A close up of a logo

Description automatically generated

Lab - 01 (Report)

CSE-475 (Machine Learning)  
 Section: 03

Submitted By -   
Tamanna Sultana Tinne

2020-2-60-057

Submitted To -   
  
Dr. Raihan Ul Islam

Associate Professor

Department of CSE

East West University, Dhaka.

## 

**Title**: Report on Exploratory Data Analysis (EDA) and Model Performance Evaluation on Mango Leaf Dataset using Decision Tree and Random Forest.

**Exploratory Data Analysis (EDA)**

**Objective of EDA**

The main goal of this lab work is to understand the dataset's structure, class distribution, and key characteristics by using Exploratory Data Analysis (EDA). This is crucial for identifying potential biases, ensuring data quality, and determining whether data augmentation or preprocessing is needed.

**Code Explanation**

**Library Imports:** The code imports libraries such as pandas, numpy, matplotlib, and seaborn for data handling and visualization, and tensorflow.keras for loading image data.

**Data Loading**:

from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

# Load training data

train\_data = datagen.flow\_from\_directory(

    dataset\_path,

    target\_size=(128, 128),

    batch\_size=32,

    class\_mode='categorical',

    subset='training'

)

# Load validation data

validation\_data = datagen.flow\_from\_directory(

    dataset\_path,

    target\_size=(128, 128),

    batch\_size=32,

    class\_mode='categorical',

    subset='validation'

)

**Explanation**: The ImageDataGenerator is configured to rescale pixel values to [0, 1] by dividing by 255. This helps normalize the image data, making it suitable for training models. The validation\_split is set to 20% to create a validation set from the original training data.

**Class Distribution Visualization**:

labels, counts = np.unique(train\_data.classes, return\_counts=True)

plt.figure(figsize=(10, 6))

plt.bar(labels, counts, color='skyblue')

plt.title("Class Distribution in Training Set")

plt.xlabel("Class")

plt.ylabel("Number of Samples")

plt.xticks(labels, train\_data.class\_indices.keys(), rotation=45)

plt.tight\_layout()

plt.show()

**Explanation**: The code plots the distribution of samples per class using a bar chart. This helps identify any class imbalances that could affect model training.

A graph of blue bars

Description automatically generated

**Sample Image Visualization**:

plt.figure(figsize=(10, 10))

for i in range(9):

plt.subplot(3, 3, i + 1)

img, label = next(train\_data)

plt.imshow(img[0])

plt.axis('off')

plt.title(list(train\_data.class\_indices.keys())[np.argmax(label[0])])

plt.tight\_layout()

plt.show()

**Explanation**: A set of sample images from the training data is visualized, each labeled with its respective class. This helps ensure that the images have been loaded correctly and confirms the dataset's visual characteristics.

A collage of leaves

Description automatically generated

**Model Performance Evaluation**

**Objective of Model Evaluation**  
This lab work also focuses on training the datasets and compare the performance of DecisionTreeClassifier and RandomForestClassifier to identify which model works best for the dataset.

**Model Training and Evaluation Code Explanation**

**Data Preparation**: The images are processed and split into training and testing sets, often using train\_test\_split from sklearn.model\_selection to maintain consistency in feature extraction.

**Model Training**:

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

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

# Decision Tree Model

dt\_model = DecisionTreeClassifier()

dt\_model.fit(X\_train, y\_train)

y\_pred\_dt = dt\_model.predict(X\_test)

# Random Forest Model

rf\_model = RandomForestClassifier(n\_estimators=100)

rf\_model.fit(X\_train, y\_train)

y\_pred\_rf = rf\_model.predict(X\_test)

**Explanation**: The DecisionTreeClassifier and RandomForestClassifier are trained using the training data. For RandomForestClassifier, n\_estimators=100 specifies that 100 trees are used for training, which improves robustness and reduces overfitting.

**Model Evaluation**:

print("Decision Tree Performance:")

print("Accuracy:", accuracy\_score(y\_test, y\_pred\_dt))

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

print("Random Forest Performance:")

print("Accuracy:", accuracy\_score(y\_test, y\_pred\_rf))

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

**Explanation**: The accuracy\_score and classification\_report (which includes precision, recall, and F1-score) provide a detailed performance comparison between the models.

