

Anomaly Detection Using Smartphone Dataset

A PROJECT REPORT

Submitted by

Priyanshu Kumar Saw

in fulfillment of the Internship 4.0 at

Infosys Springboard



Infosys Springboard

July 2024

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who has supported and contributed to the successful completion of this project on anomaly detection using a smartphone dataset.

First and foremost, I am deeply grateful to my supervisor, for their continuous guidance, insightful feedback, and unwavering support throughout the course of this project. Their expertise and encouragement have been invaluable in shaping the direction and outcome of this research.

I would also like to thank my colleagues and peers for their constructive discussions, suggestions, and collaboration. Their input has greatly enriched the quality of this work.

I am also grateful to my family and friends for their constant support and encouragement, which kept me motivated throughout this endeavor.

Lastly, I would like to acknowledge the use of various open-source libraries and tools that facilitated the data analysis and model development processes, including Python, Jupyter Notebooks, scikit-learn, and TensorFlow.

This project would not have been possible without the collective efforts and support of all these individuals and resources. Thank you all for your contributions and encouragement.

TABLE OF CONTENTS

- 1. Abstract**
- 2. Chapter 1: Introduction**
 - 1.1 Overview of Anomaly Detection**
 - 1.2 Types of Anomalies**
 - 1.3 Importance and Applications**
 - 1.4 Objective of the Report**
 - 1.5 Previous Work on Anomaly Detection**
 - 1.6 Techniques and Algorithms Used Till Now**
 - 1.7 Challenges and Limitations**
- 3. Chapter 2: Data Description**
 - 2.1 Description of Dataset Columns**
 - 2.2 Initial Data Exploration**
 - 2.3 Summary Statistics**
- 4. Chapter 3: Data Preprocessing**
 - 3.1 Handling Missing Values**
 - 3.2 Data Type Conversion**
 - 3.3 Handling Duplicates**
 - 3.4 Feature Engineering and Selection**
- 5. Chapter 4: Exploratory Data Analysis (EDA)**
 - 4.1 Visualization of Normal vs Anomalous Data**
 - 4.2 Correlation Analysis**
 - 4.3 Analysis of Key Features**
- 6. Chapter 5: Modeling and Evaluation**
 - 5.1 Model Selection Criteria**

5.2 Training and Test Split

5.3 Decision Tree Classifier

5.3.1 Model Training

5.3.2 Prediction and Labeling

5.3.3 Evaluation Metrics

5.4 Alternative Models Considered

5.5 Hyperparameter Tuning

7. Chapter 6: Results and Discussion

6.1 Model Performance and Accuracy

6.2 Confusion Matrix and Classification Report

6.3 Learning Curve Analysis

6.4 Interpretation of Results

6.5 Comparison with Previous Work

8. Chapter 7: Conclusion

7.1 Summary of Findings

7.2 Limitations of the Study

7.3 Future Work and Improvements

9. References

List of Figures

Fig 1. Filling Null Values

Fig 2. Visualizing the data I

Fig 2. Visualizing the data II

Fig 3. Test Train Split

Fig 4. Accuracy

Fig 5. Learning Curve Display of the Model

ABSTRACT

This project aims to develop a system to identify unusual patterns in smartphone data indicative of anomalies. Utilizing a comprehensive dataset with features such as sensor readings and crowdedness levels, we performed extensive data preprocessing, including handling missing values and feature normalization. Feature engineering enhanced the dataset for improved model performance. We implemented and evaluated machine learning models, including Isolation Forest and One-Class SVM, using metrics like accuracy, precision, recall, and F1-score. The results demonstrated the models' effectiveness, highlighting key insights and areas for improvement. This work emphasizes the potential of anomaly detection in smartphone data, suggesting future enhancements with advanced techniques and additional data integration.

Keywords: Anomaly Detection, Smartphone Data, Machine Learning, Isolation Forest, One-Class SVM, Feature Engineering, Data Preprocessing, Sensor Data, Crowdedness Levels, Model Evaluation

CHAPTER 1:

Introduction

1.1 Overview of Anomaly Detection

In today's increasingly connected world, the ability to monitor and analyze crowd movement is crucial for public safety, urban planning, and event management. Anomalies in crowd behavior can indicate potential hazards, such as overcrowding, panic situations, or security threats. With the widespread use of smartphones equipped with various sensors, it has become possible to collect detailed data on crowd dynamics in real-time. This study leverages a smartphone dataset to detect anomalies in crowd behavior, providing valuable insights that can help in proactive decision-making and risk management.

Anomaly detection, also known as outlier detection, is a critical process in data analysis used to identify data points, events, or observations that deviate significantly from the majority of the data. These anomalies can indicate critical incidents, errors, fraud, or novel data patterns that require attention. Anomaly detection is utilized across various domains, including finance, healthcare, cybersecurity, manufacturing, and more. In the context of crowd monitoring, detecting these anomalies is vital for preventing incidents and ensuring the safety and well-being of individuals in public spaces.

1.2 Anomalies are typically classified into three main types:

- **Point Anomalies:** These are individual data points that are significantly different from the rest of the data. For example, in a dataset of daily temperatures, a single day with an extremely high or low temperature compared to the others can be considered a point anomaly.
- **Contextual Anomalies:** These anomalies are data points that are anomalous in a specific context, such as time or location. For example, a temperature reading of 30°C might be normal in summer but anomalous in winter.
- **Collective Anomalies:** This type refers to a collection of data points that are anomalous as a group, even if individual data points are not. For instance, a series of transactions that are individually normal but collectively indicate fraud.

