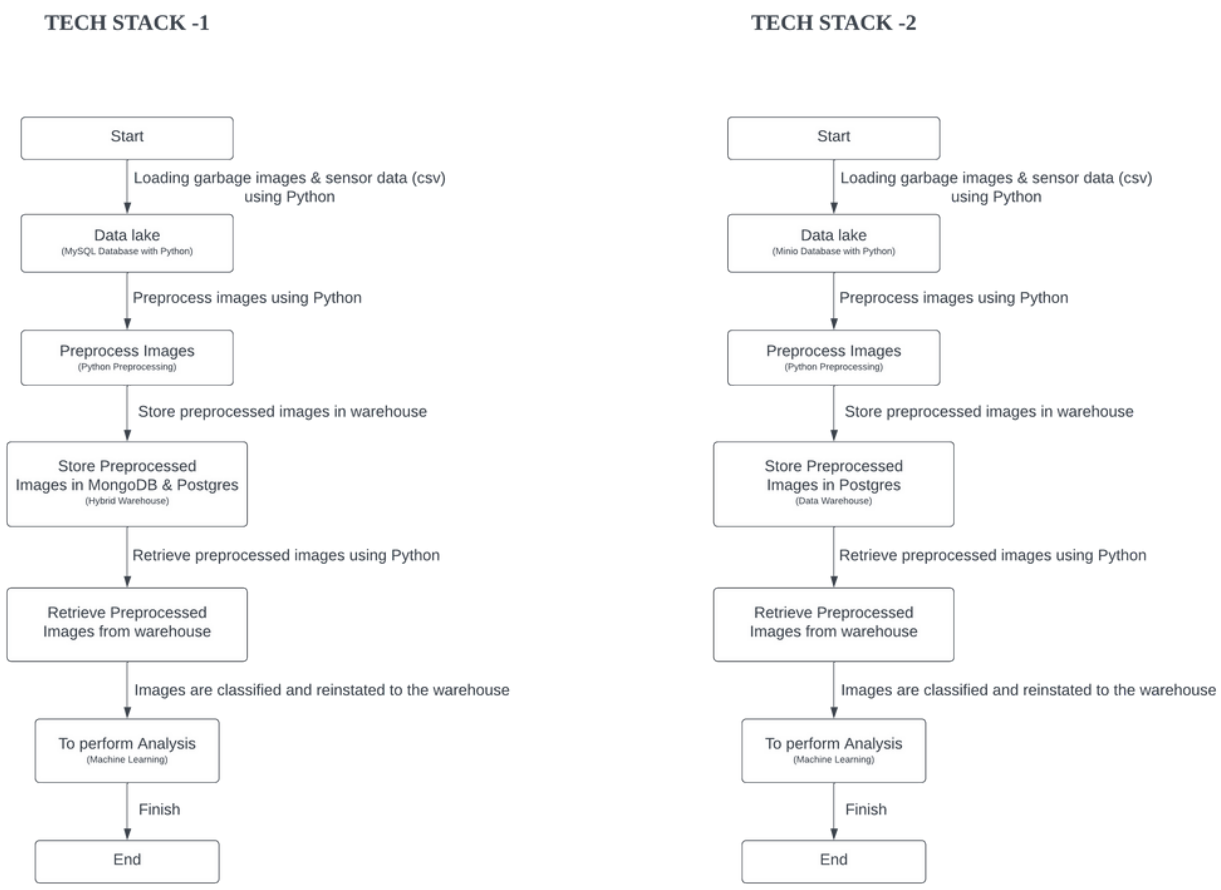


Figure 6.2: UML design of two tech stacks

Technology Stack 1:

1. Data Collection: The process initiates with sensors placed inside garbage bins gathering diverse data such as fill levels, trash weight, and the air quality in the surrounding area.
2. Garbage Level Check: The sensors generate an alert if the garbage level in the bin is overfilled. Otherwise, no alert is issued, thereby reducing the frequency of unnecessary pickups.
3. Data Storage: Python scripts are employed to ingest the collected sensor data and image data, storing them in MySQL database acting as the Data Lake.
4. Pre-processing: The image data undergoes pre-processing, encompassing tasks such as de-brightness, resizing, outlier detection, and image scaling. Similarly, the sensor data is also cleaned to extract important features.
5. Storing Pre-processed Data: The preprocessed images are then stored in the data warehouse, utilizing MongoDB and Postgres as the chosen Data Warehouse solutions. MongoDB is a NoSQL database known for its flexible and schema-less data model. It excels at handling unstructured or semi-structured data, making it suitable for certain types of data, while PostgreSQL, being a relational database, enforces a structured schema.

6. Waste Classification: Python scripts were employed to retrieve preprocessed images from the data warehouse, which were subsequently input into the classification model for image categorization. The deep learning algorithm MobileNet V2 is utilized to classify the waste images into distinct categories such as Plastic, Paper, Glass, Trash, E-waste etc.

Technology Stack 2:

1. Data Collection: The process begins with sensors placed inside garbage bins collecting diverse data, encompassing fill levels, trash weight, and air quality in the vicinity.
2. Garbage Level Check: Sensors generate an alert if the garbage level in the bin gets overfilled. If it doesn't, no alert is issued, reducing the frequency of unnecessary pickups.
3. Data Storage: Python scripts are used to ingest the collected sensor data and image data, storing them into the MinIO database, functioning as the Data Lake.
4. Pre-processing: The image data undergoes pre-processing, involving tasks like increasing the brightness, resizing, outlier detection, and image scaling.
5. Storing Pre-processed Data: The preprocessed images and the trash can sensor data are then stored in Postgres, serving as the Data Warehouse.
6. Waste Classification: Python scripts retrieve preprocessed images and sensor data from the data warehouse, subsequently loading them into the classification model for image categorization. The deep learning algorithm MobileNet V2 is employed to classify the waste images into various categories such as Plastic, Paper, Glass, Trash etc, and further medical waste classification into 11 categories.

Use case diagram

The use case diagram depicted in figure 6.3 outlines the garbage collection and disposal process:

1. Data providers, interacting with the system—be it a smart garbage bin or a waste management application—are tasked with importing and preprocessing the data.
2. The system plays a pivotal role in coordinating the entire process, facilitating communication with the user, the garbage truck, and the sensors.
3. Sensors, strategically positioned within the garbage bins, gather data related to fill level, weight, and the surrounding air quality. Upon reaching a specific threshold, they trigger alerts to authorities, prompting the dispatch of a garbage truck for bin clearance.
4. The garbage truck is responsible for the collection and transportation of garbage.
5. An authorized individual, potentially a waste management official, oversees the process by ensuring the timely dispatch of the garbage truck.
6. Public users can access data visualizations and image classifications through the waste management application.

