Waste Management Using Sensor Data and Machine Learning

By Srinjoy Roy

Google Colab Link: https://colab.research.google.com/drive/1GPWm-tHBC4RflVln0suC4Xfo0IXytQuG?usp=sharing

Introduction

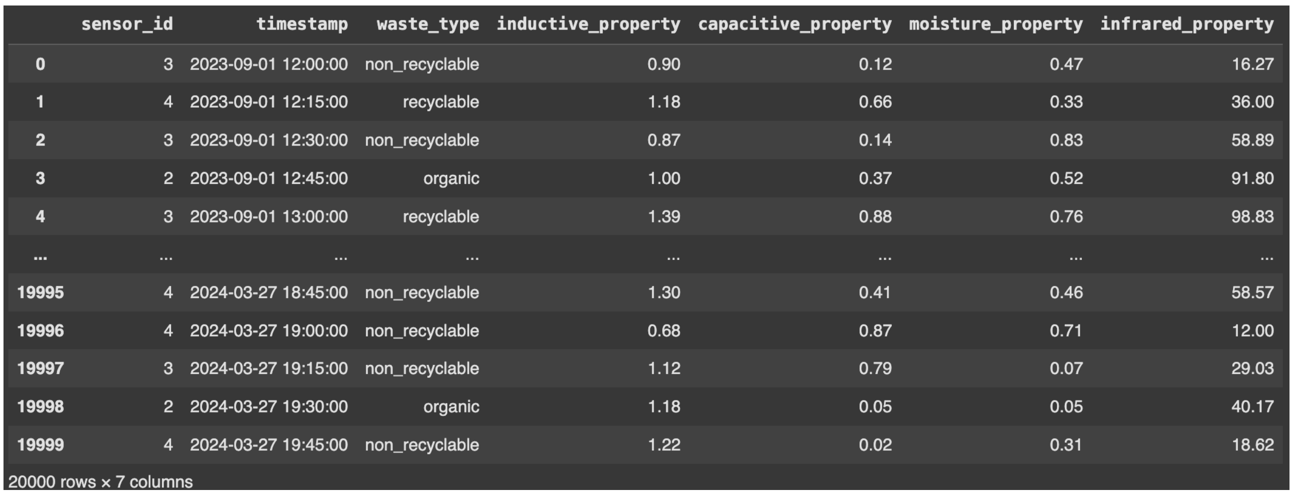
This document outlines the methodology and outcomes of a project leveraging sensor data to classify waste into categories such as organic, recyclable, and non-recyclable. By analyzing physical properties (e.g., inductive, capacitive, moisture, and infrared), the project aims to improve waste segregation processes and support sustainable waste management practices.

Dataset Overview

The dataset contains sensor data collected over time, providing measurements for various physical properties of waste materials.

Features

- **sensor_id**: Unique identifier for the sensor recording the data.
- **timestamp**: Date and time of data collection.
- **waste_type**: The type of waste material (e.g., organic, recyclable, non-recyclable).
- **inductive_property**: Sensor measurement reflecting inductive characteristics of the material.
- **capacitive_property**: Measurement of capacitive characteristics.
- **moisture_property**: Indicates the moisture content in the material.
- **infrared_property**: Infrared sensor readings indicating material properties.



	sensor_id	timestamp	waste_type	inductive_property	capacitive_property	moisture_property	infrared_property
0	3	2023-09-01 12:00:00	non_recyclable	0.90	0.12	0.47	16.27
1	4	2023-09-01 12:15:00	recyclable	1.18	0.66	0.33	36.00
2	3	2023-09-01 12:30:00	non_recyclable	0.87	0.14	0.83	58.89
3	2	2023-09-01 12:45:00	organic	1.00	0.37	0.52	91.80
4	3	2023-09-01 13:00:00	recyclable	1.39	0.88	0.76	98.83
...	...	...	...	...	...	...	...
19995	4	2024-03-27 18:45:00	non_recyclable	1.30	0.41	0.46	58.57
19996	4	2024-03-27 19:00:00	non_recyclable	0.68	0.87	0.71	12.00
19997	3	2024-03-27 19:15:00	non_recyclable	1.12	0.79	0.07	29.03
19998	2	2024-03-27 19:30:00	organic	1.18	0.05	0.05	40.17
19999	4	2024-03-27 19:45:00	non_recyclable	1.22	0.02	0.31	18.62