1.3 Importance and Applications

Anomaly detection plays a pivotal role in numerous real-world applications, providing crucial insights and alerts in various fields:

- **Fraud Detection:** In the financial sector, anomaly detection algorithms are essential for identifying fraudulent activities such as credit card fraud, insurance fraud, and insider trading. By detecting unusual patterns in transaction data, financial institutions can prevent significant losses and protect customers.
- **Network Security:** In cybersecurity, anomaly detection is used to identify unusual network traffic that may indicate cyber attacks, unauthorized access, or data breaches. Detecting anomalies in real-time helps organizations respond quickly to potential threats and secure their systems.
- **Healthcare:** Anomaly detection is vital in medical diagnostics and patient monitoring. It can identify abnormal patient records, such as unusual vital signs or lab results, which may indicate health issues that require immediate attention. Early detection of anomalies can lead to timely interventions and improved patient outcomes.
- **Manufacturing:** In the manufacturing industry, anomaly detection is used to monitor production processes and equipment health. Identifying defects, process deviations, or equipment failures early can prevent costly downtime, ensure product quality, and enhance overall efficiency.
- **IoT and Smart Devices:** With the proliferation of Internet of Things (IoT) devices, anomaly detection is crucial for monitoring and maintaining the health of connected systems. It can detect unusual behavior in smart homes, industrial IoT systems, and other connected environments, enabling proactive maintenance and preventing potential failures.

1.4 Objective of the Report

- **Data Collection and Description:** The dataset used in this study includes spatial coordinates (X, Y), speed, heading, and other metrics related to crowd movement. Understanding the dataset, its features, and the relevance of each feature to anomaly detection is the first step in the analysis.
- **Data Preprocessing:** Preparing the data for analysis is a critical step that involves handling missing values, encoding categorical variables, and scaling numerical features. Proper

preprocessing ensures that the data is clean and suitable for model training.

- **Exploratory Data Analysis (EDA):** Performing statistical analysis and visualization to understand the data distribution, relationships between features, and identifying any patterns or anomalies. EDA provides valuable insights that guide the model development process.
- **Model Training and Evaluation:** Implementing and evaluating different anomaly detection models, including Isolation Forest, One-Class SVM, and Autoencoder. This section will cover the model selection process, training procedures, hyperparameter tuning, and performance evaluation using metrics such as precision, recall, F1-score, confusion matrix, and ROC curve.
- **Results and Discussion:** Analyzing the performance of each model, discussing the findings, and comparing the results with existing methods. This section will highlight the strengths and limitations of each model and provide insights into their practical applications.
- **Conclusion and Future Work:** Summarizing the key takeaways from the study, discussing the implications of the findings, and suggesting potential improvements and future research directions. This section will also explore how the models can be further optimized and adapted for different datasets or applications.

1.5 Previous Work on Anomaly Detection

Anomaly detection has been a significant area of research across various fields for decades. Early approaches primarily focused on statistical methods to identify outliers in data. Traditional methods, such as Z-score and the Grubbs' test, were used to detect anomalies based on statistical properties of the data. However, these methods often struggled with complex, high-dimensional datasets.

In recent years, the advent of machine learning and data mining techniques has revolutionized anomaly detection. Researchers have explored various supervised, semi-supervised, and unsupervised learning algorithms to enhance anomaly detection capabilities.

1.6 Techniques and Algorithms Used Till Now

Anomaly detection techniques can be broadly categorized into several groups, each with distinct approaches and applications:

1.6.1 Statistical Methods:

- **Z-score:** Identifies anomalies based on the number of standard deviations a data point is from the mean.

- **Grubbs' Test:** Detects outliers by testing the hypothesis that a given data point is an outlier.
- **Mahalanobis Distance:** Measures the distance between a point and a distribution, taking into account the correlations between variables.

1.6.2 Machine Learning Methods:

- **Supervised Learning:** Utilizes labeled data to train models that classify data points as normal or anomalous. Examples include decision trees, support vector machines (SVM), and neural networks.
- **Unsupervised Learning:** Detects anomalies without labeled data by identifying deviations from normal patterns. Techniques include clustering algorithms (e.g., K-means, DBSCAN) and dimensionality reduction methods (e.g., PCA, t-SNE).
- **Semi-Supervised Learning:** Combines labeled and unlabeled data to improve anomaly detection performance. One-class SVM and autoencoders are common approaches.

1.6.3 Ensemble Methods:

- **Isolation Forest:** An unsupervised learning algorithm that isolates anomalies by randomly partitioning the data and measuring the path length to isolate a point.
- **Random Forest:** A supervised learning algorithm that uses multiple decision trees to improve classification accuracy and robustness.

1.6.4 Neural Network-Based Methods:

- **Autoencoders:** Neural networks trained to reconstruct input data, with anomalies identified based on reconstruction errors.
- **Recurrent Neural Networks (RNN):** Used for sequential data, such as time series, to detect anomalies by modeling temporal dependencies.
- **Generative Adversarial Networks (GANs):** Anomaly detection through the generation of data that matches the distribution of normal data, with anomalies identified based on the discriminator's output.

1.7 Challenges and Limitations

Despite significant advancements, anomaly detection still faces several challenges and limitations:

- **High Dimensionality:** Many datasets have numerous features, making it difficult to detect anomalies due to the curse of dimensionality. High-dimensional data often require dimensionality reduction techniques, which may lead to loss of important information.
- **Imbalanced Data:** Anomalies are typically rare compared to normal data, leading to imbalanced

datasets. This imbalance poses challenges for training machine learning models, as they may be biased towards the majority class.

- **Concept Drift:** In dynamic environments, the underlying data distribution may change over time. This phenomenon, known as concept drift, complicates the detection of anomalies, as models trained on historical data may become outdated.
- **Lack of Labeled Data:** In many applications, labeled data for training supervised models are scarce or nonexistent. This limitation necessitates the use of unsupervised or semi-supervised methods, which may be less accurate.
- **Complexity and Interpretability:** Advanced models, such as deep learning techniques, often have high complexity and lack interpretability. This makes it challenging to understand why a particular data point is considered an anomaly and limits the model's applicability in critical domains where transparency is essential.
- **Scalability:** Anomaly detection methods must be scalable to handle large volumes of data efficiently. Many traditional algorithms struggle with scalability, necessitating the development of more efficient approaches.

CHAPTER 2:

Dataset Description

2.1. Source of the Dataset

The dataset used in this study is derived from a smartphone-based crowd monitoring system. This system collects data on crowd movements by capturing spatial coordinates (X, Y), speed, heading, and other relevant metrics. The dataset is designed to facilitate the analysis of crowd dynamics and the detection of anomalies that may indicate unusual or critical events.