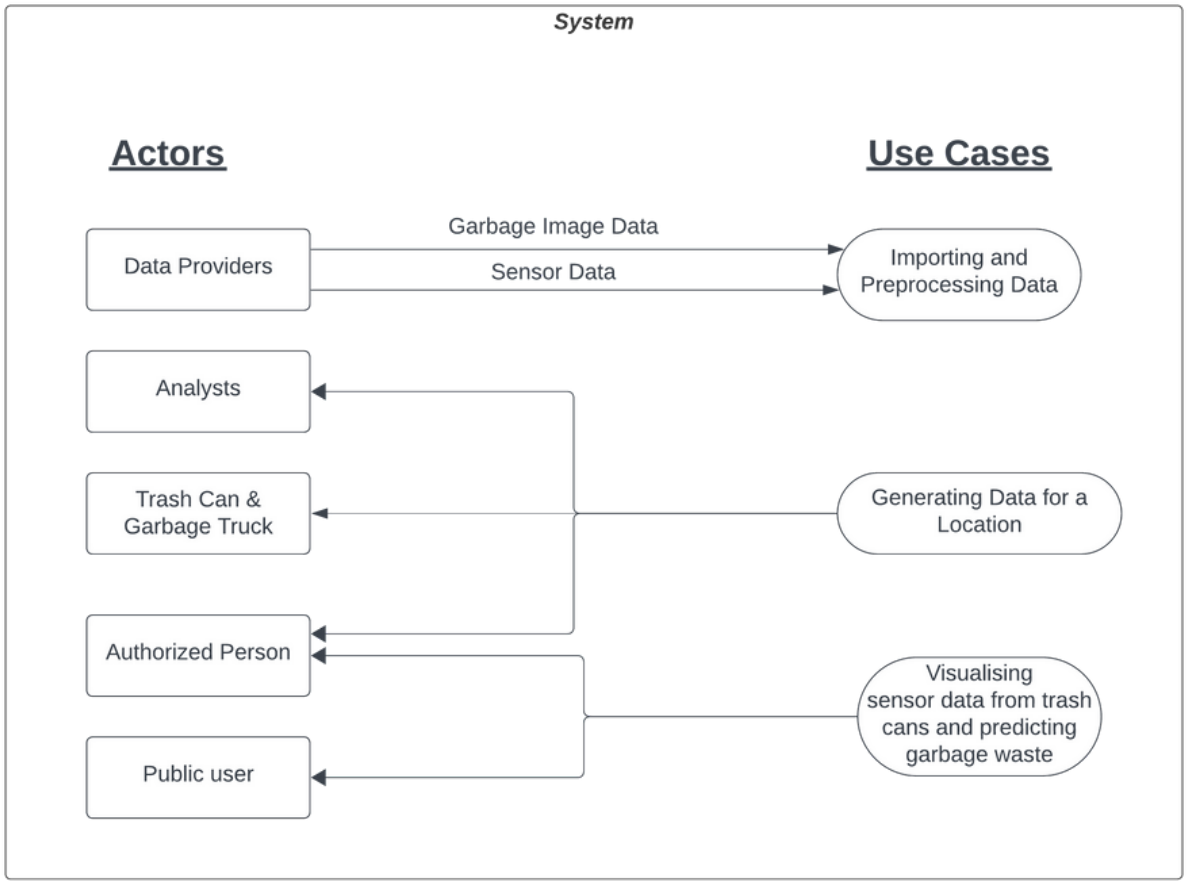


Figure 6.3: Use case diagram

7. Big Data Challenges

- **Scalability management** - Recognizing the potential scalability issues with an increase in data volume, the project implements scalable architecture and data processing techniques to effectively handle growing data loads. MinIO is used as Data Lake in one of the tech-stacks, designed for high-performance object storage. It ensures that the system can handle increased loads and storage requirements (Volume). Notably, a key advantage is MinIO's cost-effectiveness for object storage, since it is open-source.
- **Data Durability** - Data durability is the ability of a data storage system to reliably and persistently retain data over time. MinIO also ensures durability of data. The collected data can be distributed across multiple drives and nodes, ensuring that even if one or more nodes fail, the data remains accessible from other replicas.

- **Veracity** - Data integrity is paramount as data may exhibit noise, incompleteness, or inconsistency. Ensuring meticulous attention to the quality and accuracy of the data is imperative for making well-informed decisions.
- **Fault Tolerance** - The failure to detect trash can impact the planned operations of a waste management system. Data pre-processing and cleaning techniques for both the sensor and image data are designed to recognize and manage any missing or inconsistent data, ensuring the dataset's overall quality is preserved.
- **Variety** - Dealing with data variety becomes crucial because analytics platforms need to process and analyze diverse data types simultaneously. This is particularly relevant when aiming to derive meaningful insights from the extensive and diverse datasets inherent in big data, utilizing data visualization tools. In this project, Tableau and plotly using python was utilized for data visualization insights. Additionally, the implementation of a hybrid Data Warehouse structure in one of the tech stacks combines PostgreSQL, excelling in structured data with well-defined schemas, and MongoDB, proficient in managing unstructured or semi-structured data. This hybrid setup accommodates a wide range of data types.
- **Data Integration** - Integrating data from different sources and with different structures requires careful consideration. Data integration solutions handle the variety of data formats and ensure interoperability.

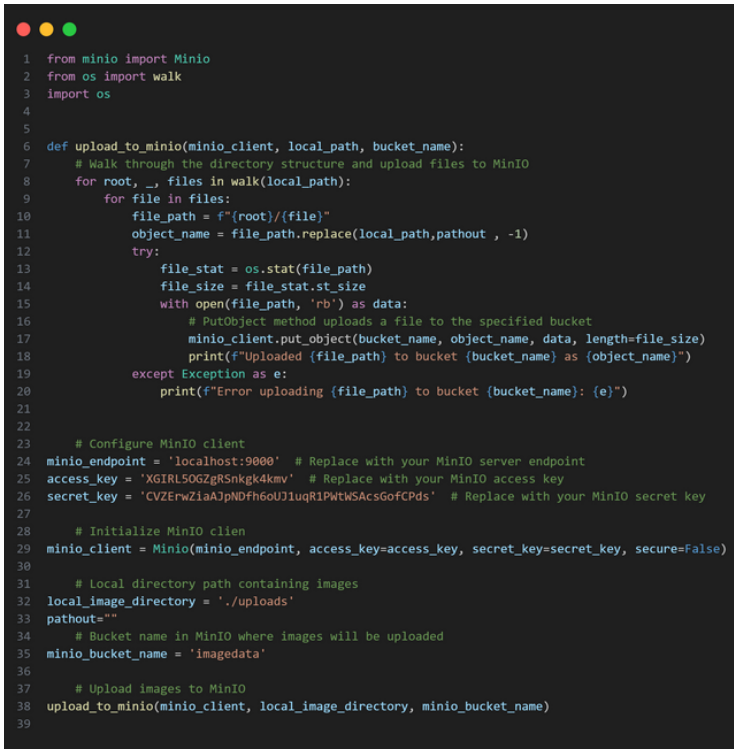
However, one challenge is that the system currently lacks the capability to provide real-time notifications alerting users about the status of garbage collection, primarily because the datasets utilized in the project are not live.