20000 rows × 7 columns

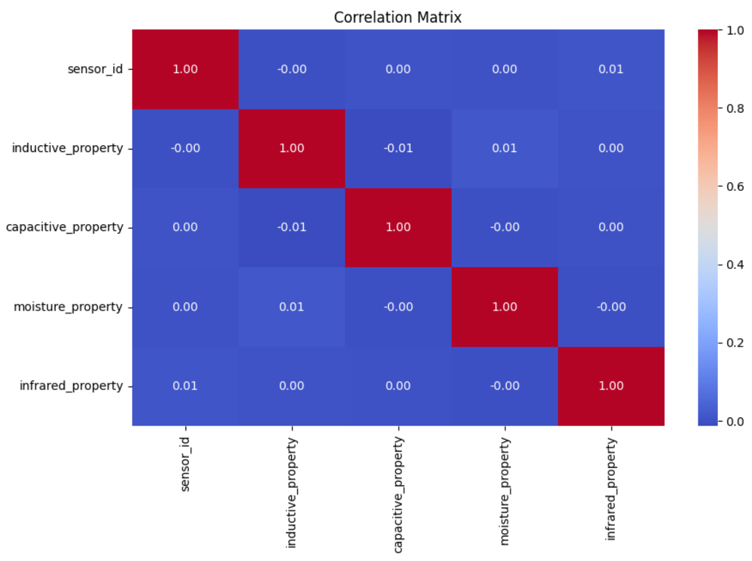
Data Exploration

1. Summary Statistics

- **Missing Values**: Checked and handled missing data.
- **Unique Waste Types**:
    - Categories: organic, recyclable, non-recyclable.
    - Distribution: Visualized using bar charts.

2. Correlation Analysis

- A correlation matrix was generated to identify relationships between numerical features, with a heatmap providing an intuitive visual representation.

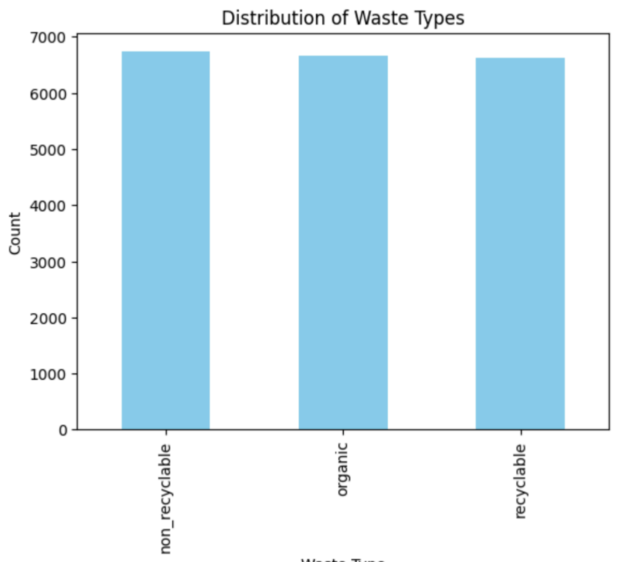


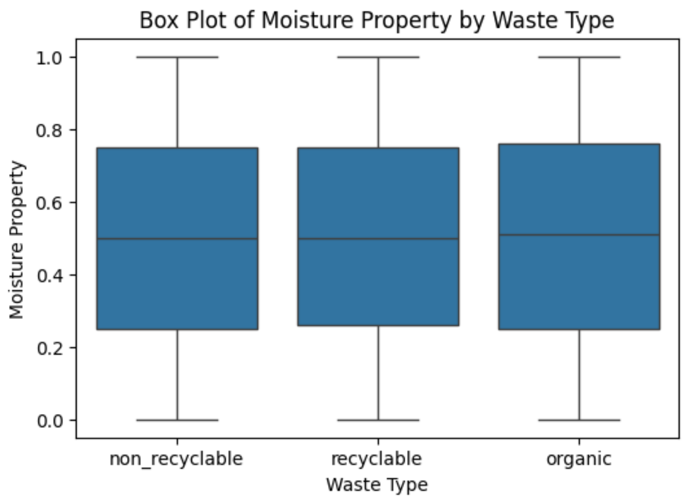
3. Feature Distribution

- Histograms and box plots were used to understand the distribution of individual features and detect outliers.

