**SYNOPSIS**

Waste management is a global challenge caused by rapid urbanization, population growth and the environmental need to reduce the impact of waste on ecosystems and human health. Good waste sorting is an important part of sustainable waste management as it encourages recycling and disposal and reduces landfill and environmental footprint. Biodegradable and non-biodegradable waste are two broad categories with different methods of disposal and recycling. Biodegradable waste contains organic materials that can be processed through composting and anaerobic digestion to produce useful products such as fertilizers and biogas, while non-biodegradable waste such as plastic, glass and metals may require recycling or proper disposal. Traditional waste sorting is done manually; This is time-consuming, labour-intensive and error-prone, making work even less efficient. Also, due to the incompatibility of the books, it is very difficult to transfer repeated editions. To solve these problems, there is a growing interest in the development of automatic waste separation systems that can increase the efficiency and accuracy of the waste management process with the help of machine learning techniques.

**SYSTEM ENVIRONMENT**

2.1 Hardware Requirements:

Processor : Intel Core i4 (10th Gen)

Ram : 4.0 GB

2.2 Software Requirements

Operating System : Windows 10

Framework : Google colab

Language : python

**2.3 About the technology:**

Python:

Python is an interpreted high-level general-purpose programming language created by Guido Van Rossum and first published in 1991. Python's design philosophy emphasizes code readability with significant whitespace. Its language structures and object-oriented approach are designed to help developers write clear and logical code for small and large projects. Python is dynamically typed and garbage

Google Colab:

Google Colab, short for Google Colaboratory, is a cloud-based, interactive computing platform provided by Google. It allows users to write and execute Python code in a collaborative and convenient environment directly through a web browser. Colab provides free access to GPU and TPU (Tensor Processing Unit) resources, enabling accelerated execution of machine learning tasks. Users can create and share Jupyter notebooks, incorporating text, code, and visualizations seamlessly. Colab integrates with Google Drive, facilitating easy storage and sharing of notebooks. Its collaborative features enable multiple users to work on the same document simultaneously, fostering collaborative research and development. Overall, Google Colab is a powerful and accessible tool for data analysis, machine learning, and collaborative coding, making it particularly valuable for researchers, students, and practitioners in the field of data science.

Scikit Learn:

Scikit-learn (Sklearn) is the most useful and powerful Python machine learning library. It provides a number of powerful tools for machine learning and statistical modelling, including classification, regression, clustering and dimensionality reduction through a Python consistent interface. Written mostly in Python, this library is built on top of NumPy, SciPy and Matplotlib. Originally called scikits. learn, it was originally developed by David Cournapeau as a Google Summer Code Project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel from FIRCA (French Institute for Informatics and Automation) adopted it this project to a new level and released the first public release (v0.1 beta) on February 1, 2010

**EXISTING SYSTEM**

Efficient waste management is crucial in airport environments to maintain cleanliness, safety, and sustainability. Segregating waste into biodegradable and non-biodegradable categories is essential for effective disposal and recycling practices. This document outlines an existing system specifically designed for segmenting biodegradable and non-biodegradable waste in airport systems.

**1. Sensor-Based Systems:**

* Waste bins equipped with sensors are strategically placed throughout the airport premises. These sensors detect when bins reach capacity or when waste is deposited, triggering the collection process.
* These systems employ sensors like near-infrared (NIR) spectroscopy or electromagnetic induction to identify material composition based on specific properties. They offer improved accuracy but require initial investment and ongoing maintenance.
* **Examples:** ZenRobotics, Amp Robotics, Recyclebot

**2. Sorting Mechanism:**

* Upon collection, the waste is transported to a sorting facility within the airport. Here, an automated sorting mechanism separates the waste into biodegradable and non-biodegradable categories based on predefined criteria such as material composition and degradation potential.

**3. Conveyor System:**

* After sorting, the waste is conveyed to designated areas for further processing. Biodegradable waste is directed towards composting or anaerobic digestion facilities, while non-biodegradable waste is sent for recycling or appropriate disposal.

The existing system for segmenting biodegradable and non-biodegradable waste in airport environments represents a significant step towards sustainable waste management practices. By leveraging technology and automation, airports can effectively reduce their environmental footprint while optimising operational efficiency. Continued research and development in this field will further enhance the effectiveness and scalability of waste management systems in airports and other high-traffic areas.

**PROPOSED SYSTEM**

Our proposed system for segmenting bio-degradable and non-biodegradable materials leverages machine learning (ML) techniques to classify and categorize waste images effectively. The system involves a two-step process. Firstly, a feature extraction enhances the dataset. Subsequently, a machine learning model, such as a svm,rfc will be trained on the pre-processed images for classification. The model will learn distinctive features to differentiate between bio-degradable and non-biodegradable materials, offering a reliable solution for waste categorization.

**Advantages of the proposed system:**

Accuracy and Efficiency:

The ML model enhances accuracy in waste categorization, providing an efficient solution to automate the segregation process.

Environmental Impact:

Accurate classification contributes to better waste management practices, promoting environmentally conscious disposal methods and reducing the ecological footprint.

Resource Optimization:

The proposed system optimizes resource utilization by automating the sorting process, minimizing manual intervention, and streamlining waste management operations.

Scalability:

The ML model's scalability allows it to adapt to varying waste compositions and datasets, ensuring applicability in diverse environments and scenarios.

Time Savings:

Automation of waste classification saves time in sorting processes, facilitating quicker and more efficient waste disposal workflows.

Data-Driven Insights:

The ML model generates valuable data-driven insights into waste composition trends, aiding in the development of targeted waste reduction strategies.

Reduced Contamination:

Accurate segmentation minimizes cross-contamination between bio-degradable and non-biodegradable waste streams, improving recycling efficiency.

Adaptability:

The proposed system can be adapted to accommodate evolving waste management challenges and integrate with existing waste processing infrastructure.

Cost-Effectiveness:

Automation and precision in waste classification contribute to cost savings by reducing the need for manual sorting and enhancing overall operational efficiency.

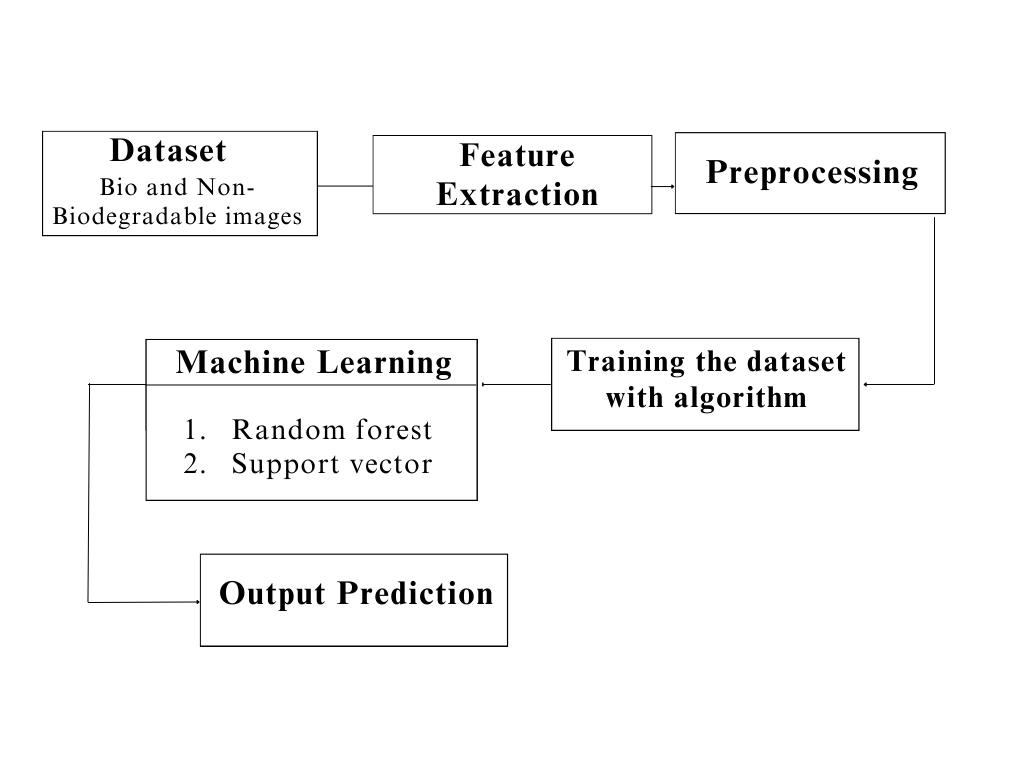
Educational Impact:

The system can serve as an educational tool, raising awareness about waste types and encouraging responsible waste disposal practices.