8. Project Outcome and Discussion

The project employs a sophisticated system architecture designed to optimize waste management. It includes automated waste classification, triggered by sensor alerts, distinguishing between recyclable and non-recyclable materials. Visualized insights empower informed decision-making, and the entire process is orchestrated in a well-defined workflow, from initial data collection through ingestion to the final waste classification.

Data Ingestion :

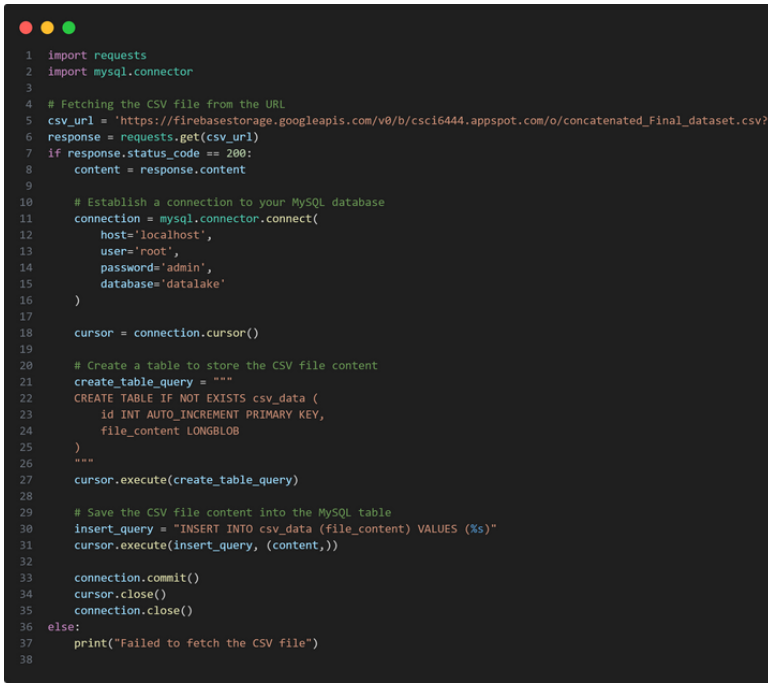
The goal of data ingestion is to gather data efficiently, ensure its quality, and make it available for downstream applications. The collected data from sensors and waste images are ingested using Python Script. Then, the ingested data is typically stored in the Data Lake, using MYSQL and MinIO in the two tech stacks. Integration with these storage systems is a critical component of the data ingestion process.



```python
from minio import Minio
from os import walk
import os


def upload_to_minio(minio_client, local_path, bucket_name):
    # Walk through the directory structure and upload files to MinIO
    for root, _, files in walk(local_path):
        for file in files:
            file_path = f"{root}/{file}"
            object_name = file_path.replace(local_path,pathout , -1)
            try:
                file_stat = os.stat(file_path)
                file_size = file_stat.st_size
                with open(file_path, 'rb') as data:
                    # PutObject method uploads a file to the specified bucket
                    minio_client.put_object(bucket_name, object_name, data, length=file_size)
                    print(f"Uploaded {file_path} to bucket {bucket_name} as {object_name}")
            except Exception as e:
                print(f"Error uploading {file_path} to bucket {bucket_name}: {e}")


    # Configure MinIO client
minio_endpoint = 'localhost:9000'  # Replace with your MinIO server endpoint
access_key = 'XGIRL5OGZgRSnkgk4kmv'  # Replace with your MinIO access key
secret_key = 'CVZErwZiaAJpNDfh6oUJ1uqR1PWtWSAcsGofCPds'  # Replace with your MinIO secret key

    # Initialize MinIO clien
minio_client = Minio(minio_endpoint, access_key=access_key, secret_key=secret_key, secure=False)

    # Local directory path containing images
local_image_directory = './uploads'
pathout=""
    # Bucket name in MinIO where images will be uploaded
minio_bucket_name = 'imagedata'

    # Upload images to MinIO
upload_to_minio(minio_client, local_image_directory, minio_bucket_name)

```

Figure 8.1: Data ingestion and storage to Data Lake

The figure 8.1 is a snippet of data ingestion performed and subsequent storage to MinIO Data Lake to further perform pre-processing.



```python
import requests
import mysql.connector

# Fetching the CSV file from the URL
csv_url = 'https://firebasestorage.googleapis.com/v0/b/csci6444.appspot.com/o/concatenated_Final_dataset.csv?
response = requests.get(csv_url)
if response.status_code == 200:
    content = response.content

    # Establish a connection to your MySQL database
    connection = mysql.connector.connect(
        host='localhost',
        user='root',
        password='admin',
        database='datalake'
    )

    cursor = connection.cursor()

    # Create a table to store the CSV file content
    create_table_query = """
    CREATE TABLE IF NOT EXISTS csv_data (
        id INT AUTO_INCREMENT PRIMARY KEY,
        file_content LONGBLOB
    )
    """
    cursor.execute(create_table_query)

    # Save the CSV file content into the MySQL table
    insert_query = "INSERT INTO csv_data (file_content) VALUES (%s)"
    cursor.execute(insert_query, (content,))

    connection.commit()
    cursor.close()
    connection.close()
else:
    print("Failed to fetch the CSV file")

```

Figure 8.2 Data ingestion of sensor data

The Figure 8.2 is a snippet of data ingestion of sensor data as csv file from firebase and storing it into MySQL database.

Descriptive Analytics:

The collected data is initially visualized to understand and gain insights before conducting analytics. This visualization includes a geographical map shown in figure 8.3, that illustrates the geographical distribution of the collected data across the United States.