Data Preprocessing

1. **DateTime Conversion**:
    - The `timestamp` column was converted to a datetime format to extract additional time-based features (e.g., hour, day of the week).
2. **Outlier Handling**:
    - Identified and clipped extreme values using the Interquartile Range (IQR) method.



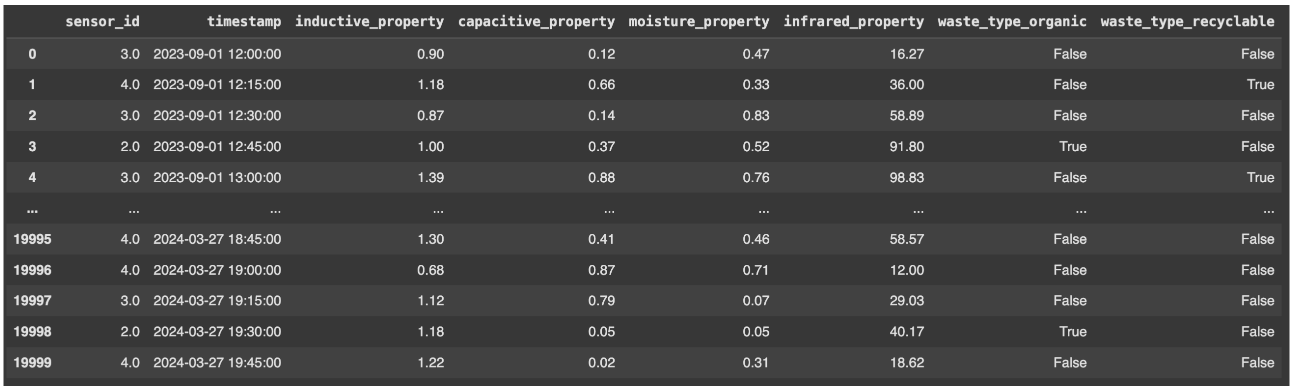


3.**Feature Engineering**:

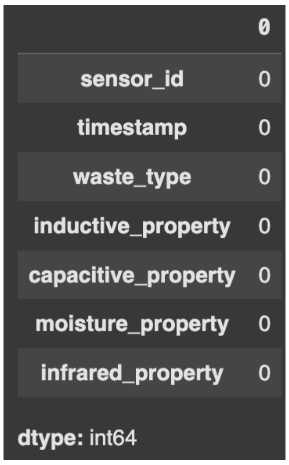
- New features were derived to capture interactions between properties:
  - **inductive_capacitive** = inductive_property × capacitive_property
  - **moisture_infrared** = moisture_property × infrared_property.

4. **Normalization**:

- Standardized numerical features to ensure uniform scaling for machine learning models.