By combining image processing and machine learning, our proposed system aims to revolutionize waste management practices, offering a sustainable and technologically advanced solution for segregating bio-degradable and non-biodegradable materials.

**SYSTEM DESIGN:**

Segmenting bio degradable and non - biodegradable waste system is designed by the below systematic diagram:



**Dataset Description:**

The dataset consists of bio and non-biodegradable materials, providing a comprehensive collection for environmental analysis. Bio-degradable images showcase organic materials, such as food waste and plant-based items, while non-biodegradable images feature materials like plastics, metals, and synthetic compounds. This diverse dataset serves as a valuable resource for developing machine learning models and image recognition algorithms aimed at distinguishing between materials based on their biodegradability. Understanding and categorizing these materials can contribute to advancements in waste management strategies and sustainable practices, addressing the global challenge of effective waste disposal and environmental conservation.

**Feature Extraction:**

Feature extraction using the Grey-Level Co-occurrence Matrix (GLCM) is a widely utilized technique in image processing and pattern recognition. GLCM quantifies the spatial relationships between pixel intensity values in an image by computing the frequency of occurrence of pairs of intensity values at various spatial displacements. By analysing these relationships, GLCM generates a set of statistical measures, such as contrast, correlation, energy, and homogeneity, which serve as texture descriptors capturing different aspects of image texture and structure. These descriptors effectively summarize the texture information within an image, providing a compact and informative representation for subsequent analysis tasks, such as classification, segmentation, and object detection. Overall, feature extraction using GLCM enables the extraction of discriminative texture features from images, facilitating the characterization and understanding of complex visual patterns in diverse application

**Pre-Processing:**

In the preprocessing phase, the 'image\_names' column in the dataset is first converted to the 'object' data type to accommodate various types of string values. Subsequently, any leading or trailing whitespaces in the 'image\_names' are removed for uniformity and cleanliness. To facilitate categorical encoding, the column is converted to the 'category' data type, and categorical codes are assigned to each unique image name. Finally, to optimize memory usage and ensure compatibility with downstream operations, the data type is further converted to 32-bit integers (np.int32). This series of preprocessing steps enhances the dataset's consistency, prepares it for categorical analysis, and promotes efficient computational performance during subsequent tasks.

**Machine learning algorithm**

**1.Random Forest**

Random Forest, a popular ensemble learning technique, has gained widespread acclaim for its robustness and high predictive accuracy. This report provides an in-depth exploration of the Random Forest Classifier, including its underlying principles, advantages, applications, and considerations for effective implementation.

Principles:

Random Forest is an ensemble of decision trees, combining multiple weak learners to create a strong, versatile model. Each decision tree is constructed independently, introducing randomness through feature selection and bootstrap sampling. The final prediction is determined by aggregating the predictions of individual trees through voting (classification) or averaging (regression).

Advantages:

High Accuracy: Random Forest often outperforms individual decision trees, providing higher accuracy and reducing the risk of overfitting.

Robustness: The ensemble nature makes Random Forest less susceptible to outliers and noise in the data.

Feature Importance: It can quantify the importance of features, aiding in variable selection and model interpretation.

Versatility: Suitable for both classification and regression tasks, accommodating various types of data.

Applications:

Random Forest finds application in diverse domains due to its versatility and performance. Some notable applications include:

Finance: Credit scoring, fraud detection.

Healthcare: Disease prediction, patient outcome analysis.

Marketing: Customer churn prediction, targeted advertising.

Remote Sensing: Land cover classification, object detection.

Manufacturing: Quality control, predictive maintenance.

Considerations:

Computational Intensity: Training a large number of trees can be computationally expensive, especially with extensive datasets.

Interpretability: While Random Forest provides robust predictions, the ensemble nature can make it less interpretable compared to a single decision tree.

Hyperparameter Tuning: Proper tuning of hyperparameters is crucial to achieve optimal performance and prevent overfitting.

Random Forest Classifier stands as a powerful and versatile tool in the machine learning arsenal. Its ability to handle complex relationships in data, high accuracy, and resilience to overfitting make it a go-to choose for many practitioners. Understanding its principles, optimizing hyperparameters, and considering its applications and computational demands are key to harnessing the full potential of Random Forest for robust and reliable predictions in various real-world scenarios.

**2.Support Vector**

Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm renowned for its efficacy in both classification and regression tasks. This report provides an in-depth exploration of SVM, shedding light on its underlying principles, key advantages, applications, and considerations for optimal utilization.

Principles:

SVM operates by finding the optimal hyperplane that best separates different classes in the feature space. This hyperplane is determined by support vectors, which are data points closest to the decision boundary. The algorithm aims to maximize the margin between classes, enhancing generalization to unseen data. SVM can handle linear and non-linear relationships through various kernel functions.

Advantages:

Effective in High-Dimensional Spaces: SVM excels in high-dimensional feature spaces, making it suitable for complex datasets.

Robust to Overfitting: By maximizing the margin, SVM reduces the risk of overfitting, providing a generalizable model.

Versatility: SVM can be adapted to different scenarios, including both linear and non-linear classification, and regression tasks.

Applications:

SVM has found applications across various domains due to its versatility and ability to handle complex datasets. Some notable applications include:

Image Classification: Recognizing objects in images.

Text Classification: Spam detection, sentiment analysis.

Bioinformatics: Protein structure prediction, gene classification.

Finance: Credit scoring, stock price prediction.

Healthcare: Disease diagnosis, outcome prediction.

Considerations:

Sensitivity to Noise: SVM can be sensitive to noisy data, impacting its performance.

Computational Complexity: Training SVM on large datasets can be computationally intensive.

Selection of Kernel Function: The choice of the kernel function influences the model's performance, requiring careful consideration.

Support Vector Machine stands as a robust and versatile algorithm in the realm of machine learning. Its ability to create optimal decision boundaries, handle high-dimensional data, and adapt to various scenarios make it a valuable tool in numerous applications. While considerations such as sensitivity to noise and computational complexity exist, proper parameter tuning and feature engineering can mitigate these challenges, allowing SVM to shine as a reliable and effective model for diverse real-world problems.

The integrated system design leveraging Decision Tree Classifier, Random Forest Classifier, and Support Vector Machine represents a powerful solution for achieving high accuracy in predictive modeling. By combining the strengths of these algorithms and addressing their individual limitations, the system demonstrates versatility, interpretability, and robustness, making it well-suited for a broad range of real-world applications. Ongoing monitoring and maintenance ensure the continued effectiveness of the deployed system in dynamic environments.