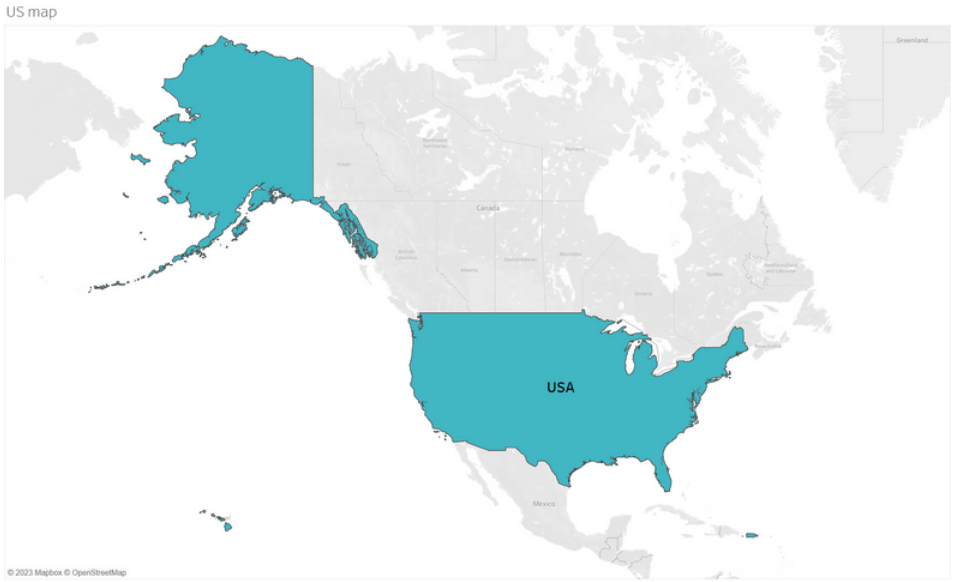


Figure 8.3: Geographical distribution of data collection in US

Additionally, in figure 8.4 a bar plot is employed to convey cumulative waste collection trends in the United States over the time period spanning from 2000 to 2018. It can be inferred that the amount of waste collected in the US has remained relatively stable over the years. However, there is a notable surge in 2018, compared to the previous years.

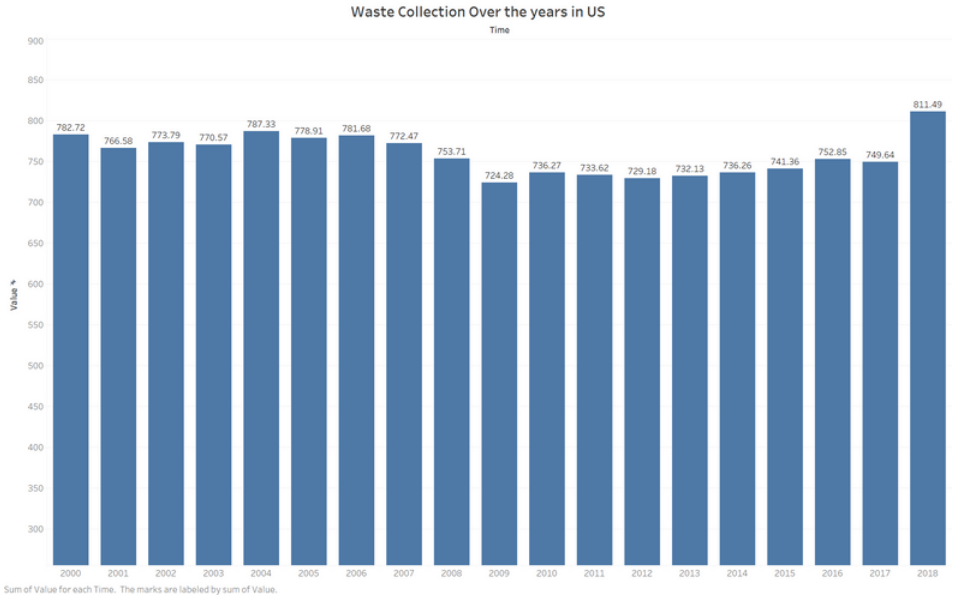


Figure 8.4 : Bar plot of waste collection over the years in US

Figure 8.5 illustrates the Density distribution plot of the trash can sensor data. This shows that the data follows a normal distribution.

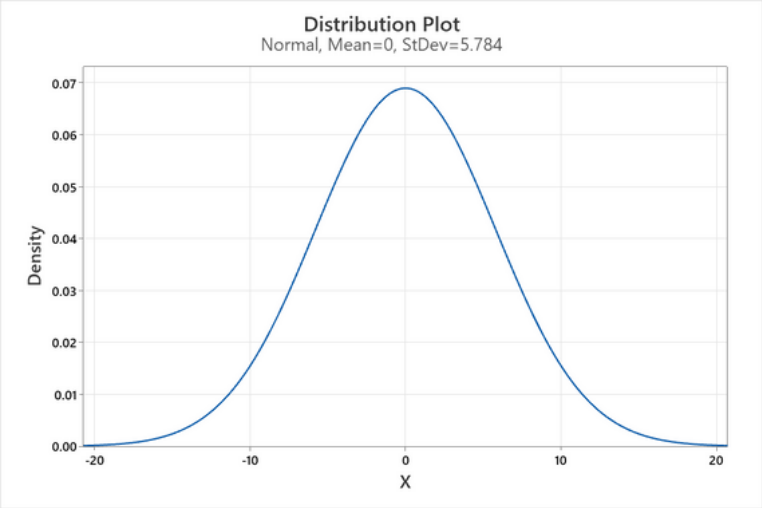


Figure 8.5: Data distribution plot

Pre-processing :

The trash can data sample before pre-processing and feature extraction are shown in table 1 below.