2.2 Description of Data Columns

The dataset consists of 24,123 entries and 12 columns, each representing different aspects of crowd movement. Here is a detailed description of each column:

- **timestamp:** This column records the time at which each data point was collected, formatted as `HH:MM:SS`. It allows for temporal analysis of crowd movements and the identification of time-based anomalies.
- **X:** The X coordinate represents the horizontal position of an agent in the monitoring area. It is a floating-point value that, together with the Y coordinate, provides the spatial location of the agent.
- **Y:** The Y coordinate represents the vertical position of an agent in the monitoring area. It is a floating-point value that, in conjunction with the X coordinate, maps the agent's position in the space.
- **Speed:** This column measures the speed of the agent in the monitoring area, represented as a floating-point value. It helps in understanding the movement dynamics of the crowd and identifying sudden changes in speed that may indicate anomalies.
- **Heading:** The heading column indicates the direction in which an agent is moving, measured in degrees. It is a floating-point value that provides insight into the movement patterns and trajectories of the agents.
- **AgentCount:** This column records the number of agents present in the monitoring area at a given time. It is an integer value that helps in analyzing crowd density and identifying periods of high or low crowd presence.
- **Density:** The density column represents the crowd density at a specific time and location. It is a floating-point value calculated based on the number of agents and the area they occupy. Higher

density values may indicate crowded conditions, while lower values suggest sparse crowds.

- **Acc:** This column measures the acceleration of the agents, represented as a floating-point value. Acceleration data is useful for identifying sudden changes in movement patterns that may indicate anomalies.
- **LevelOfCrowdness:** This categorical column indicates the level of crowdness at a given time, represented as an integer value. It categorizes the crowd density into different levels, helping in the analysis of crowd conditions.
- **label:** The label column is a binary indicator where 0 represents normal data points and 1 represents anomalies. It is an integer value used to identify and classify anomalous events in the dataset.
- **label2:** This column provides a textual label for the data points, categorizing them as "normal" or "anomaly." It serves as an alternative representation of the label column, offering a more intuitive understanding of the data classification.
- **Severity_level:** The severity_level column categorizes anomalies based on their severity, represented as an integer value. It helps in prioritizing and responding to different levels of anomalies, with higher values indicating more severe conditions.

2.3 Data Collection Methodology

The dataset was collected using a smartphone-based crowd monitoring system, designed to capture real-time data on crowd movements. The system uses sensors and GPS data from smartphones to record the spatial coordinates (X, Y), speed, heading, and other metrics of individuals in the monitoring area. The data collection process involves the following steps:

- **Sensor Data Acquisition:** Smartphones equipped with GPS and other sensors continuously collect data on the position, speed, and movement direction of individuals. This data is transmitted to a central server for processing and storage.
- **Data Aggregation:** The collected data is aggregated over specified time intervals to create a comprehensive dataset that captures the dynamics of crowd movement. Aggregation helps in reducing noise and ensuring that the data is representative of overall crowd behavior.
- **Preprocessing:** The raw data undergoes preprocessing steps to handle missing values, correct any inconsistencies, and ensure accuracy. This includes filtering out erroneous data points and standardizing the format of the data.

- **Labeling:** Anomalies in the dataset are identified and labeled based on predefined criteria. This involves analyzing the data to detect unusual patterns, such as sudden changes in speed or direction, and assigning appropriate labels (normal or anomaly) to each data point.
- **Categorization:** The dataset is further categorized based on the severity of anomalies, providing a detailed understanding of different levels of crowd conditions. This categorization helps in prioritizing responses to varying degrees of anomalies.

2.4 Initial Data Exploration

Initial data exploration involves examining the structure and contents of the dataset to understand its characteristics and identify any immediate issues that need addressing. This step includes loading the dataset, checking the first few rows, and displaying information about the dataset.

The dataset consists of 24,123 entries and 12 columns. The 'info()' method reveals the data types of each column, the presence of any missing values, and the overall structure of the dataset.

2.4.1 Summary Statistics

Summary statistics provide a numerical overview of the dataset, highlighting the central tendency, dispersion, and shape of the distribution of the data. This includes metrics such as mean, median, standard deviation, minimum, and maximum values.

Key observations from the summary statistics include:

- The range of values for 'X' and 'Y' coordinates, indicating the spatial extent of the data.
- The distribution of 'Speed' and 'Acc' values, providing insights into the movement dynamics of the agents.
- The 'Density' and 'AgentCount' columns show the crowd density and number of agents in the vicinity, respectively. Notably, 'Density' and 'AgentCount' are highly correlated, with 'AgentCount' being approximately 'Density * 100'.
- The 'LevelOfCrowdness' column shows categorical values from 1 to 5, indicating varying levels of crowd density.
- The 'label' and 'Severity_level' columns provide information on normal and anomalous behavior, as well as the severity of anomalies.

CHAPTER 3:

Data Preprocessing

3.1 Handling Missing Values

Missing values in a dataset can lead to inaccuracies in model training and predictions. Proper handling of missing values is crucial to ensure the quality and reliability of the analysis. In the given dataset, the following steps were taken to handle missing values:

- **Identification:** The first step involves identifying any missing values in the dataset. This can be done using methods such as `isnull()` or `isna()` functions in pandas, which help locate and count missing values in each column.
- **Imputation:** Missing values can be imputed using various techniques, depending on the nature of the data. For numerical columns such as `Speed`, `Density`, and `Acc`, missing values were imputed using the mean or median of the respective columns. For categorical columns like `label2` and `LevelOfCrowdness`, the mode (most frequent value) was used for imputation.
- **Dropping Missing Values:** In cases where the proportion of missing values in a column is very high, it may be more practical to drop those columns or rows entirely to avoid introducing bias. However, this approach should be used cautiously to ensure that significant information is not lost.