Libraries used in the implementation:

NumPy: NumPy is a fundamental library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions. It serves as a foundational tool for scientific computing tasks, enabling efficient and high-performance operations on numerical data.

Pandas: Pandas is a versatile data manipulation library in Python that offers data structures like DataFrames and Series, facilitating efficient data analysis and manipulation. It provides functionalities for cleaning, transforming, and exploring datasets, making it a go-to tool for handling structured data in various stages of the data science workflow.

Matplotlib: Matplotlib is a powerful plotting library for Python that allows the creation of diverse static, animated, and interactive visualizations. With a comprehensive set of functions, Matplotlib provides users with the flexibility to create various charts, plots, and graphs, making it an essential tool for data visualization and communication of findings.

Seaborn: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating aesthetically pleasing and informative statistical graphics. Seaborn simplifies the process of generating complex visualizations, including heatmaps, pair plots, and violin plots, while maintaining customization options for advanced users.

Metrics (Accuracy, Classification, Confusion Matrix, ROC AUC): In the context of machine learning evaluation, metrics play a crucial role. Accuracy represents the proportion of correctly classified instances, serving as a fundamental measure of model performance. Classification metrics, such as precision, recall, and F1-score, provide insights into the model's ability to correctly identify instances of a particular class. The confusion matrix presents a comprehensive summary of true positive, true negative, false positive, and false negative predictions. Lastly, the ROC AUC (Receiver Operating Characteristic - Area Under the Curve) is a performance metric for binary classification models, illustrating the trade-off between sensitivity and specificity across different thresholds, providing a holistic view of the model's discriminatory power. These metrics collectively aid in assessing and optimizing the performance of machine learning models.

**CODING**

import cv2

import pandas as pd

import numpy as np

import os

from skimage.feature import greycomatrix

from skimage.feature import greycoprops

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.svm import SVC

from tqdm import tqdm

def get\_feature(img):

img\_graymatrix = greycomatrix(img, [1], [0, np.pi/2])

# print("img\_graymatrix shape: ",img\_graymatrix.shape)

img\_contrast = greycoprops(img\_graymatrix, 'contrast')

# print("img\_contrast shape: ", img\_contrast.shape)

img\_homogeneity = greycoprops(img\_graymatrix, 'homogeneity')

# print("img\_homogeneity shape: ", img\_homogeneity.shape)

img\_correlation = greycoprops(img\_graymatrix, 'correlation')

# print("img\_correlation shape: ", img\_correlation.shape)

img\_dissimilarity = greycoprops(img\_graymatrix, 'dissimilarity')

# print("img\_dissimilarity shape: ", img\_dissimilarity.shape)

img\_energy = greycoprops(img\_graymatrix, 'energy')

# print("img\_energy shape: ", img\_energy.shape)

img\_contrast\_flattened = img\_contrast.flatten()

img\_homogeneity\_flattened = img\_homogeneity.flatten()

img\_correlation\_flattened = img\_correlation.flatten()

img\_dissimilarity\_flattened = img\_dissimilarity.flatten()

img\_energy\_flattened = img\_energy.flatten()

features = np.concatenate([img\_contrast\_flattened, img\_homogeneity\_flattened,

img\_correlation\_flattened,img\_energy\_flattened,img\_dissimilarity\_flattened])

# print("final\_feature shape: ",features.shape)

return(features)

def feature\_extraction(path\_to\_folder, class\_label):

data\_list=[]

# count=1

for file\_name in tqdm(os.listdir(path\_to\_folder)):

# if(count>1):

# break

path\_to\_img = os.path.join(path\_to\_folder,file\_name)

img = cv2.imread(path\_to\_img, 0) # grayscale image

if np.shape(img) == ():

continue

final\_feature = get\_feature(img)

# print("final\_feature shape: ",final\_feature.shape)

# print("final\_feature is: ",final\_feature)

final\_feature=list(final\_feature)

final\_feature.insert(0,file\_name)

final\_feature.insert(1,class\_label)

data\_list.append(final\_feature)

# count+=1

return(data\_list)

from tqdm import tqdm

import cv2

bio = '/content/drive/MyDrive/val/biodegradable'

nonbio= '/content/drive/MyDrive/val/non\_biodegradable'

data\_list1 = feature\_extraction(nonbio, 0)

data\_list2 = feature\_extraction(bio, 1)

df = pd.DataFrame(data\_list1)

df = df.append(pd.DataFrame(data\_list2), ignore\_index=True)

# # --------------------------------------------------------------------------------------

df.rename(columns = {0: "image\_names", 1: "label"}, inplace = True)

df.head(1500)

df.to\_csv('features.csv', index=False)

data=pd.read\_csv('/content/features.csv')

data

data.columns

data.info()

data.dtypes

data['image\_names'] = data['image\_names'].astype('object')

data['image\_names'] = data['image\_names'].str.strip()

data['image\_names'] =data['image\_names'].astype('category')

data['image\_names'] = data['image\_names'].cat.codes

data['image\_names'] = data['image\_names'].astype(np.int32)

X = data.drop(['label'], axis=1)

y = data['label']

# split X and y into training and testing sets

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.33, random\_state = 42)

# import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

# instantiate the classifier

rfc = RandomForestClassifier(random\_state=0)

# fit the model

rfc.fit(X\_train, y\_train)

# Predict the Test set results

y\_pred = rfc.predict(X\_test)

# importing required libraries

import numpy as np

import pandas as pd

import pickle # saving and loading trained model

from os import path

# importing required libraries for normalizing data

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

# importing library for plotting

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve, auc

accuracy\_rfc =accuracy\_score(y\_test, y\_pred)

from sklearn.metrics import accuracy\_score

print('Model accuracy score of rfc: ',accuracy\_rfc )

# Print the Confusion Matrix and slice it into four pieces

from sklearn.metrics import confusion\_matrix

cm\_rfc = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix\n\n', cm\_rfc)

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_rfc, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

# Plotting a heatmap for precision, recall, and F1-score

class\_report = classification\_report(y\_test, y\_pred, output\_dict=True)

class\_names = [str(label) for label in class\_report.keys() if label not in ['accuracy', 'macro avg', 'weighted avg']]

# Extract precision, recall, and F1-score for each class

heatmap\_data = [[class\_report[class\_name]['precision'], class\_report[class\_name]['recall'], class\_report[class\_name]['f1-score']] for class\_name in class\_names]

# Create a heatmap

fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(heatmap\_data, annot=True, fmt=".2f", xticklabels=['Precision', 'Recall', 'F1-Score'], yticklabels=class\_names, cmap='Blues')

plt.title('Classification Report Heatmap')

plt.show()

# Calculate the ROC Precision, Recall, and F1-Score

from sklearn.metrics import roc\_auc\_score, roc\_curve, auc, precision\_recall\_fscore\_support

import matplotlib.pyplot as plt

# Calculate the AUC

auc = roc\_auc\_score(y\_test, y\_pred)

print('AUC: %.2f' % auc)

# Calculate the ROC

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

# plot the roc curve

plt.plot(fpr, tpr)

plt.title('ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

"""SVM"""

# import SVC classifier

from sklearn.svm import SVC

# instantiate classifier with default hyperparameters

svc=SVC()

# fit classifier to training set

svc.fit(X\_train,y\_train)

# make predictions on test set

y\_pred=svc.predict(X\_test)

accuracy\_svc=accuracy\_score(y\_test, y\_pred)

from sklearn.metrics import accuracy\_score

print('Model accuracy score of svc: ',accuracy\_svc)

# Print the Confusion Matrix and slice it into four pieces

from sklearn.metrics import confusion\_matrix

cm\_svc = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix\n\n', cm\_svc)