fill_percentage	fill_status	battery_percentage	Distance (mm)	trash_can_type	width(ft)	height(ft)	depth(ft)	filled_capacity(lbs)	manufacture_year	Garbage_can_id	Nitrogen Oxide	Sulphur Dioxide	Carbon Monoxide	Organic Carbon	NMVOCs	Black Carbon	Ammonia	air_quality
0	Empty	11	2946	6 yard dumpster	6'0"	5'10"	5'8"	0	2011	10220273	278	437	479	246	301	269	192	0
16	Less Filled	10	1984	6 yard dumpster	6'0"	5'10"	5'8"	192	2011	9623407	2881	900	1988	1488	3081	2500	938	32
17	Less Filled	10	1803	6 yard dumpster	6'0"	5'10"	5'8"	204	2009	11637568	1388	2412	1535	1618	918	1118	988	34
18	Less Filled	10	1830	6 yard dumpster	6'0"	5'10"	5'8"	216	2010	8441848	767	2328	2478	972	1183	2633	2211	36
19	Less Filled	10	1827	6 yard dumpster	6'0"	5'10"	5'8"	228	2011	8641665	1400	2279	905	2142	1784	868	716	38
20	Less Filled	10	1799	2 yard dumpster	5'10"	3'10"	2'10"	80	2011	10675534	625	1700	980	820	2165	1200	1900	20
20	Less Filled	10	1809	2 yard dumpster	5'10"	3'10"	2'10"	80	2008	12682998	2175	1205	2385	640	1340	525	2345	20
30	Filled	10	1617	2 yard dumpster	5'10"	3'10"	2'10"	120	2008	8225200	540	1087	1537	447	1147	480	1097	30
40	Filled	10	1399	2 yard dumpster	5'10"	3'10"	2'10"	160	2012	9648534	512	1130	388	842	1220	740	1105	40
50	Half Filled	10	1294	2 yard dumpster	5'10"	3'10"	2'10"	200	2010	12697943	620	480	1100	975	432	248	264	50
50	Half Filled	89	1298	4 yard dumpster	5'10"	5'1"	4'2"	400	2012	8473402	649	370	798	860	1160	647	988	75
50	Half Filled	89	1311	6 yard dumpster	6'0"	5'10"	5'8"	852	2011	8632240	307	798	470	824	613	424	222	142
71	Filled	89	1256	6 yard dumpster	6'0"	5'10"	5'8"	852	2011	8632240	307	798	470	824	613	424	222	142
71	Filled	89	1259	6 yard dumpster	6'0"	5'10"	5'8"	852	2008	10691505	802	430	315	902	327	1042	956	142
71	Filled	89	2287	6 yard dumpster	6'0"	5'10"	5'8"	852	2010	9677565	480	212	614	431	357	608	782	142
72	Filled	89	745	6 yard dumpster	6'0"	5'10"	5'8"	864	2012	9482021	515	610	217	308	401	388	397	144
80	Filled	89	619	4 yard dumpster	5'10"	5'1"	4'2"	640	2012	10623515	249	374	169	206	185	618	375	120
80	Filled	89	578	4 yard dumpster	5'10"	5'1"	4'2"	640	2009	11485780	184	234	246	388	388	399	154	120
90	Over Filled	10	563	2 yard dumpster	5'10"	3'10"	2'10"	360	2008	9657493	261	313	194	486	274	267	501	90
90	Over Filled	89	544	4 yard dumpster	5'10"	5'1"	4'2"	720	2009	11429454	189	182	424	238	489	467	116	135
90	Over Filled	10	1097	6 yard dumpster	6'0"	5'10"	5'8"	1080	2008	9674621	589	547	198	406	386	249	330	180
91	Over Filled	10	340	6 yard dumpster	6'0"	5'10"	5'8"	1092	2009	11418431	311	298	119	215	512	182	220	182
90	Over Filled	10	1097	6 yard dumpster	6'0"	5'10"	5'8"	1080	2008	9674621	589	547	198	406	386	249	330	180
91	Over Filled	10	340	6 yard dumpster	6'0"	5'10"	5'8"	1092	2009	11418431	311	298	119	215	512	182	220	182

Table 1: Data overview before pre-processing

The data undergoes pre-processing to detect any missing values and inconsistencies, which are then addressed through standard imputation methods to ensure the overall quality and reliability of the dataset. The columns such as manufacture year, concentration of organic carbon and other gases are dropped to extract the key features for subsequent analysis and visualization insights. Table 2 provides an overview of the data after undergoing these processing steps.

fill_percentage	fill_status	battery_percentage	Distance (mm)	trash_can_type	width(ft)	height(ft)	depth(ft)	filled_capacity(lbs)	Garbage_can_id	air_quality
0	Empty	11	2946	6 yard dumpster	6'0"	5'10"	5'8"	0	10220273	0
16	Less Filled	10	1984	6 yard dumpster	6'0"	5'10"	5'8"	192	9623407	32
17	Less Filled	10	1803	6 yard dumpster	6'0"	5'10"	5'8"	204	11637568	34
18	Less Filled	10	1830	6 yard dumpster	6'0"	5'10"	5'8"	216	8441848	36
19	Less Filled	10	1827	6 yard dumpster	6'0"	5'10"	5'8"	228	8641665	38
20	Less Filled	10	1799	2 yard dumpster	5'10"	3'10"	2'10"	80	10675534	20
20	Less Filled	10	1809	2 yard dumpster	5'10"	3'10"	2'10"	80	12682998	20
30	Filled	10	1617	2 yard dumpster	5'10"	3'10"	2'10"	120	8225200	30
40	Filled	10	1399	2 yard dumpster	5'10"	3'10"	2'10"	160	9648534	40
50	Half Filled	10	1294	2 yard dumpster	5'10"	3'10"	2'10"	200	12697943	50
50	Half Filled	89	1298	4 yard dumpster	5'10"	5'1"	4'2"	400	8473402	75
50	Half Filled	89	1311	6 yard dumpster	6'0"	5'10"	5'8"	852	8632240	142
71	Filled	89	1256	6 yard dumpster	6'0"	5'10"	5'8"	852	8632240	142
71	Filled	89	1259	6 yard dumpster	6'0"	5'10"	5'8"	852	10691505	142
71	Filled	89	2287	6 yard dumpster	6'0"	5'10"	5'8"	852	9677565	142
72	Filled	89	745	6 yard dumpster	6'0"	5'10"	5'8"	864	9482021	144
80	Filled	89	619	4 yard dumpster	5'10"	5'1"	4'2"	640	10623515	120
80	Filled	89	578	4 yard dumpster	5'10"	5'1"	4'2"	640	11485780	120
90	Over Filled	10	563	2 yard dumpster	5'10"	3'10"	2'10"	360	9657493	90
90	Over Filled	89	544	4 yard dumpster	5'10"	5'1"	4'2"	720	11429454	135
90	Over Filled	10	1097	6 yard dumpster	6'0"	5'10"	5'8"	1080	9674621	180
91	Over Filled	10	340	6 yard dumpster	6'0"	5'10"	5'8"	1092	11418431	182
90	Over Filled	10	1097	6 yard dumpster	6'0"	5'10"	5'8"	1080	9674621	180
91	Over Filled	10	340	6 yard dumpster	6'0"	5'10"	5'8"	1092	11418431	182