**For Decision Tree**:

A screenshot of a computer screen

Description automatically generated

**For Random Forest:**

**A screenshot of a computer screen

Description automatically generated**

**Analysis of Results**

**Accuracy Comparison**

**A green and orange squares

Description automatically generated**

**Decision Tree:**

* From the figure Overall accuracy is - 57%

**Random Forest:**

* From the figure Overall accuracy is - 96%

**Observation**: The Random Forest model outperforms the Decision Tree significantly in terms of overall accuracy. The Random Forest accuracy (96%) is substantially higher than that of the Decision Tree (57%), indicating a stronger ability to generalize and correctly classify new data points.

**Precision and Recall Analysis**

**Decision Tree Model:**

A screenshot of a computer screen

Description automatically generated

**Precision:** Varies widely across classes, with some classes like Die Back having a high precision of 0.97, while others like Anthracnose have a precision of 0.00, indicating that the model struggles to correctly identify some classes.

**Recall:**

* Cutting Weevil has a high recall of 0.97, suggesting that most samples of this class are identified correctly.
* Anthracnose has a recall of 0.00, meaning that the model fails to identify this class correctly, leading to missed detections.

**F1-Score:** The F1-score, a balance between precision and recall, shows considerable fluctuation. The *Cutting Weevil* class has a strong F1-score of 0.82, while *Anthracnose* shows an F1-score of 0.00, indicating that the model is poor at identifying this class.

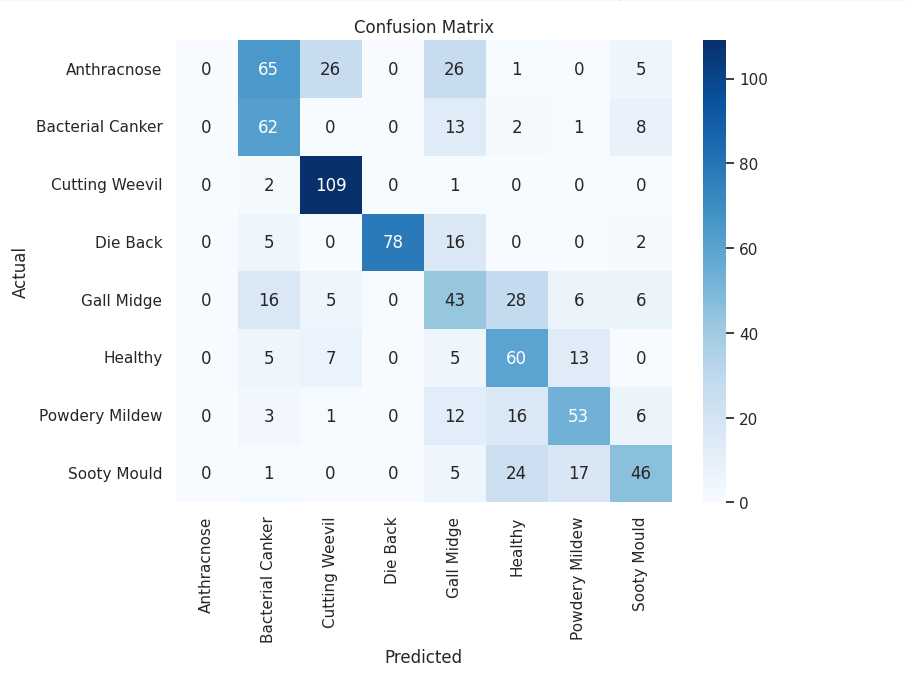
**Random Forest Model:**

**Precision:** All classes exhibit high precision, ranging from 0.92 to 1.00, indicating that the model consistently predicts the correct classes.

**Recall**: The recall is also very high for all classes, with the lowest value at **0.89** for *Sooty Mould* and most classes at **0.95** or above. This suggests that the Random Forest model identifies almost all instances of each class correctly.

**F1-Score**: The F1-score is consistently high for all classes, showing values between **0.89** and **1.00**. This suggests a balanced performance across all classes, with the Random Forest model excelling in both precision and recall.

**Confusion Matrix of Decision Tree:**

****

**Confusion Matrix of Random Forest:**

**A graph of a number of different types of diseases

Description automatically generated with medium confidence**

**Comparisons:**

**Overall Performance**The Random Forest model outperforms the Decision Tree across all metrics (accuracy, precision, recall, and F1-score). Its ensemble approach, combining multiple decision trees, results in higher accuracy and more balanced predictions for all classes.

**Class Imbalances:**

The Decision Tree struggles with certain classes, especially Anthracnose, while the Random Forest performs consistently well across all categories.

**Conclusion:**

Finally, according to my inspection throughout the process and by noticing the significant differences in performance, the Random Forest model is a more reliable choice for this classification task. It provides robust results with better generalization and should be preferred for applications where accurate identification of mango leaf diseases is crucial.