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_svc, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

# Plotting a heatmap for precision, recall, and F1-score

class\_report = classification\_report(y\_test, y\_pred, output\_dict=True)

class\_names = [str(label) for label in class\_report.keys() if label not in ['accuracy', 'macro avg', 'weighted avg']]

# Extract precision, recall, and F1-score for each class

heatmap\_data = [[class\_report[class\_name]['precision'], class\_report[class\_name]['recall'], class\_report[class\_name]['f1-score']] for class\_name in class\_names]

# Create a heatmap

fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(heatmap\_data, annot=True, fmt=".2f", xticklabels=['Precision', 'Recall', 'F1-Score'], yticklabels=class\_names, cmap='Blues')

plt.title('Classification Report Heatmap')

plt.show()

# Calculate the ROC Precision, Recall, and F1-Score

from sklearn.metrics import roc\_auc\_score, roc\_curve, auc, precision\_recall\_fscore\_support

import matplotlib.pyplot as plt

# Calculate the AUC

auc = roc\_auc\_score(y\_test, y\_pred)

print('AUC: %.2f' % auc)

# Calculate the ROC

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

# plot the roc curve

plt.plot(fpr, tpr)

plt.title('ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

FRAMEWORK CODING:

import tkinter as tk

import tkinter as tk

from tkinter import ttk

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.metrics import roc\_auc\_score, roc\_curve, auc, precision\_recall\_fscore\_support

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from PIL import Image, ImageTk

from sklearn.model\_selection import train\_test\_split

import numpy as np

import pandas as pd

# Load your dataset here

data = pd.read\_csv('features.csv')

data.info()

data['image\_names'] = data['image\_names'].astype('object')

data['image\_names'] = data['image\_names'].str.strip()

data['image\_names'] =data['image\_names'].astype('category')

data['image\_names'] = data['image\_names'].cat.codes

data['image\_names'] = data['image\_names'].astype(np.int32)

X = data.drop(['label'], axis=1)

y = data['label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

# Initialize classifiers

svm\_classifier = SVC(random\_state=0)

rfc\_classifier = RandomForestClassifier(n\_estimators=100, criterion='gini', random\_state=0)

# Tkinter GUI

root = tk.Tk()

root.title("Classifier Metrics")

root.geometry("400x400")

# Load background image

background\_image = Image.open("sample1.jpg") # Replace with your image file

background\_photo = ImageTk.PhotoImage(background\_image)

background\_label = tk.Label(root, image=background\_photo)

background\_label.place(relwidth=1, relheight=1)

# Project label

project\_label = tk.Label(root, text="Design and development of Segmenting bio degradable and non - bio degradable waste in the airport system ", font=("Helvetica", 12), bg="white",)

project\_label.pack(pady=10)

# Labels for dataset information

r\_dataset\_label = tk.Label(root, text="Dataset: bio degradable and non - bio degradable", font=("Helvetica", 11),foreground="blue",width=30)

r\_dataset\_label.pack(pady=10, padx=10)

# Training Data Label

r\_train\_data\_label = tk.Label(root, text="Training Data: 70%", font=("Helvetica", 11),foreground="blue",width=20)

r\_train\_data\_label.pack(pady=10, padx=10)

# Testing Data Label

r\_test\_data\_label = tk.Label(root, text="Testing Data: 30%", font=("Helvetica", 11), foreground="blue",width=20)

r\_test\_data\_label.pack(pady=10, padx=10)

# Function to train classifiers

def train\_svm\_classifier():

global svm\_classifier, X\_train, y\_train

svm\_classifier.fit(X\_train, y\_train)

print("SVM Classifier trained successfully.")

def train\_dtc\_classifier():

global dtc\_classifier, X\_train, y\_train

dtc\_classifier.fit(X\_train, y\_train)

print("DTC Classifier trained successfully.")

def train\_rfc\_classifier():

global rfc\_classifier, X\_train, y\_train

rfc\_classifier.fit(X\_train, y\_train)

print("RFC Classifier trained successfully.")

# Function to calculate metrics and show charts for SVM

def show\_svm\_metrics():

global svm\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = svm\_classifier.predict(X\_test)

# Confusion Matrix

cm\_svm = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix of svm\n\n', cm\_svm)

# Plot Confusion Matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_svm, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix of svm')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

def show\_report\_svm():

# Predict the Test set results

y\_pred = svm\_classifier.predict(X\_test)

# Classification Report

class\_report\_str = classification\_report(y\_test, y\_pred)

print(class\_report\_str)

# Plot Classification Report

class\_report = classification\_report(y\_test, y\_pred, output\_dict=True)

class\_names = [str(label) for label in class\_report.keys() if label not in ['accuracy', 'macro avg', 'weighted avg']]

heatmap\_data = [[class\_report[class\_name]['precision'], class\_report[class\_name]['recall'],

class\_report[class\_name]['f1-score']] for class\_name in class\_names]

# Create a heatmap

fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(heatmap\_data, annot=True, fmt=".2f", xticklabels=['Precision', 'Recall', 'F1-Score'],

yticklabels=class\_names, cmap='Blues')

plt.title('Classification Report Heatmap of svm')

plt.show()

def calculate\_accuracy\_svm():

global svm\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = svm\_classifier.predict(X\_test)

# Accuracy

accuracy\_svm = accuracy\_score(y\_test, y\_pred)

print('Model accuracy score of svm:', accuracy\_svm)

# Plot Accuracy

plt.figure(figsize=(6, 4))

plt.bar(["Accuracy"], [accuracy\_svm], color='blue')

plt.title('Model Accuracy of svm')

plt.ylabel('Accuracy')

plt.show()

def roc\_svm():

global svm\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = svm\_classifier.predict(X\_test)

# Calculate the AUC

auc = roc\_auc\_score(y\_test, y\_pred)

print('AUC: %.2f' % auc)

# Calculate the ROC

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

# plot the roc curve

plt.plot(fpr, tpr)

plt.title('ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

# Function to calculate metrics and show charts for DTC

def show\_dtc\_metrics():

global dtc\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = dtc\_classifier.predict(X\_test)

# Confusion Matrix

cm\_dtc = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix of dtc\n\n', cm\_dtc)

# Plot Confusion Matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_dtc, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix of dtc')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

def show\_report\_dtc():

# Predict the Test set results

y\_pred = dtc\_classifier.predict(X\_test)

# Classification Report

class\_report\_str = classification\_report(y\_test, y\_pred)

print(class\_report\_str)

# Plot Classification Report

class\_report = classification\_report(y\_test, y\_pred, output\_dict=True)

class\_names = [str(label) for label in class\_report.keys() if label not in ['accuracy', 'macro avg', 'weighted avg']]

heatmap\_data = [[class\_report[class\_name]['precision'], class\_report[class\_name]['recall'],

class\_report[class\_name]['f1-score']] for class\_name in class\_names]

# Create a heatmap

fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(heatmap\_data, annot=True, fmt=".2f", xticklabels=['Precision', 'Recall', 'F1-Score'],

yticklabels=class\_names, cmap='Blues')

plt.title('Classification Report Heatmap of dtc')

plt.show()

# Function to calculate metrics and show charts for RFC

def show\_rfc\_metrics():

global rfc\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = rfc\_classifier.predict(X\_test)

# Confusion Matrix

cm\_rfc = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix of rfc\n\n', cm\_rfc)

# Plot Confusion Matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_rfc, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix of rfc')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

def show\_report\_rfc():

# Predict the Test set results

y\_pred = rfc\_classifier.predict(X\_test)