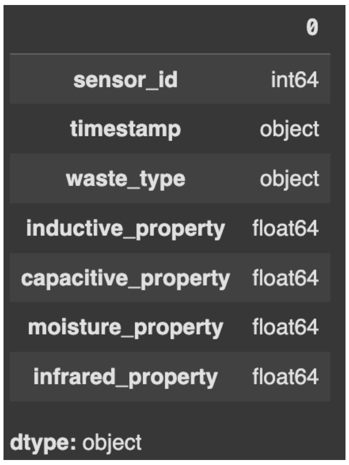


	sensor_id	timestamp	inductive_property	capacitive_property	moisture_property	infrared_property	waste_type_organic	waste_type_recyclable
0	3.0	2023-09-01 12:00:00	0.90	0.12	0.47	16.27	False	False
1	4.0	2023-09-01 12:15:00	1.18	0.66	0.33	36.00	False	True
2	3.0	2023-09-01 12:30:00	0.87	0.14	0.83	58.89	False	False
3	2.0	2023-09-01 12:45:00	1.00	0.37	0.52	91.80	True	False
4	3.0	2023-09-01 13:00:00	1.39	0.88	0.76	98.83	False	True
...	...	...	...	...	...	...	...	...
19995	4.0	2024-03-27 18:45:00	1.30	0.41	0.46	58.57	False	False
19996	4.0	2024-03-27 19:00:00	0.68	0.87	0.71	12.00	False	False
19997	3.0	2024-03-27 19:15:00	1.12	0.79	0.07	29.03	False	False
19998	2.0	2024-03-27 19:30:00	1.18	0.05	0.05	40.17	True	False
19999	4.0	2024-03-27 19:45:00	1.22	0.02	0.31	18.62	False	False



	0
sensor_id	0
timestamp	0
waste_type	0
inductive_property	0
capacitive_property	0
moisture_property	0
infrared_property	0

dtype: int64

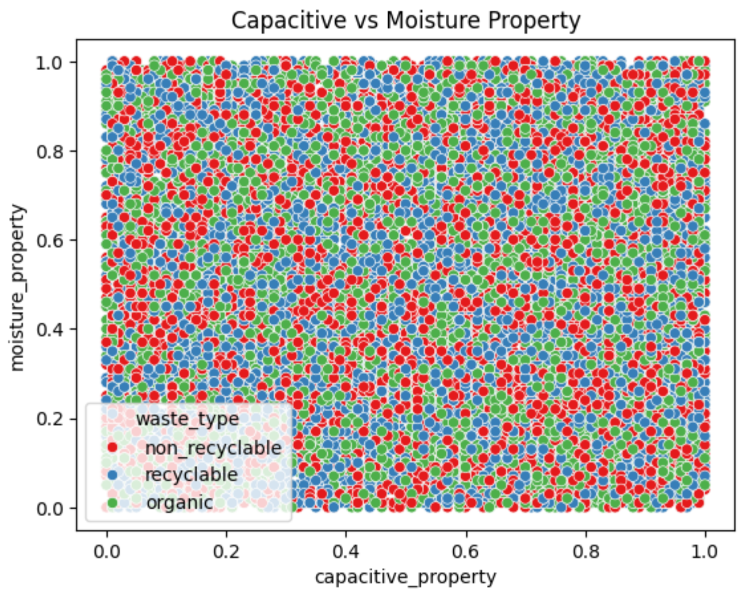


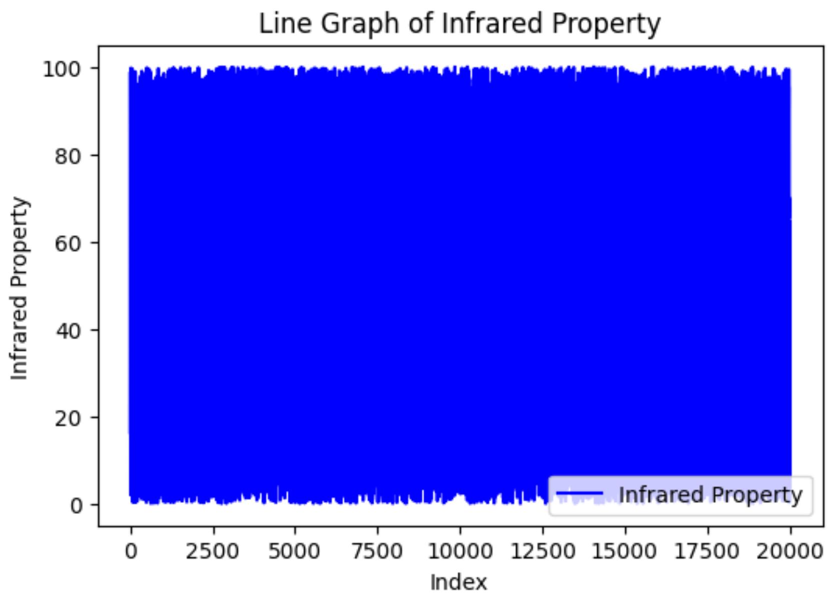
	0
sensor_id	int64
timestamp	object
waste_type	object
inductive_property	float64
capacitive_property	float64
moisture_property	float64
infrared_property	float64