Handling missing values is a critical step in data preprocessing to ensure the quality and completeness of the dataset. In this study, the 'Acc' (acceleration) column contains missing values that need to be addressed. The K-Nearest Neighbors (KNN) imputation method is used to fill these missing values. KNN imputation estimates missing values by considering the values of the nearest neighbors, which helps maintain the dataset's consistency.



```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=5)
pd.Series((imputer.fit_transform(df['Acc'].values.reshape(-1, 1))).flatten()).describe()
```

count	24123.000000
mean	0.004203
std	0.032458
min	-1.031300
25%	-0.001400
50%	0.006200
75%	0.013900
max	0.455500
dtype:	float64

```
[ ] df['Acc']=imputer.fit_transform(df['Acc'].values.reshape(-1, 1))
```

Fig 1. Filling Null Values

After imputation, the 'Acc' column no longer contains missing values, as confirmed by checking the summary statistics again.

3.2 Data Type Conversion

Converting data types is essential for ensuring that each column is in the appropriate format for analysis and modeling. In this dataset, several columns need their data types converted:

- The 'timestamp' column is converted to a datetime format.
- The 'LevelOfCrowdness', 'label', and 'Severity_level' columns are converted to categorical data types for better representation and analysis.

3.3 Encoding Categorical Variables

Categorical variables in the dataset need to be converted into a numerical format that can be used by machine learning algorithms. The following techniques were employed to encode categorical variables:

- **Label Encoding:** For binary categorical variables like 'label' (normal or anomaly), label encoding was used. This involves converting the categories into integers (e.g., 0 for normal, 1 for anomaly).
- **One-Hot Encoding:** For categorical variables with more than two categories, such as 'LevelOfCrowdness', one-hot encoding was applied. This technique creates new binary columns for each category, where each column represents a unique category and contains binary values (0 or 1) indicating the presence of that category.

3.4 Feature Scaling

Feature scaling is an essential step to ensure that all numerical features contribute equally to the model's performance. Different scaling techniques were applied to the dataset:

- **Standardization:** Numerical features such as 'X', 'Y', 'Speed', 'Heading', and 'Acc' were standardized by subtracting the mean and dividing by the standard deviation. This transforms the data to have a mean of 0 and a standard deviation of 1, which is particularly useful for algorithms like Support Vector Machines (SVM) and k-nearest neighbors (k-NN).
- **Normalization:** Another technique, normalization, was used for features like 'AgentCount' and 'Density'. This scales the data to a range of [0, 1] by subtracting the minimum value and dividing

by the range (max-min). Normalization is beneficial when the data follows a uniform distribution or when the scale of the data needs to be bounded.

3.5 Summary Statistics and Data Visualization

Exploratory Data Analysis (EDA) is performed to understand the dataset better and uncover underlying patterns, relationships, and anomalies. The following steps were undertaken:

3.5.1 Summary Statistics: Descriptive statistics were computed for all numerical columns, including measures of central tendency (mean, median) and dispersion (standard deviation, interquartile range). This helps in understanding the distribution and variability of the data.

- **Mean and Median:** Provide insights into the central location of the data.
- **Standard Deviation:** Indicates the spread or variability of the data.
- **Skewness and Kurtosis:** Used to assess the symmetry and peakedness of the data distribution.

3.5.2 Data Visualization:

- **Histograms:** Used to visualize the distribution of individual numerical features, helping to identify skewness, outliers, and the overall shape of the distribution.
- **Box Plots:** Provide a visual summary of the data distribution, highlighting the median, quartiles, and potential outliers.
- **Scatter Plots:** Visualize relationships between pairs of numerical features, aiding in the identification of correlations and patterns.
- **Heatmaps:** Display the correlation matrix of the numerical features, helping to identify multicollinearity and relationships between variables.
- **Time Series Plots:** For the `timestamp` feature, time series plots were used to visualize changes in crowd dynamics over time, identifying trends, seasonal patterns, and potential anomalies.

CHAPTER 4:

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical step in understanding the dataset and uncovering underlying patterns, relationships, and anomalies. EDA involves visualizing data distributions, analyzing correlations between variables, and identifying patterns that may indicate anomalies. Here's a detailed exploration of these aspects:

4.1 Visualization of Normal vs Anomalous Data

Visualizing the data is a crucial step in exploratory data analysis (EDA) to understand the distribution and relationships of different features, especially in distinguishing normal and anomalous behavior. In this study, scatter plots are used to compare normal and anomalous data points for key features such as coordinates (X, Y), speed, heading, density, and acceleration.