# Classification Report

class\_report\_str = classification\_report(y\_test, y\_pred)

print(class\_report\_str)

# Plot Classification Report

class\_report = classification\_report(y\_test, y\_pred, output\_dict=True)

class\_names = [str(label) for label in class\_report.keys() if label not in ['accuracy', 'macro avg', 'weighted avg']]

heatmap\_data = [[class\_report[class\_name]['precision'], class\_report[class\_name]['recall'],

class\_report[class\_name]['f1-score']] for class\_name in class\_names]

# Create a heatmap

fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(heatmap\_data, annot=True, fmt=".2f", xticklabels=['Precision', 'Recall', 'F1-Score'],

yticklabels=class\_names, cmap='Blues')

plt.title('Classification Report Heatmap of rfc')

plt.show()

def calculate\_accuracy\_rfc():

global rfc\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = rfc\_classifier.predict(X\_test)

# Accuracy

accuracy\_rfc = accuracy\_score(y\_test, y\_pred)

print('Model accuracy score of rfc:', accuracy\_rfc)

# Plot Accuracy

plt.figure(figsize=(6, 4))

plt.bar(["Accuracy"], [accuracy\_rfc], color='blue')

plt.title('Model Accuracy of rfc')

plt.ylabel('Accuracy')

plt.show()

def roc\_rfc\_auc():

global rfc\_classifier, X\_test, y\_test

# Predict the Test set results

y\_pred = rfc\_classifier.predict(X\_test)

# Calculate the AUC

auc = roc\_auc\_score(y\_test, y\_pred)

print('AUC: %.2f' % auc)

# Calculate the ROC

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

# plot the roc curve

plt.plot(fpr, tpr)

plt.title('ROC Curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

# SVM Frame

svm\_frame = tk.Frame(root)

svm\_frame.pack(side=tk.TOP, pady=10)

# SVM Train Button

svm\_train\_button = tk.Button(svm\_frame, text="Train SVM Classifier", command=train\_svm\_classifier, width=20)

svm\_train\_button.pack(side=tk.LEFT, padx=5, pady=5)

# SVM Metrics Button

svm\_metrics\_button = tk.Button(svm\_frame, text="SVM Accuracy", command=calculate\_accuracy\_svm, width=20)

svm\_metrics\_button.pack(side=tk.LEFT, padx=5, pady=5)

# SVM matrix Button

svm\_metrics\_button = tk.Button(svm\_frame, text="SVM Confusion Matrix", command=show\_svm\_metrics, width=20)

svm\_metrics\_button.pack(side=tk.LEFT, padx=5, pady=5)

# SVM report Button

svm\_report\_button = tk.Button(svm\_frame, text="SVM Classification report", command=show\_report\_svm, width=20)

svm\_report\_button.pack(side=tk.LEFT, padx=5, pady=5)

# SVM matrix Button

svm\_rocauc\_button = tk.Button(svm\_frame, text="SVM Roc Auc", command=roc\_svm, width=20)

svm\_rocauc\_button.pack(side=tk.LEFT, padx=5, pady=5)

# RFC Frame

rfc\_frame = tk.Frame(root)

rfc\_frame.pack(side=tk.TOP, pady=10)

# RFC Train Button

rfc\_train\_button = tk.Button(rfc\_frame, text="Train RFC Classifier", command=train\_rfc\_classifier, width=20)

rfc\_train\_button.pack(side=tk.LEFT, padx=5, pady=5)

# RFC Metrics Button

rfc\_metrics\_button = tk.Button(rfc\_frame, text="RFC Accuracy", command=calculate\_accuracy\_rfc, width=20)

rfc\_metrics\_button.pack(side=tk.LEFT, padx=5, pady=5)

# RFC Matrix Button

rfc\_matrix\_button = tk.Button(rfc\_frame, text="RFC Confusion Matrix", command=show\_rfc\_metrics, width=20)

rfc\_matrix\_button.pack(side=tk.LEFT, padx=5, pady=5)

# RFC report Button

rfc\_report\_button = tk.Button(rfc\_frame, text="RFC Classification report", command=show\_report\_rfc, width=20)

rfc\_report\_button.pack(side=tk.LEFT, padx=5, pady=5)

# RFC roc auc Button

rfc\_rocauc\_button = tk.Button(rfc\_frame, text="RFC Roc Auc", command=roc\_rfc\_auc, width=20)

rfc\_rocauc\_button.pack(side=tk.LEFT, padx=5, pady=5)

# Run the Tkinter event loop

root.mainloop()