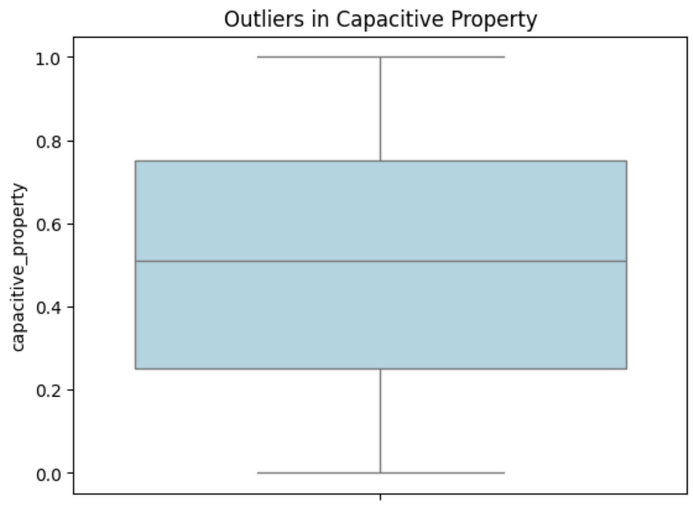
dtype: object

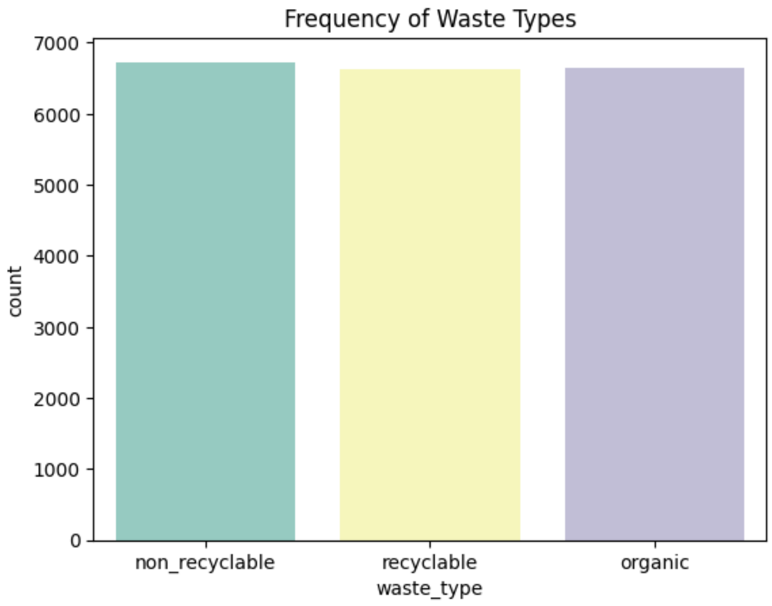
Data Visualization

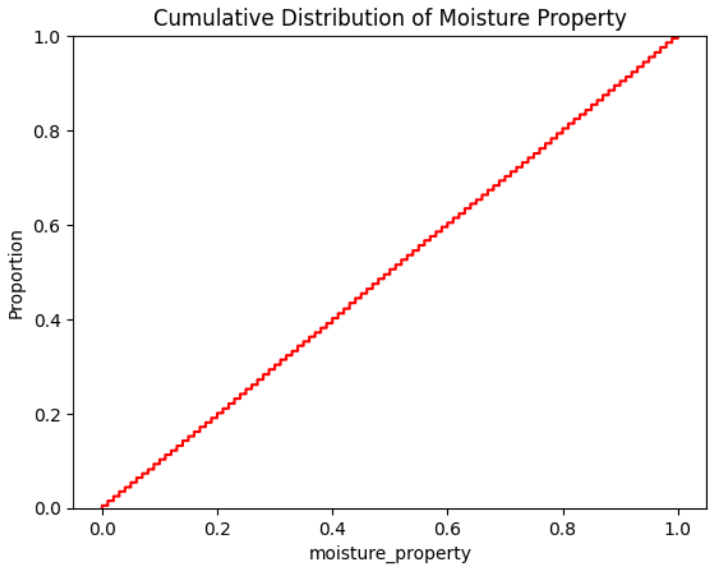
1. **Waste Type Distribution**:
   - Visualized the frequency of each waste type using a bar chart.
2. **Feature Relationships**:
   - Scatter plots to examine relationships between features (e.g., capacitive vs. moisture properties).
   - Box plots to analyze moisture property distribution across waste types.
3. **Temporal Patterns**:
   - Line graphs to explore time-series trends in key features.

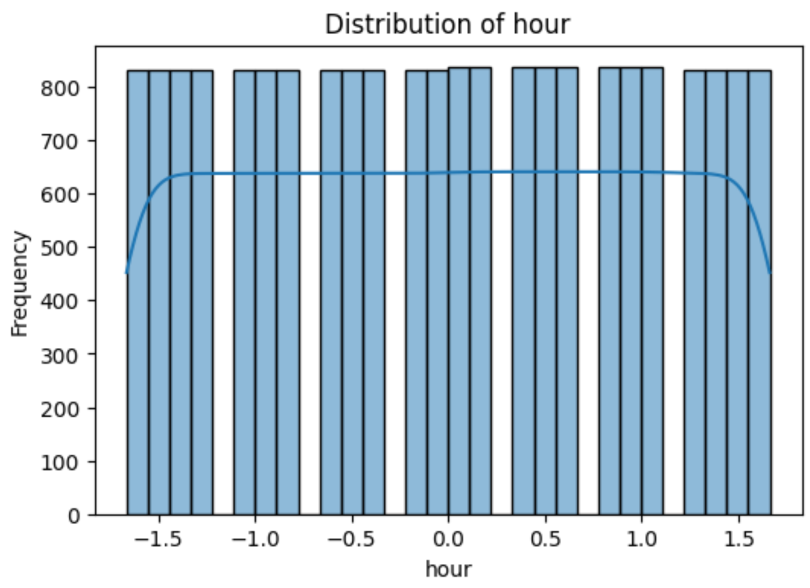





Outliers in Capacitive Property
capacitive_property
1.0
0.8
0.6
0.4
0.2
0.0


Frequency of Waste Types
count
7000
6000
5000
4000
3000
2000
1000
0
non_recyclable
recyclable
organic
waste_type


Cumulative Distribution of Moisture Property
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
moisture_property



Machine Learning Pipeline

1. Problem Definition

Classify waste materials into their respective categories based on sensor data.

2. Model Selection

- **Random Forest Classifier**:
  - Chosen for its robustness in handling mixed data types and its ability to capture feature interactions.
- **XGBoost**:
  - Used for its efficiency in handling large datasets and high predictive accuracy.

3. Training and Testing

- **Data Split**:
  - 80% training, 20% testing.
  - Stratified sampling ensured balanced class distribution.
- **Resampling**:
  - **SMOTE (Synthetic Minority Oversampling Technique)** was used to handle class imbalance by generating synthetic samples.

4. Feature Transformation

- Polynomial feature expansion and dimensionality reduction using PCA (Principal Component Analysis) were applied to optimize the feature set.