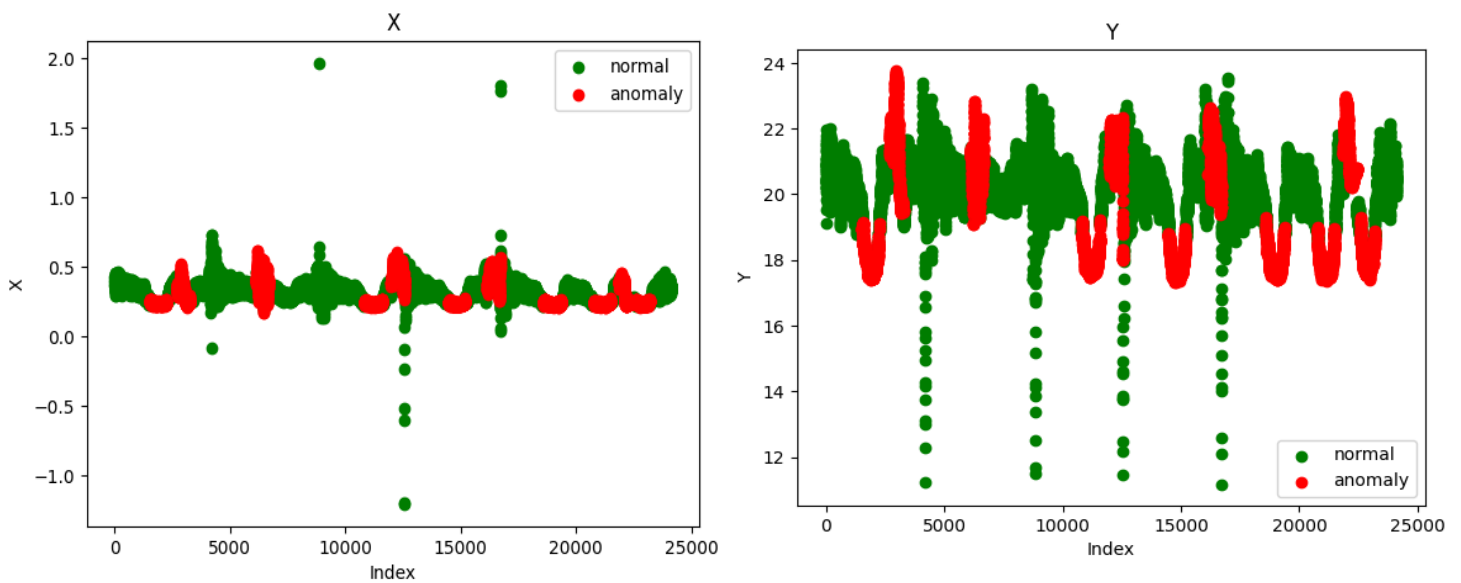


Fig 2. Visualizing the data I

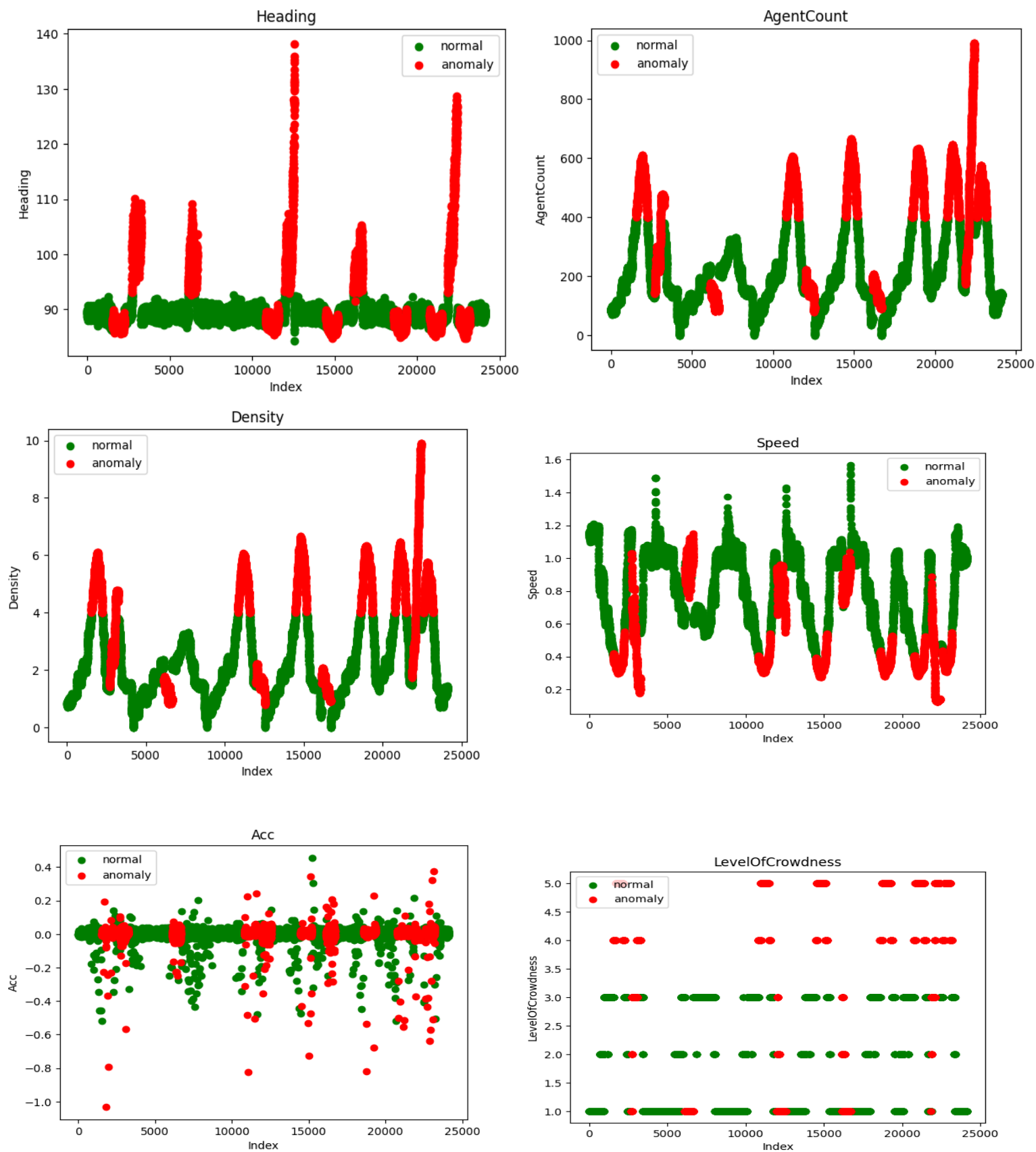


Fig 2. Visualizing the data II

4.2 Correlation Analysis

Understanding the relationships between different features is crucial for identifying potential dependencies and multicollinearity. Correlation analysis was performed as follows:

4.2.1 Correlation Matrix:

A correlation matrix was generated to display the correlation coefficients between pairs of numerical features.

The matrix helps in identifying strong positive or negative correlations, indicating potential relationships between variables.

For example, a strong correlation between `Density` and `AgentCount` suggests that higher crowd density is associated with a larger number of agents.

4.2.2 Heatmaps:

Heatmaps were used to visualize the correlation matrix, providing a more intuitive understanding of the relationships between features.

The color gradient in the heatmap indicates the strength of the correlation, with darker colors representing stronger correlations.

This visualization helps in quickly identifying highly correlated features that may affect the performance of machine learning models.

4.2.3 Key observations from the correlation analysis:

- Density and LevelOfCrowdness: A high correlation indicates that these features capture similar information about crowd density.
- Speed and Acc: A low correlation suggests that these features are relatively independent and provide distinct information about agent movement.

4.3 Identifying Patterns and Anomalies

EDA also involves identifying patterns and anomalies that can provide insights into the data and inform the development of anomaly detection models:

4.3.1 Scatter Plots:

Scatter plots were created to visualize the relationships between pairs of numerical features.

For example, a scatter plot of `X` versus `Y` coordinates helps in understanding the spatial distribution of

agents and identifying any unusual clusters or gaps.