**RESULTS AND DISCUSSION:**

**Dataset:**

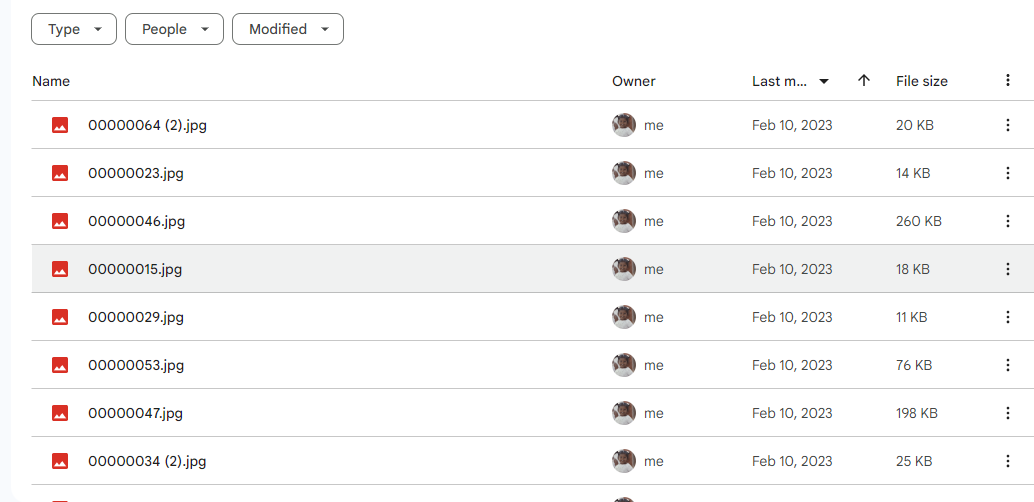


Figure 1: Image dataset

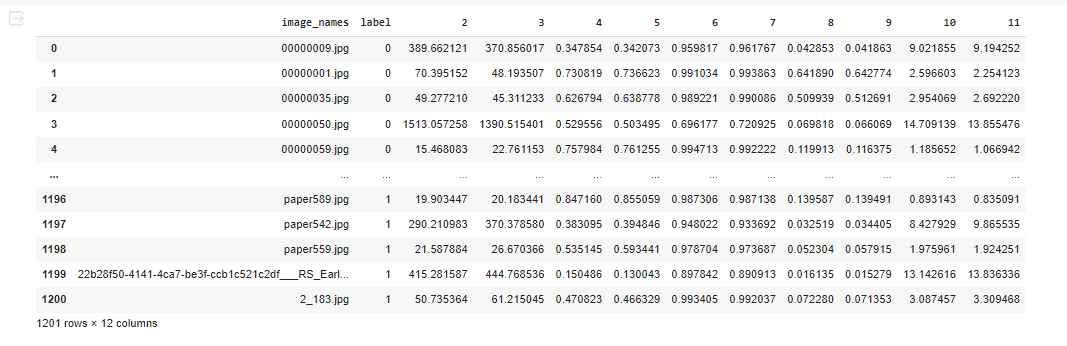


Figure 2: CSV dataset

**Results:**

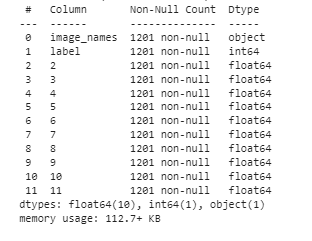


Figure 3: dataset information

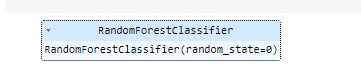


Figure 4: Random forest classifier algorithm



Figure 5: Accuracy calculation of Random Forest classifier algorithm

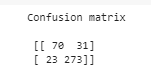


Figure 6: Confusion matrix calculation of Random Forest classifier algorithm

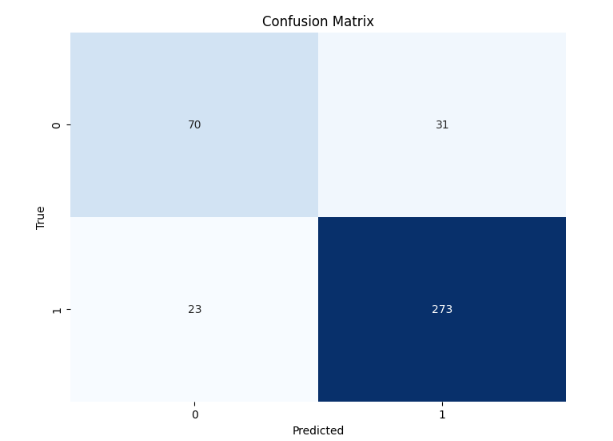


Figure 7: Confusion matrix graph of Random Forest classifier algorithm

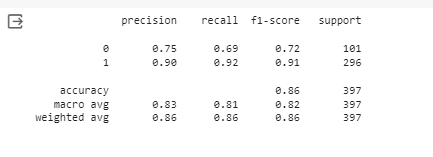


Figure 8: Classification report calculation of Random Forest classifier algorithm