Table 2: Data overview after pre-processing

The code displayed in figure 8.6, fetches images stored from PostgreSQL and retrieves the corresponding images from MongoDB based on certain metadata. The function identifies images that meet the criteria for being outliers based on their dimensions. The images are then pre-processed by de-brightening to adjust the exposure, resizing and scaling.

```python
# Now 'retrieved_images' list contains OpenCV images fetched from PostgreSQL
# You can further process or display these images as needed
# Connect to MongoDB
mongo_client = MongoClient('mongodb://localhost:27017/')
mongo_db = mongo_client['mongodb_database']
mongo_collection = mongo_db['images']

# List to store retrieved images
retrieved_images = []
filenames=[]
# Function to convert retrieved byte data to OpenCV image
def convert_byte_data_to_image(byte_data):
    nparr = np.frombuffer(byte_data, np.uint8)
    return cv2.imdecode(nparr, cv2.IMREAD_COLOR)

# Fetch images from MongoDB based on metadata fetched from PostgreSQL
for metadata in images_data:
    image_id = metadata[0]  # Assuming image_id is stored in the first column
    filename = metadata[1]
    # Fetch image data from MongoDB based on image_id
    image_data_mongodb = mongo_collection.find_one({'pg_id': image_id})
    if image_data_mongodb and 'pg_id' in image_data_mongodb:
        img = convert_byte_data_to_image(image_data_mongodb['image_data'])
        retrieved_images.append(img)
        filenames.append(image_id)

```

Figure 8.6: Detecting outliers in image data

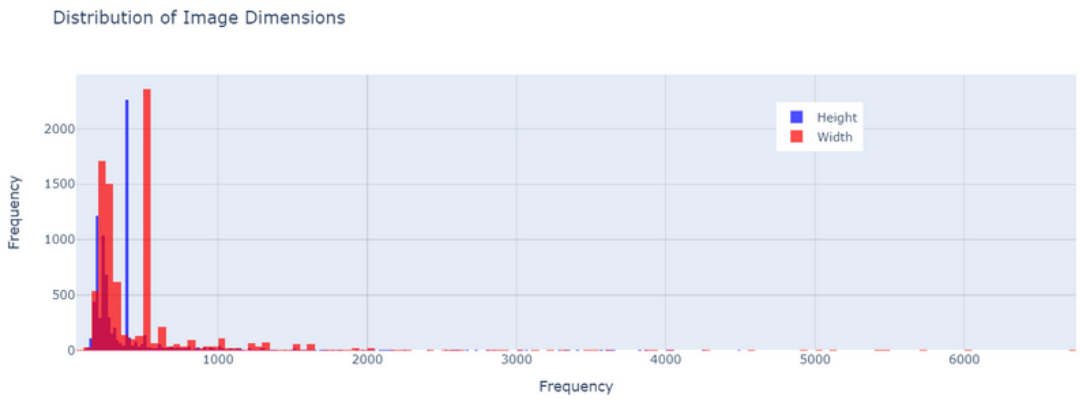


Figure 8.7: Distribution of image dimensions

It can be inferred from the distribution graph in figure 8.7, that a large number of images have dimensions concentrated around a lower range.

Data Warehouse:

After pre-processing, the images and sensor data are loaded into the data warehouse. The figure 8.8 is a snippet of code to establish connection with PostgresSQL.

```python
# Lake to warehouse
from minio import Minio
from io import BytesIO
import psycopg2
from psycopg2 import sql
import os
from datetime import datetime

minio_bucket_name = 'preprocessbucket'
# PostgreSQL connection details
postgres_host = 'localhost'
postgres_port = '5432'
postgres_dbname = 'mydatabase'
postgres_user = 'postgres'
postgres_password = 'admin123'

# Connect to PostgreSQL
conn = psycopg2.connect(host=postgres_host,
                        port=postgres_port,
                        dbname=postgres_dbname,
                        user=postgres_user,
                        password=postgres_password)

cursor = conn.cursor()

```

Figure 8.8 : Postgres SQL connection establishment

After successful connection, the pre-processed image fetched from the MinIO is loaded into a table in Postgres SQL.

```python
import uuid


output_minio_bucket ='imagedata'
object = objects = minio_client.list_objects(output_minio_bucket, recursive=True)
for obj in objects:
    if(obj.object_name in outlier_images):
        continue
    else:
        object_data = minio_client.get_object(output_minio_bucket, obj.object_name)
        object_info = minio_client.stat_object(output_minio_bucket, obj.object_name)
        content = object_data.read()
        img = convert_byte_data_to_image(content)
        height=img.shape[0]
        id=str(uuid.uuid4())
        width=img.shape[1]
        category = 'temp'
        filesize = object_info.metadata["Content-Length"]
        created_at = datetime.strptime(object_info.metadata["Date"], '%a, %d %b %Y %H:%M:%S %Z')  # Parse last_modified to datetime


        insert_query = '''
            INSERT INTO images  (id, filename, filesize , image, height, width, updated_time, category)
            VALUES (%s, %s, %s,%s, %s, %s, %s, %s)
        '''
        cursor.execute(insert_query, (id,obj.object_name, filesize, content, height, width, created_at,category))
        conn.commit()
```

Figure 8.9 : Loading preprocessed images into PostgresSQL

Data Visualization:

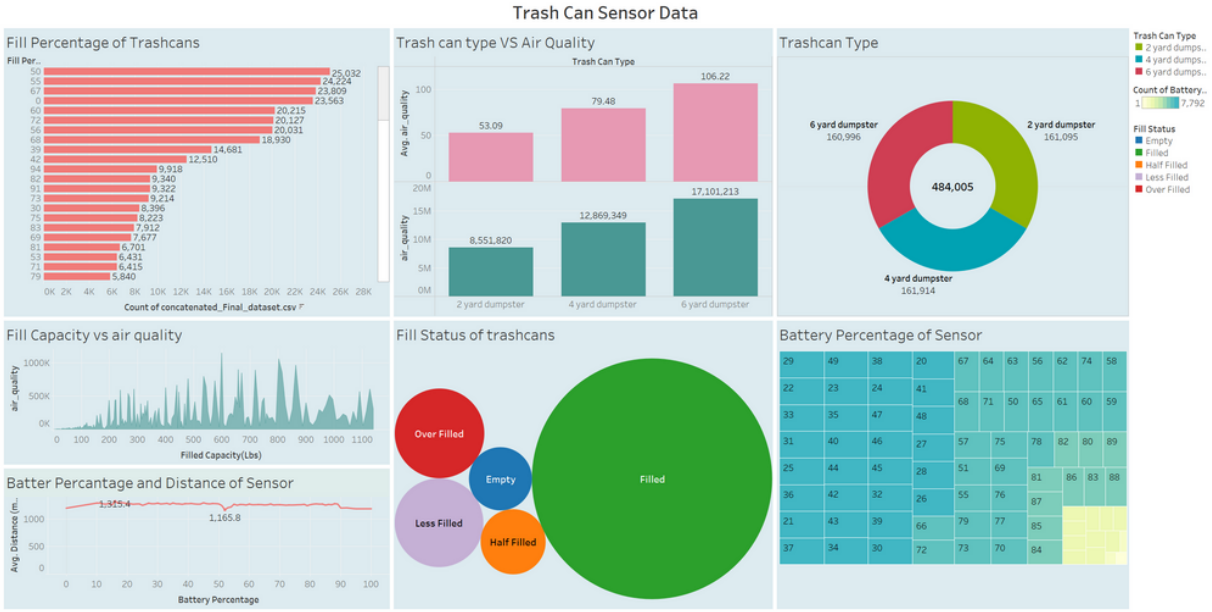


Figure 8.10 : Visualization Dashboard for Trash can sensor data

The dashboard in figure 8.10 gives a comprehensive analysis of the sensor data. The following inferences are made from each of the tools:

- The bar plot on the top left depicts the count of all the trash-cans with respect to their different fill percentages. It is noticeable that the count of trash cans containing approximately 50% of waste and those that are empty is nearly equal. Some trash cans are more full than others, indicating that they may need to be emptied more frequently.
- The Stacked bar plot on Trash can type Vs Air quality, shows the air quality surrounding the different types of trash cans. The 6 yard dumpsters seem to have a bad air quality on average (106.22) compared to the other two types of dumpsters. As the size of dumpster increases, the average air quality is also compromised.
- The Donut chart visually represents the distribution of different types of trash cans, specifically the 2-yard, 4-yard, and 6-yard dumpsters. Notably, these categories exhibit a nearly equal distribution among the total dataset of 484,005.
- The area graph on Fill Capacity Vs Air Quality is unevenly distributed, this shows there is no significant correlation between these factors.
- The bubble chart highlighting trash can fill status reveals that bins labeled as "filled" have the highest representation, indicating that a considerable number of bins may require more frequent emptying.
- The heatmap on the dashboard illustrates the distribution of sensor battery percentages, where darker colors indicate a higher concentration of sensors at that specific percentage level.
- The line chart of Battery percentage Vs Distance of sensor shows that as the battery percentage decreases, the average distance (in meters) of the sensor is almost stable. As the battery drains to around 50%, there is a significant drop in the sensor's distance to about 1,165.8 meters.

In summary, it can be inferred that larger dumpsters may impact air quality, and there might be a critical threshold in battery percentage (around 50%) where sensor distances experience a notable decrease. This analysis aids in optimizing waste management strategies, to ensure timely pick-ups for bins that are likely to reach or exceed capacity.

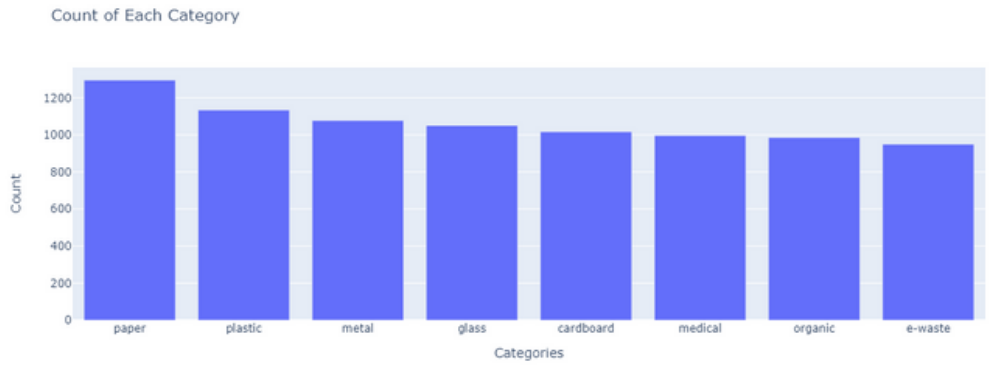


Figure 8.11: Bar chart for count of images in each category of waste

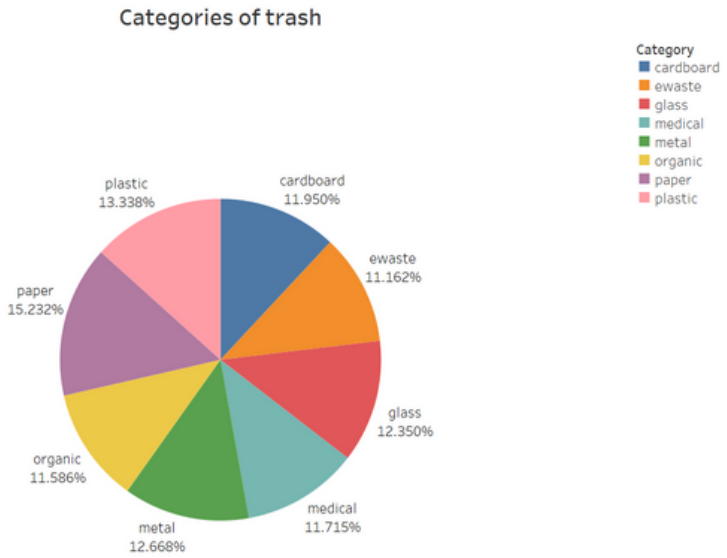


Figure 8.12: Pie chart of categories of trash

Using the image data of collected wastes, the bar plot in figure 8.11 reveals the count of images in each waste category, including paper, plastic, metal, glass, cardboard, medical, organic, and e-waste. The distribution across these categories appears fairly even, mitigating any bias in the prediction model for classification.

Additionally, the pie chart visually represents the proportion of images in each waste category. Notably, the paper waste images have a slightly higher proportion compared to other categories.