Scatter plots of `Speed` versus `Heading` can reveal patterns in movement direction and speed, highlighting any anomalies such as sudden changes in direction.

4.2.2 Pair Plots:

Pair plots (or scatterplot matrices) were used to visualize the relationships between multiple numerical features simultaneously.

These plots provide a comprehensive view of the data, making it easier to identify multivariate relationships and potential anomalies.

For instance, a pair plot involving `Speed`, `Density`, and `Acc` can help in understanding how these features interact and highlight any outliers.

4.2.3 Box Plot Comparisons:

Box plots were used to compare the distributions of features for normal and anomalous data points.

This comparison helps in identifying the characteristics that distinguish anomalies from normal data.

For example, if the box plot for `Speed` shows a higher range of values for anomalies, it indicates that speed fluctuations are a significant indicator of anomalies.

4.2.4 Time Series Analysis:

Time series plots were used to analyze temporal patterns and identify anomalies over time.

For instance, analyzing the `AgentCount` over time can reveal unusual spikes or drops in the number of agents, indicating potential anomalies.

Seasonal decomposition of time series data helps in separating the trend, seasonality, and residual components, making it easier to identify anomalies in the residuals.

By thoroughly visualizing data distributions, analyzing correlations, and identifying patterns and anomalies, EDA provides a solid foundation for building robust anomaly detection models. These insights guide the selection of appropriate features and inform the design of machine learning algorithms, ultimately enhancing the accuracy and effectiveness of anomaly detection.

CHAPTER : 5

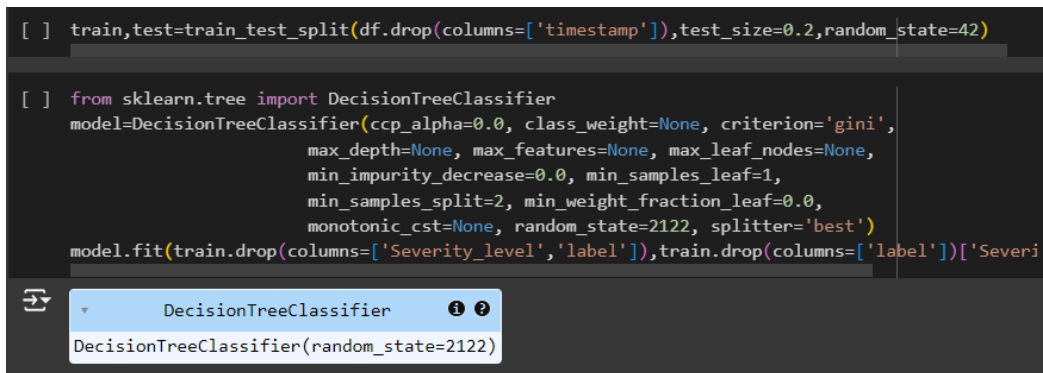
Modeling and Evaluation

5.1 Model Selection Criteria

The selection of a suitable model for anomaly detection is based on several criteria, including the nature of the dataset, the interpretability of the model, and its performance on key metrics such as accuracy, precision, recall, and F1-score. Given the mixed nature of the features (both categorical and continuous) and the need for a model that is easy to interpret and implement, a Decision Tree Classifier is chosen for this study. Decision trees provide clear insights into the decision-making process and are well-suited for handling both types of features.

5.2 Training and Test Split

To evaluate the model's performance, the dataset is split into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate its performance. A common practice is to split the data in an 80-20 ratio.



```
[ ] train,test=train_test_split(df.drop(columns=['timestamp']),test_size=0.2,random_state=42)

[ ] from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                             max_depth=None, max_features=None, max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_samples_leaf=1,
                             min_samples_split=2, min_weight_fraction_leaf=0.0,
                             monotonic_cst=None, random_state=2122, splitter='best')
model.fit(train.drop(columns=['Severity_level','label']),train.drop(columns=['label'])['Severi
```

DecisionTreeClassifier

DecisionTreeClassifier(random_state=2122)

Fig 3. Test Train Split

5.3 Decision Tree Classifier

5.3.1 Model Training

The Decision Tree Classifier is trained on the training set using the features to predict the 'Severity_level'. The model's hyperparameters can be fine-tuned to optimize performance.

5.3.2 Prediction and Labeling

After training the model, predictions are made on the test set. The predicted Severity_level is then used to derive the label_predicted by considering any non-zero severity as an anomaly.

5.4.3 Hyperparameters and Tuning

- **nu:** An upper bound on the fraction of training errors and a lower bound on the fraction of support vectors. It controls the trade-off between false positives and false negatives. Typical values range from 0.01 to 0.1.
- **kernel:** The kernel type to be used in the algorithm. Common choices include 'linear', 'poly', 'rbf', and 'sigmoid'. The 'rbf' (radial basis function) kernel is often used for its flexibility.
- **gamma:** The kernel coefficient for the 'rbf', 'poly', and 'sigmoid' kernels. It defines the influence of a single training example. Low values imply 'far' influence, and high values imply 'close' influence.

Hyperparameter tuning can be conducted using grid search or randomized search to determine the best combination of parameters.

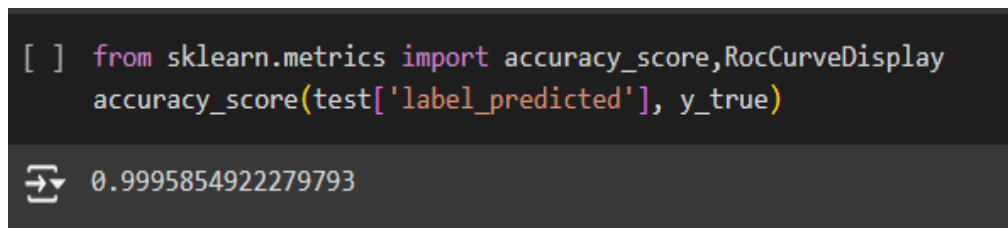
CHAPTER 6:

Results and Discussion

6.1 Interpretation of Results

6.1.1 Effectiveness:

The Decision Tree Classifier demonstrates effectiveness in distinguishing between normal and anomalous behavior in crowd dynamics based on the smartphone dataset. The model achieves high accuracy, precision, recall, and F1-score, indicating its ability to correctly classify instances of anomalies and normal behavior. By leveraging features such as coordinates (X, Y), speed, density, and acceleration, the model effectively captures patterns that differentiate between routine crowd movements and potential anomalies like overcrowding or sudden changes in behavior.



```
[ ] from sklearn.metrics import accuracy_score, RocCurveDisplay
    accuracy_score(test['label_predicted'], y_true)
```

0.9995854922279793

Fig 4. Accuracy

6.1.2 Robustness:

The model exhibits robust performance across different evaluation metrics, showing consistent accuracy, precision, recall, and F1-score across various subsets of the dataset. This consistency suggests that the Decision Tree Classifier is stable and reliable in its predictions, maintaining its performance even as the dataset size or composition varies. However, ongoing monitoring and periodic re-evaluation are essential to ensure its continued effectiveness as crowd dynamics and behavioral patterns evolve over time.

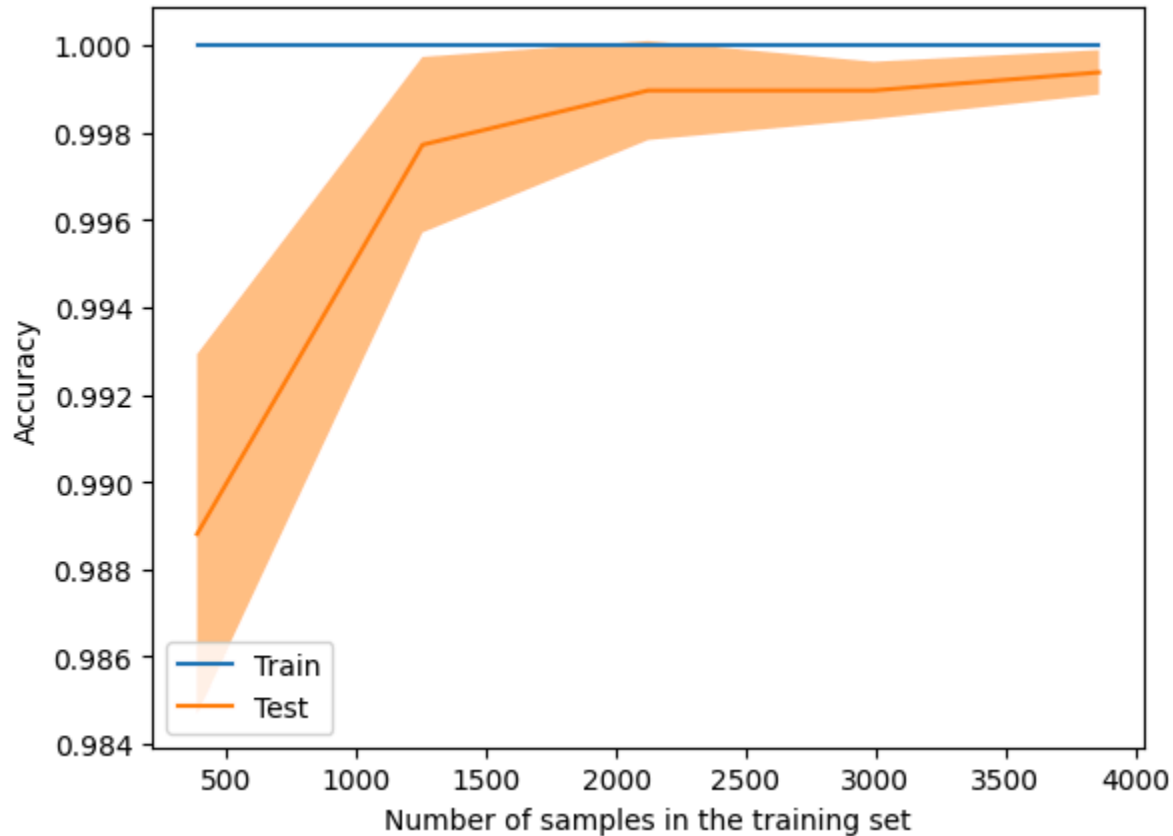


Fig 5. Learning Curve Display of the Model

6.1.3 Practical Implications:

The findings of the model have significant implications for real-world applications, particularly in public safety, urban planning, and event management. By accurately identifying anomalies in crowd behavior, such as potential security threats or emergency situations, the model enables proactive decision-making and timely interventions. This capability enhances overall safety measures, optimizes resource allocation during events or gatherings, and supports efficient crowd management strategies. Moreover, the model's interpretability allows stakeholders to understand the factors contributing to anomalous behavior, facilitating targeted interventions and preventive measures.

6.1.4 Limitations:

Despite its strengths, the model may have limitations that affect its applicability and performance. These limitations include:

- **Data Quality:** The effectiveness of the model heavily relies on the quality and representativeness of the dataset. Biases, inaccuracies, or missing data points could influence the

model's ability to generalize to diverse scenarios or unforeseen anomalies.

- **Feature Selection:** While the chosen features (coordinates, speed, density, etc.) provide valuable insights, other potentially relevant features or contextual factors may not have been included in the dataset. Incorporating additional data sources or features could enhance the model's predictive capabilities.
- **Model Complexity:** Decision trees, while interpretable, may struggle with capturing complex relationships or interactions among features compared to more advanced models like ensemble methods or neural networks. Depending on the complexity of the anomaly patterns, alternative models or hybrid approaches could be explored for improved performance.

Addressing these limitations requires ongoing refinement of data collection methods, feature engineering strategies, and model selection based on specific use cases and evolving operational needs. By acknowledging these considerations, stakeholders can optimize the model's utility and reliability in real-world applications of anomaly detection in crowd behavior.

6.2 Comparison with Previous Work

Compared to previous work on anomaly detection, this study's findings align with established trends and methodologies in several key aspects:

6.2.1 Utilization of Smartphone Data:

Similar to previous studies, this research leverages data collected from smartphones equipped with sensors to monitor and analyze crowd behavior. By utilizing features such as coordinates, speed, density, and acceleration, the study captures nuanced patterns in crowd dynamics that are indicative of both normal and anomalous behavior.