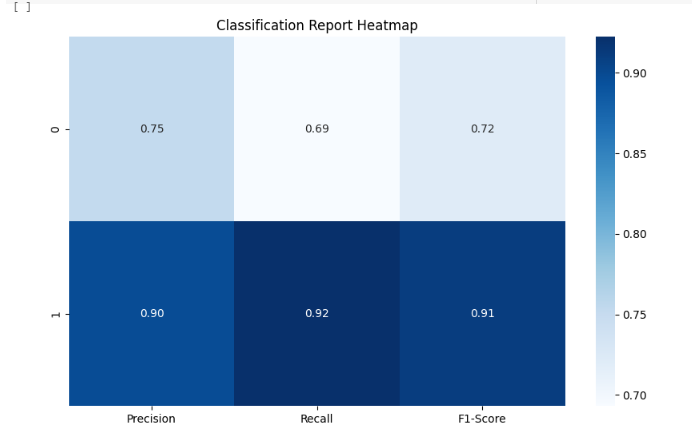


Figure 9: Classification report graph of Random Forest classifier algorithm

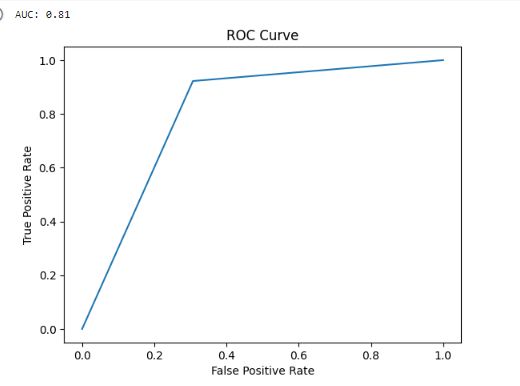


Figure 10: ROC AUC graph of Random Forest classifier algorithm

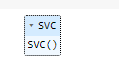


Figure 11: Support vector classifier algorithm



Figure 12: Accuracy calculation of support vector classifier algorithm



Figure 13: Confusion matrix of support vector classifier algorithm

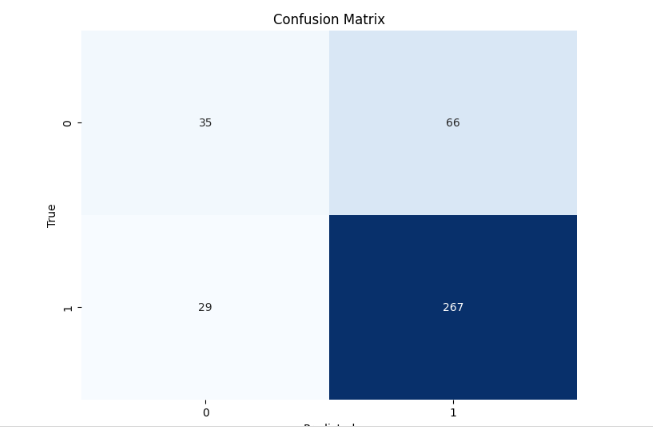


Figure 14: Confusion matrix graph of Support vector classifier algorithm

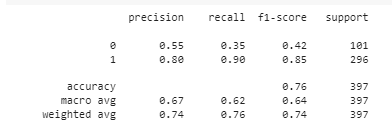


Figure 15: Classification report of support vector classifier algorithm

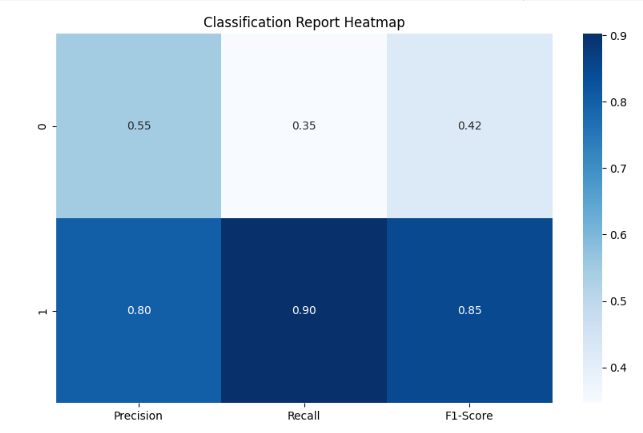


Figure 16: Classification report graph of Support vector classifier algorithm

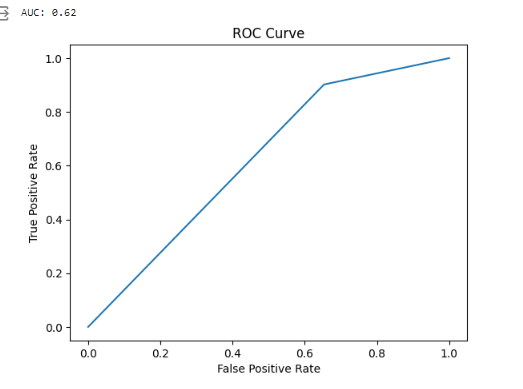


Figure 17: ROC AUC graph of Support vector classifier algorithm

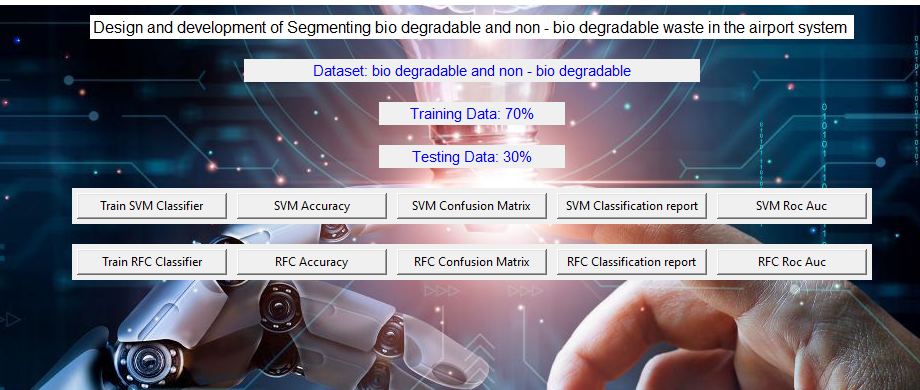


Figure 18: Frame work design



Figure 19: Classifier training

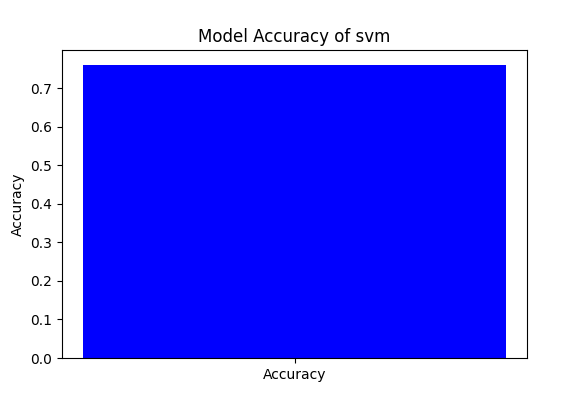


Figure 20: Accuracy graph of Support vector classifier algorithm

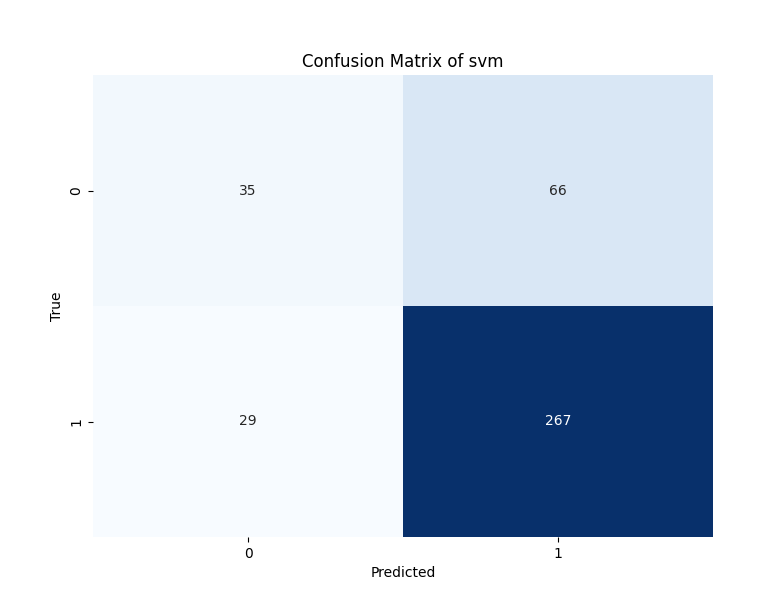


Figure 21: Confusion matrix graph of Support vector classifier algorithm

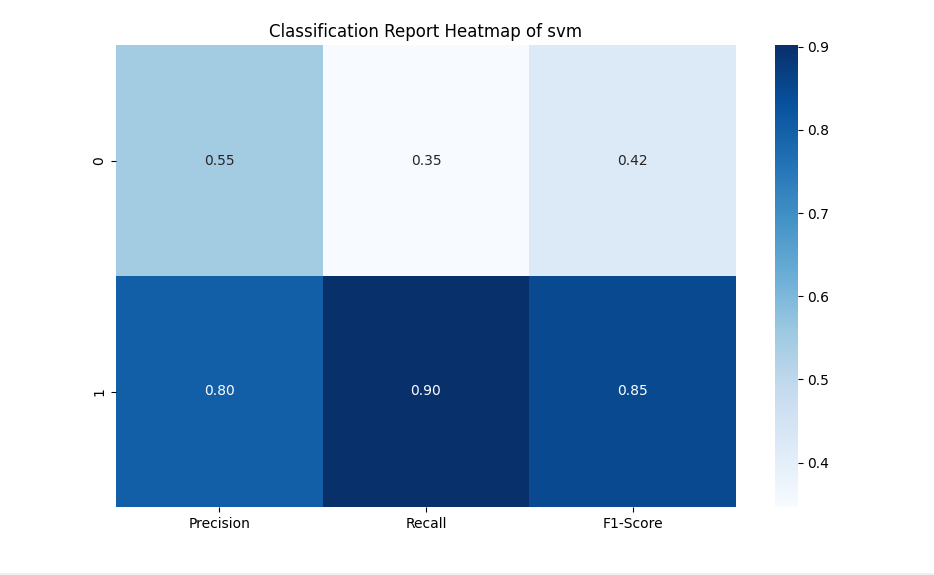


Figure 22: Classification report graph of Support vector classifier algorithm

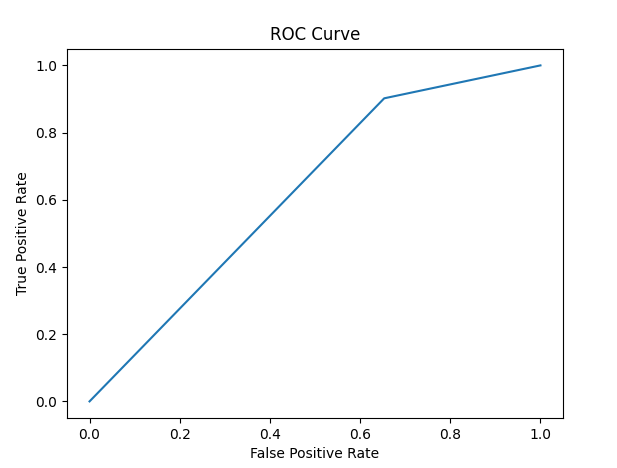


Figure 23: ROC AUC graph of Support vector classifier algorithm

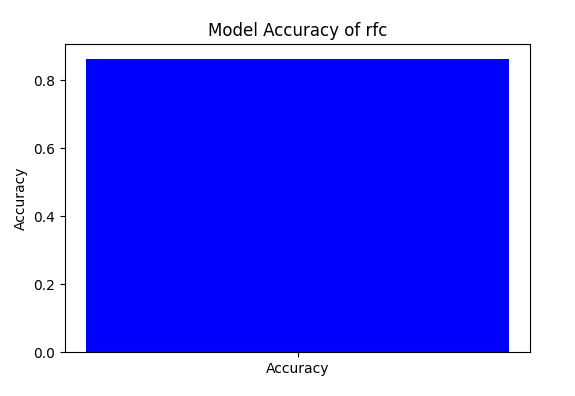


Figure 24: Accuracy graph of Random Forest classifier algorithm

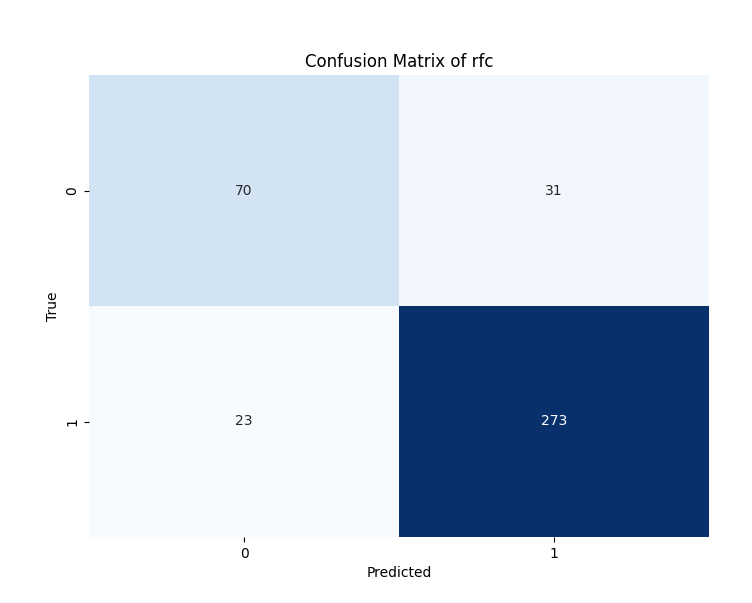


Figure 25: Confusion matrix graph of Random Forest classifier algorithm

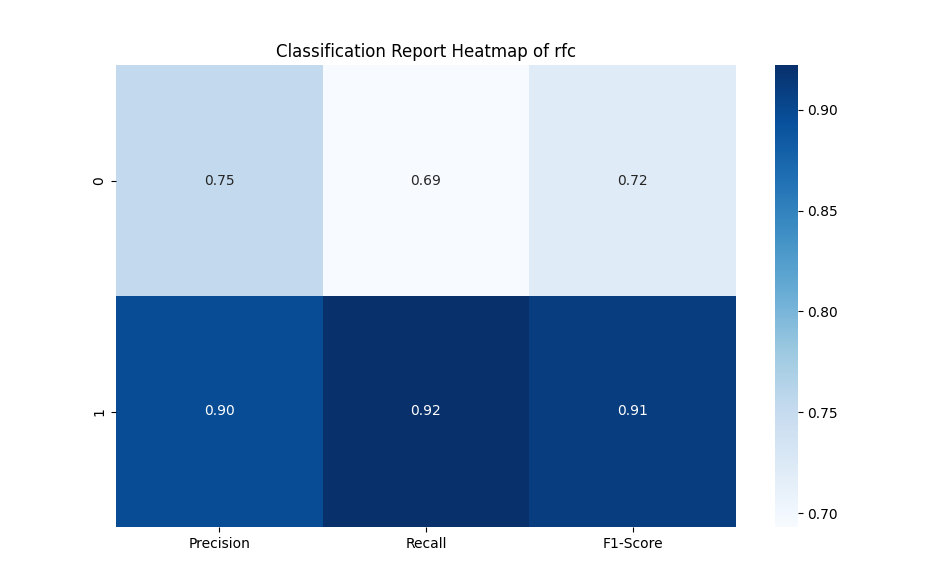


Figure 26: Classification report graph of Random Forest classifier algorithm

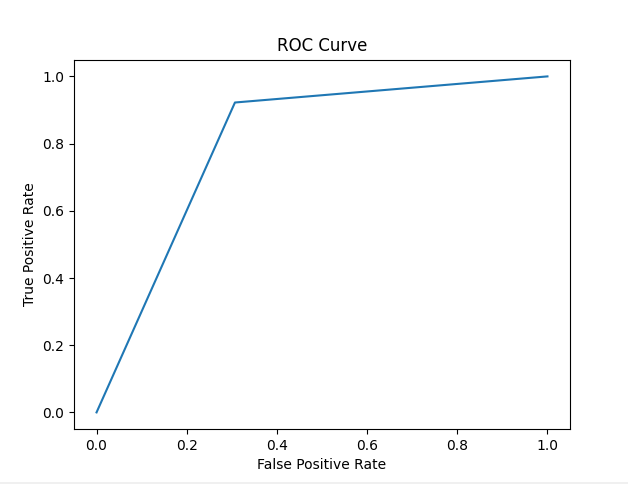


Figure 27: ROC AUC graph of Random Forest classifier algorithm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 – score | Support |
| 0 | 0.75 | 0.69 | 0.72 | 101 |
| 1 | 0.90 | 0.92 | 0.91 | 296 |
| accuracy |  |  | 0.86 | 397 |
| Macro avg | 0.83 | 0.81 | 0.82 | 397 |
| Weighted avg | 0.86 | 0.86 | 0.86 | 397 |

Table 1: classification report of RFC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 – score | Support |
| 0 | 0.55 | 0.35 | 0.42 | 101 |
| 1 | 080 | 0.90 | 0.94 | 296 |
| accuracy |  |  | 0.76 | 397 |
| Macro avg | 0.67 | 0.62 | 0.64 | 397 |
| Weighted avg | 0.74 | 0.76 | 0.74 | 397 |

Table 2: classification report of SVM

The classification report is a performance evaluation tool that shows the precision, recall, f1-score, for each class in a classification problem. In training images using the deep learning model, the classification report would provide information about how well the model performed in classifying images into different categories. The precision represents the percentage of correctly classified images among all the images classified as belonging to a specific class. The recall represents the percentage of correctly classified images among all the images that actually belong to a specific class. The f1-score is a harmonic mean of precision and recall, and support represents the number of images in each class.