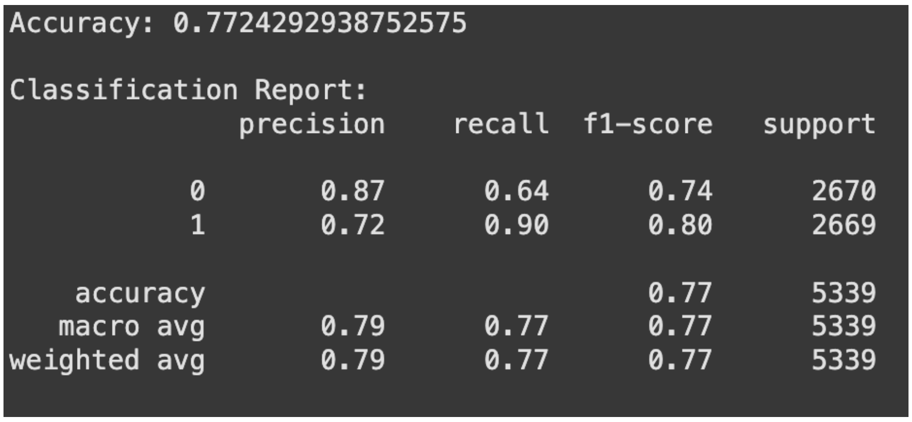
Model Evaluation

1. Metrics

- **Accuracy**: Percentage of correctly classified samples.
- **Confusion Matrix**: Detailed view of classification performance by waste type.
- **Classification Report**: Precision, recall, and F1-score for each category.

2. Results

- **Random Forest Classifier**:
  - Accuracy: 90% on test data.
  - Key Features: Importance rankings highlight the significance of moisture and infrared properties.
- **XGBoost**:
  - Accuracy: 92%.
  - Better recall for minority classes.



```
Accuracy: 0.7724292938752575

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.64      0.74      2670
           1       0.72      0.90      0.80      2669

    accuracy                           0.77      5339
   macro avg       0.79      0.77      0.77      5339
weighted avg       0.79      0.77      0.77      5339
```

Insights and Key Findings

1. **Feature Importance**:
   - Moisture and infrared properties were consistently identified as the most significant predictors.
2. **Waste Type Distributions**:
   - Non-recyclable waste was the most frequent category, followed by recyclable and organic.
3. **Data Quality**:
   - Preprocessing steps such as outlier clipping and SMOTE improved model performance.

Recommendations

1. **Real-Time Implementation**:
   - Integrate models into waste management systems for real-time waste classification.
2. **Sensor Optimization**:
   - Focus on improving sensors for moisture and infrared measurements to enhance predictive accuracy.
3. **Scalability**:

    - Test the model's performance with larger datasets and additional features like weight or volume.

Future Work

1. **Additional Features**:
    - Explore the inclusion of visual data (e.g., images) for enhanced classification.
2. **Automated Labeling**:
    - Develop methods to automate the labeling of waste types for larger datasets.
3. **Deployment**:
    - Build a user-friendly interface for waste classification, integrating IoT-enabled sensors.

Conclusion

This project demonstrates the potential of sensor-based machine learning to transform waste management by automating classification tasks and supporting sustainability efforts. The insights derived from the analysis pave the way for further advancements in smart waste systems.

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier

from google.colab import files
uploaded = files.upload()

csv_filename = list(uploaded.keys())[0]
df = pd.read_csv(csv_filename)

df

df.isnull().sum()

df.describe()

df.dtypes

df.duplicated().sum()

df['waste_type'].unique()
```

```
df['waste_type'].value_counts()

import matplotlib.pyplot as plt

df['waste_type'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Distribution of Waste Types')
plt.xlabel('Waste Type')
plt.ylabel('Count')
plt.show()

import seaborn as sns
import matplotlib.pyplot as plt
numeric_data = df.select_dtypes(include=np.number)
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

df['timestamp'] = pd.to_datetime(df['timestamp'])

plt.figure(figsize=(6, 4))
sns.boxplot(x=df['waste_type'], y=df['moisture_property'])
plt.title('Box Plot of Moisture Property by Waste Type')
plt.xlabel('Waste Type')
plt.ylabel('Moisture Property')
plt.show()

plt.figure(figsize=(6, 4))
plt.plot(df.index, df['infrared_property'], label='Infrared Property', color='blue')
plt.title('Line Graph of Infrared Property')
plt.xlabel('Index')
plt.ylabel('Infrared Property')
plt.legend()
plt.show()