Garbage image classification and prediction:

The Preprocessed data from the Data Warehouse, is loaded into the machine learning model for image classification and prediction. The deep neural network model Mobile-net V2 is used for image classification. The figure 8.13 below shows the model layers.

```
mobilenetv2_layer = mobilenetv2.MobileNetV2(include_top = False, input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT,IMAGE_CHANNELS),
                                            weights = None)


mobilenetv2_layer.trainable = False


model1 = Sequential()
model1.add(keras.Input(shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))


def mobilenetv2_preprocessing(img):
  return mobilenetv2.preprocess_input(img)

model1.add(Lambda(mobilenetv2_preprocessing))

model1.add(mobilenetv2_layer)
model1.add(tf.keras.layers.GlobalAveragePooling2D())
model1.add(Dense(len(categories), activation='softmax'))

model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])

model1.summary()
```

Figure 8.13 : Model layer description

The preprocessed images are loaded into the model, the classification model gives an overall test accuracy of 82.75% on garbage classification and 97.54% on medical waste classification. Then the test images were loaded to predict the category and the classified images were reinstated into the data warehouse for future reference. The figure 8.14 and 8.15 below shows the snippet code of prediction classes.

```
1  import keras.applications.mobilenet_v2 as mobilenetv2
2  def mobilenetv2_preprocessing(img):
3    return mobilenetv2.preprocess_input(img)
4  import cv2
5  from PIL import Image
6  import io
7  import numpy as np
8  from keras.models import load_model
9  model2=load_model('FinalModelGarbageClassification.h5')
10 def perform_model_prediction(img):
11     img = Image.open(io.BytesIO(img))
12     img = img.resize((224, 224))
13     img=np.expand_dims(img,axis=0)
14     out=['cardboard','e-waste','glass','medical','metal','organic','paper', 'plastic']
15     predictions = model2.predict(img)
16     prediction_result = out[np.argmax(predictions)]
17     return prediction_result
```

Figure 8.14: Prediction code snippet Garbage classification

```
# Open an image file
image_path = "./69551_model_1__79366.jpg"
original_image = Image.open(image_path)

# Resize the image to 224x224
resized_image = original_image.resize((224, 224))

# Save the resized image
out= ['externalcovers', 'gloves', 'mask', 'medicines', 'syringe']
img=np.expand_dims(resized_image,axis=0)


# Make predictions
predictions = model2.predict(img)
print(out[np.argmax(predictions)])
```

```
1/1 [==============================] - 0s 25ms/step
externalcovers
```

Figure 8.15: Prediction code snippet Medical waste classification

9. Comparison of Tech Stacks

- While both stacks likely require scalability considerations, Stack 1 might have fewer complexities related to object storage scalability, making it potentially more straightforward to scale.
- On the other hand, MinIO is designed for efficient object storage (operates on a bucket), making it well-suited for handling unstructured and large volumes of data such as images, videos, and documents, unlike MySQL. It is also compatible with the S3 API, making it interoperable with applications designed for Amazon S3.This makes it a better choice of Data Lake used in Tech Stack 2.
- The hybrid use of PostgreSQL and MongoDB as data warehouse in Stack 1 provides a flexible and efficient solution, accommodating diverse data requirements and supporting applications with varied data structures. This approach is particularly beneficial when dealing with applications that demand both relational and NoSQL database features within a unified data storage and analytics framework as opposed having only PostgreSQL, although it supports complex queries and analytics.

In light of the observed advantages and strengths of MinIO as the Data Lake and the hybrid PostgreSQL-MongoDB warehouse configuration, it becomes evident that a new pipeline incorporating MinIO as the data lake and the hybrid warehouse will yield a more robust and optimized outcome for the system as a future work.

10. Conclusion

The incorporation of big data analytics in this project plays a pivotal role in transforming traditional waste management practices. By harnessing the power of advanced analytics, the project aims to optimize various facets of waste collection and disposal. Two distinct technology stacks were implemented in this project to compare the performance using different data pipeline.

The first stack utilized MySQL as the Data Lake, employing Python scripts for data ingestion and preprocessing. The processed data was then stored in a hybrid Data Warehouse consisting of PostgreSQL and MongoDB, providing a flexible and efficient solution for diverse data types. The second stack leveraged MinIO as the Data Lake, emphasizing high-performance object storage, and integrated Postgres as the Data Warehouse. The Visualizations and trend analyses provided by Tableau and plotly facilitated informed decision-making to optimize waste collection based on fill levels of bins and other critical parameters. Furthermore, waste classification using deep learning model (MobileNet V2) achieved a robust performance in classifying the general and medical wastes.

In essence, this project leverages big data analytics principles and technologies to handle, process, and derive valuable insights from the large and diverse datasets associated with waste management and address the goal of sustainable waste management and classification.

11. References

[1] T. Vafeiadis et al., "Data Analytics Platform for the Optimization of Waste Management Procedures," 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini, Greece, 2019, pp. 333-338, doi: 10.1109/DCOSS.2019.00074.

[2] Thaseen Ikram, S.; Mohanraj, V.; Ramachandran, S.; Balakrishnan, A.” An Intelligent Waste Management Application Using IoT and a Genetic Algorithm–Fuzzy Inference System”. Appl. Sci. 2023, 13, 3943. https://doi.org/10.3390/app13063943

[3] Z. Hisham Che Soh, M. Azeer Al-Hami Husa, S. Afzal Che Abdullah and M. Affandi Shafie, "Smart Waste Collection Monitoring and Alert System via IoT," 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Malaysia, 2019, pp. 50-54, doi: 10.1109/ISCAIE.2019.8743746.

[4] Cheema, S.M.; Hannan, A.; Pires, I.M. “Smart Waste Management and Classification Systems Using Cutting Edge Approach”. Sustainability 2022, 14, 10226. https://doi.org/10.3390/su141610226

[5] E. Al-Masri, I. Diabate, R. Jain, M. H. Lam and S. Reddy Nathala, "Recycle.io: An IoT-Enabled Framework for Urban Waste Management," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 5285-5287, doi: 10.1109/BigData.2018.8622117.

[6]Jobin Joseph et al.,”Smart Waste Management Using Deep Learning with IoT”, International Journal of Networks and Systems, 8(3), April - May 2019, 37 - 40.

[7] Kulkarni, Hrushikesh N. et al., “Waste Object Detection and Classification.” (2019)

[8] Md. Wahidur Rahman et al., “Intelligent waste management system using deep learning with IoT”, Journal of King Saud University - Computer and Information Sciences, 34(5), 2022, Pages 2072-2087, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2020.08.016.

[9] Hussain et al., “Waste Management and Prediction of Air Pollutants Using IoT and Machine Learning Approach”. (2020). Energies. 3930. 10.3390/en13153930.

[10] Mookkaiah, Senthil Sivakumar et al. “Design and development of smart Internet of Things-based solid waste management system using computer vision.” Environmental science and pollution research international vol. 29,43 (2022): 64871-64885. doi:10.1007/s11356-022-20428-2

[11] Zhou, H., Yu, X., Alhaskawi, A. et al. “A deep learning approach for medical waste classification”. Sci Rep 12, 2159 (2022). https://doi.org/10.1038/s41598-022-06146-2

[12] Yang Z, Xia Z, Yang G, Lv Y.”A Garbage Classification Method Based on a Small Convolution Neural Network”. Sustainability. 2022; 14(22):14735. https://doi.org/10.3390/su142214735

.