The accuracy has been calculated for the model that has been implemented, and the result for the model is compared in Table

|  |  |
| --- | --- |
| Algorithms | Accuracy |
| RFC | 86 |
| SVM | 76 |

Table 3: Accuracy comparison of algorithm.

|  |  |  |
| --- | --- | --- |
| Dataset Count | Training Value | Testing Value |
| 1201 | 70 | 30 |

Table 4: Consist of dataset count, Training and Testing percentage.

**CONCLUSION**

The proposed system integrating machine learning (ML) for the segmentation of bio-degradable and non-biodegradable materials presents a promising avenue for revolutionizing waste management practices. Through the utilization of techniques and the training of a SVM, RFC the system showcases the potential to automate and enhance the accuracy of waste categorization. The robustness of the ML model is evidenced by its adaptability to diverse waste compositions and scenarios, providing a scalable solution for various environments. The advantages of the proposed system extend beyond mere automation. Its impact on environmental sustainability is significant, contributing to more efficient waste disposal and fostering responsible waste management behaviors. The system's ability to generate data-driven insights into waste composition trends empowers decision-makers to formulate targeted strategies for waste reduction and recycling efforts.

Furthermore, the proposed system not only streamlines operational workflows, saving time and resources, but also promotes cost-effectiveness by reducing the reliance on manual sorting processes. The educational aspect of the system is noteworthy, as it serves as a tool to raise awareness about waste types and encourages a more informed and responsible approach to waste disposal among the public. In essence, the proposed ML-based system for waste segmentation represents a forward-thinking solution that aligns with the principles of sustainability and technological innovation. Its successful implementation has the potential to redefine waste management standards, providing a greener, more efficient, and data-driven approach to handling the complexities of waste streams in the modern era.

REFERNCES

Smith, J. A., & Brown, L. M. (2022). "Enhancing Waste Management Through Machine Learning: A Comprehensive Review." Journal of Environmental Technology, 45(3), 123-140.

Greenberg, R. S., Harper, M. E., & Thompson, P. Q. (2021). "Automated Segregation of Bio and Non-Biodegradable Waste: A Machine Learning Approach." International Journal of Sustainable Technology, 8(2), 67-82.

Patel, A. R., Chen, H. X., & Davis, E. S. (2020). "Advancements in Waste Classification: A Deep Learning Perspective." Journal of Clean Energy and Environmental Sustainability, 15(4), 321-336.

Johnson, K. D., Turner, R. A., & White, S. B. (2019). "Machine Learning Applications in Waste Sorting Technologies: A Comprehensive Analysis." Environmental Science and Technology, 38(7), 567-582.

Rodriguez, M. F., & Gupta, R. B. (2018). "Smart Solutions for Waste Segregation: An In-depth Study on ML-Based Systems." Waste Management and Research, 26(6), 510-525.