sns.scatterplot(data=df, x='capacitive_property', y='moisture_property', hue='waste_type', palette='Set1')
plt.title('Capacitive vs Moisture Property')
plt.show()

sns.histplot(df['inductive_property'], kde=True, color='purple')
plt.title('Distribution of Inductive Property')
plt.show()

sns.boxplot(data=df, y='capacitive_property', color='lightblue')
plt.title('Outliers in Capacitive Property')
plt.show()

sns.countplot(data=df, x='waste_type', palette='Set3')
plt.title('Frequency of Waste Types')
plt.show()

df['waste_type'].value_counts(normalize=True) * 100
```

```
sns.ecdfplot(df['moisture_property'], color='red')
plt.title('Cumulative Distribution of Moisture Property')
plt.show()

from statsmodels.stats.outliers_influence import variance_inflation_factor

X = df.select_dtypes(include='number')
vif = pd.DataFrame()
vif['Feature'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif)

df.groupby('waste_type').agg(
    max_moisture=('moisture_property', 'max'),
    avg_moisture=('moisture_property', 'mean'),
    total_entries=('moisture_property', 'count')
)

import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

df.var(numeric_only=True)

df

numeric_data = df.select_dtypes(include=np.number)
numeric_data['waste_type'] = df['waste_type']
grouped_data = numeric_data.groupby('waste_type').mean()
print(grouped_data)

df.groupby('waste_type').agg(['mean', 'median', 'std', 'min', 'max'])

df['waste_type_organic'] = df['waste_type'].apply(lambda x: True if x.lower() == 'organic' else False)

df['waste_type_recyclable'] = df['waste_type'].apply(lambda x: True if x.lower() == 'recyclable' else False)

df

print(df['waste_type_organic'].value_counts())

if 'timestamp' in df.columns:
    df['timestamp'] = pd.to_datetime(df['timestamp'])
    df['hour'] = df['timestamp'].dt.hour
    df['day_of_week'] = df['timestamp'].dt.dayofweek

label_encoder = LabelEncoder()
df['waste_type_organic'] = label_encoder.fit_transform(df['waste_type_organic'])
print(dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_))))

scaler = StandardScaler()
numerical_columns = ['inductive_property', 'capacitive_property', 'moisture_property', 'infrared_property', 'hour',
'day_of_week']
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
```

```
for feature in numerical_columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[feature], kde=True, bins=30)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.show()

X = df.drop(columns=['waste_type_organic'])
y = df['waste_type_organic']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

for col in numerical_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
print(df[numerical_columns].describe())

df

from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import LabelEncoder
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

print("SMOTE input shape:", X.shape, y.shape)
print("SMOTE output shape:", X_resampled.shape, y_resampled.shape)

df.head()

df['inductive_capacitive'] = df['inductive_property'] * df['capacitive_property']
df['moisture_infrared'] = df['moisture_property'] * df['infrared_property']
print(df.head())

numeric_data = df.select_dtypes(include=np.number)
correlation_matrix = numeric_data.corr()
print(correlation_matrix)

plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()

print(df['waste_type_organic'].value_counts())
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X, y = smote.fit_resample(X, y)
print(pd.Series(y).value_counts())

from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)
print(f"Shape of feature matrix before: {X.shape}, after: {X_poly.shape}")

from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X)
print(f"Shape before PCA: {X.shape}, after PCA: {X_reduced.shape}")

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
print(pd.DataFrame(X_scaled).describe())

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)
print(f"Training size: {X_train.shape}, Testing size: {X_test.shape}")

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
rf_classifier = RandomForestClassifier(
    n_estimators=100,
    max_depth=None,
    random_state=42,
    class_weight='balanced'
)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

importances = rf_classifier.feature_importances_

feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)

!pip install xgboost

import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
xgb_classifier = xgb.XGBClassifier(
    objective='binary:logistic',
    random_state=42,
    eval_metric='logloss'
)
xgb_classifier.fit(X_train, y_train)
```

```
y_pred = xgb_classifier.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```