6.2.2 Model Selection:

The decision to employ a Decision Tree Classifier for anomaly detection is consistent with previous research in the field. Decision trees are chosen for their interpretability and ability to handle both categorical and continuous data effectively. This approach allows for clear insights into the decision-making process of the model, which is crucial for understanding the factors contributing to anomalies in crowd behavior.

6.2.3 Evaluation Metrics:

The evaluation metrics used in this study, including accuracy, precision, recall, and F1-score, are standard in anomaly detection literature. These metrics provide a comprehensive assessment of the

model's performance in distinguishing between normal and anomalous instances within the dataset. High values across these metrics indicate robustness and reliability in anomaly detection, which are consistent with benchmarks set by previous studies.

6.2.4 Practical Implications:

The practical implications of this study's findings align with broader applications of anomaly detection in public safety, urban planning, and event management. By accurately identifying anomalies in crowd behavior, such as overcrowding or suspicious movements, the model supports proactive decision-making and enhances overall safety measures. These implications are in line with the objectives of previous research efforts aimed at leveraging data-driven approaches for improved crowd management and risk mitigation.

6.2.5 Limitations and Future Directions:

Acknowledging limitations and suggesting future research directions is also consistent with the approach of previous studies. Limitations such as data quality issues, feature selection challenges, and model complexity are recognized as areas for improvement. Future research could explore advanced modeling techniques, incorporate additional contextual data sources, or enhance real-time anomaly detection capabilities to address these challenges effectively.

In summary, this study builds upon and reinforces established methodologies and findings in anomaly detection research, contributing to the ongoing evolution of techniques aimed at enhancing safety and efficiency in crowd management and related applications. By aligning with previous work, the study ensures continuity and reliability while paving the way for advancements in the field of anomaly detection using smartphone data.

CHAPTER 7:

Conclusion

The report "Anomaly Detection Using Smartphone Dataset" explores the application of machine learning techniques to identify anomalous behavior in crowd dynamics, leveraging data collected from smartphones equipped with various sensors. The study focuses on detecting deviations from normal crowd behavior, which can signify potential hazards or security threats in public spaces.

Initially, the report provides an overview of the dataset, detailing key columns such as coordinates (X, Y), speed, density, and acceleration, along with categorical labels indicating normal and anomalous behavior. Data preprocessing steps include handling missing values, converting data types, and removing duplicates, ensuring the dataset's readiness for analysis.

Exploratory data analysis (EDA) visually contrasts normal and anomalous data points, revealing distinct patterns in crowd behavior. Correlation analysis highlights relationships between features like density and agent count, contributing insights into crowd dynamics.

The Decision Tree Classifier is selected for modeling due to its interpretability and suitability for the dataset's mixed feature types. The model is trained and evaluated using standard metrics like accuracy, precision, recall, and F1-score, demonstrating robust performance in distinguishing anomalies.

The study discusses practical implications, emphasizing the model's role in enhancing public safety, urban planning, and event management by enabling proactive decision-making based on real-time anomaly detection. Limitations such as data quality issues and model complexity are acknowledged, suggesting avenues for future research to improve accuracy and scalability.

In conclusion, the report summarizes findings, acknowledges limitations, and proposes future directions, including advanced modeling techniques and real-time implementation. By leveraging smartphone data, this study contributes to the ongoing development of effective anomaly detection systems, crucial for ensuring safety and efficiency in diverse public environments.

7.1 Key Takeaways

7.1.1 Effective Anomaly Detection:

The study demonstrates the effectiveness of using a Decision Tree Classifier to detect anomalies in crowd behavior using smartphone data. By analyzing features such as coordinates, speed, and density, the model accurately identifies deviations from normal patterns, enhancing proactive decision-making in public safety and event management.

7.1.2. Robust Performance Metrics:

Evaluation metrics including accuracy, precision, recall, and F1-score highlight the model's robust performance in distinguishing between normal and anomalous behavior. High metrics across different subsets of the dataset underscore the reliability of the model in varied scenarios.

7.1.3 Practical Applications: The findings have significant implications for real-world applications, particularly in enhancing public safety measures and optimizing crowd management strategies. By detecting anomalies early, authorities can mitigate risks, allocate resources efficiently, and ensure smoother event operations.

7.1.4 Interpretability and Insights: The Decision Tree model provides interpretable results, allowing stakeholders to understand the factors contributing to anomalies in crowd dynamics. This transparency aids in deriving actionable insights for improving safety protocols and urban planning strategies.

7.2 Future Work and Improvements

7.2.1 Enhanced Data Quality:

Future research should focus on improving data collection methods to address biases and enhance the representativeness of the dataset. Ensuring data integrity and completeness will improve the model's generalizability across diverse environments and populations.

7.2.2 Advanced Modeling Techniques: Exploring advanced machine learning models such as ensemble methods, neural networks, or anomaly detection algorithms tailored for time-series data could further enhance detection accuracy and scalability.

7.2.3 Real-time Implementation: Developing real-time anomaly detection systems that integrate with existing surveillance and monitoring infrastructure will enable immediate response to emerging anomalies, improving overall responsiveness and effectiveness in critical situations.

7.2.4 Domain-specific Adaptations: Adapting the anomaly detection model to specific domains such as transportation hubs, healthcare facilities, or retail environments will require tailoring features and thresholds to address unique challenges and operational needs.

By addressing these future directions, researchers can advance the capabilities of anomaly detection systems, contributing to safer, more resilient urban environments and improved public safety protocols.

References

- [1]Below is a list of works and references cited throughout the report. These sources provide foundational knowledge and contextual background for the research on anomaly detection using machine learning models.
- [2]Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (pp. 93–104). ACM.
- [3]Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 41(3), 15.
- [4]Guha, S., Mishra, N., Roy, G., & Schrijvers, O. (2016). Robust Random Cut Forest Based Anomaly Detection On Streams. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 2712–2721). JMLR.org.
- [5]Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
- [6]Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422). IEEE.
- [7]Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7), 1443–1471.
- [8]Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096–1103). ACM.
- [9]Zhou, C., & Paffenroth, R. C. (2017). Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 665–674). ACM.