Faculty of Engineering and Technology.

Department of Electrical and Computer Engineering.

ENCS3340- Artificial Intelligence

# Image Classification Using Machine Learning

Project No.2

Prepared by: Noura Khdour.

ID:1212072.

Instructor: Dr.Yazen Abu Farha

Section:4

Date:30/6/2025

Birzeit-June

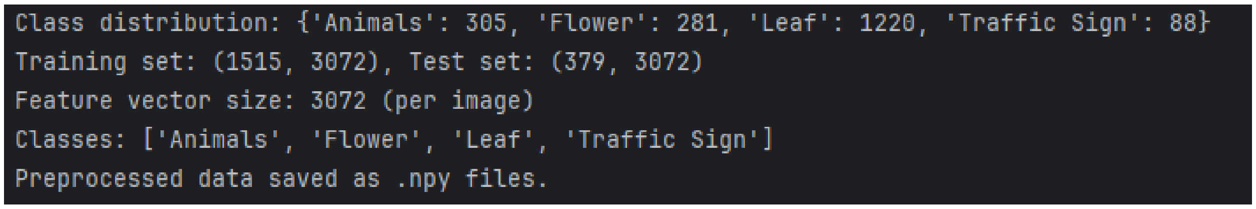
## ❖ Introduction.

This project aims to compare the performance of three different machine learning models: Naive Bayes, Decision Tree, and Feedforward Neural Network (MLPClassifier) for the task of image classification. I use a dataset consisting of labeled images from multiple categories, such as animals, traffic signs, leaves and flowers. Each image is resized, preprocessed, and converted into a numerical format suitable for model training. By evaluating these models using metrics like accuracy, precision, recall, and F1-score, I aim to understand the strengths and weaknesses of each approach and how well they generalize to new ,unseen data. The models were implemented using the Scikit-learn library.

## ❖ Dataset Description.

The dataset used in this project consists of labeled images categorized into four distinct classes**:** **Animals**, **Traffic Signs**, **Leaves and Flower**. The images were collected from open-source datasets called "Kaggle" and organized into folders, where each folder represents a single class.Before training, each image was:

- Resized to a fixed dimension of **32×32** pixels to ensure uniformity across the dataset, since we have more than 500 images, **32x32** is more manageable and less demanding on memory and processing power.
- Converted to **RGB** format, so each image is represented with three color channels (Red, Green, Blue).
- Flattened into a **1D** numerical vector of pixel values and normalized to range between 0 and 1, making it suitable for machine learning models.
- Split the dataset into **80% training** that used to train the models and **20% testing** used to evaluate how well the models generalize to new, unseen images.
- Each image was resized to **32×32 pixels** with **3 color channels (RGB)**, resulting in a flattened feature vector of **3072** values **(32 × 32 × 3)** for input to the machine learning models.



```
Class distribution: {'Animals': 305, 'Flower': 281, 'Leaf': 1220, 'Traffic Sign': 88}
Training set: (1515, 3072), Test set: (379, 3072)
Feature vector size: 3072 (per image)
Classes: ['Animals', 'Flower', 'Leaf', 'Traffic Sign']
Preprocessed data saved as .npy files.
```

Figure 1:Dataset summary

Figure 1 shows the class name and the number of images inside it, also the number of training images and number of testing images. The preprocessed data (flattened, normalized images with labels and training and testing set) was saved to **.npy** files to avoid repeating the preprocessing steps and to speed up model training and testing. The total image in my data set is **1,894** images.

## ❖ Naive Bayes Model.

Naive Bayes is a simple yet effective classification algorithm based on Bayes' Theorem with the assumption that all features are independent given the class label. In this project, the Naive Bayes classifier was applied to the image dataset by treating each pixel as an individual feature. Despite its simplicity, Naive Bayes performs surprisingly well for high-dimensional data like images. I was used Gaussian Naive Bayes as a baseline model to evaluate performance against more complex classifiers. The model was trained on the processed feature vectors and tested for accuracy, precision, and recall, offering a fast and interpretable starting point for comparison.

- **Results**

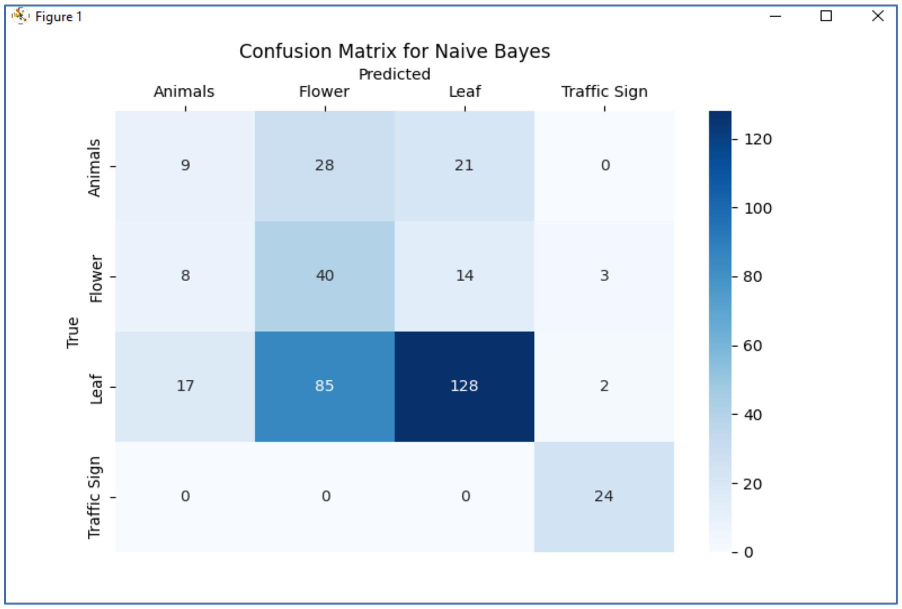
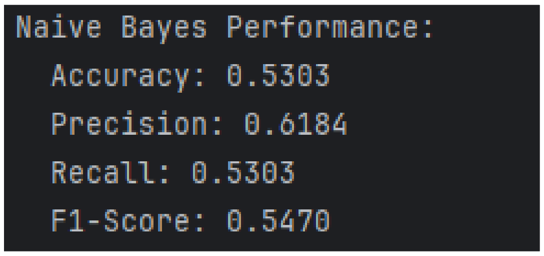


Figure 2: Confusion matrix for NB.

The image shows that the model performs well on leaf class with **128** correctly predicted as Leaf, also it confuses flower (predicted) and leaf (true) classes with each other (misclassification) with **85** misclassified and so on. The zeros indicate no images classified. If we sum all the numbers in cells which equal to 379 we found it is equal to testing set images.



```
Naive Bayes Performance:
  Accuracy: 0.5303
  Precision: 0.6184
  Recall: 0.5303
  F1-Score: 0.5470
```

Figure 3: Performance for NB

- **Visualizing Test Images.**

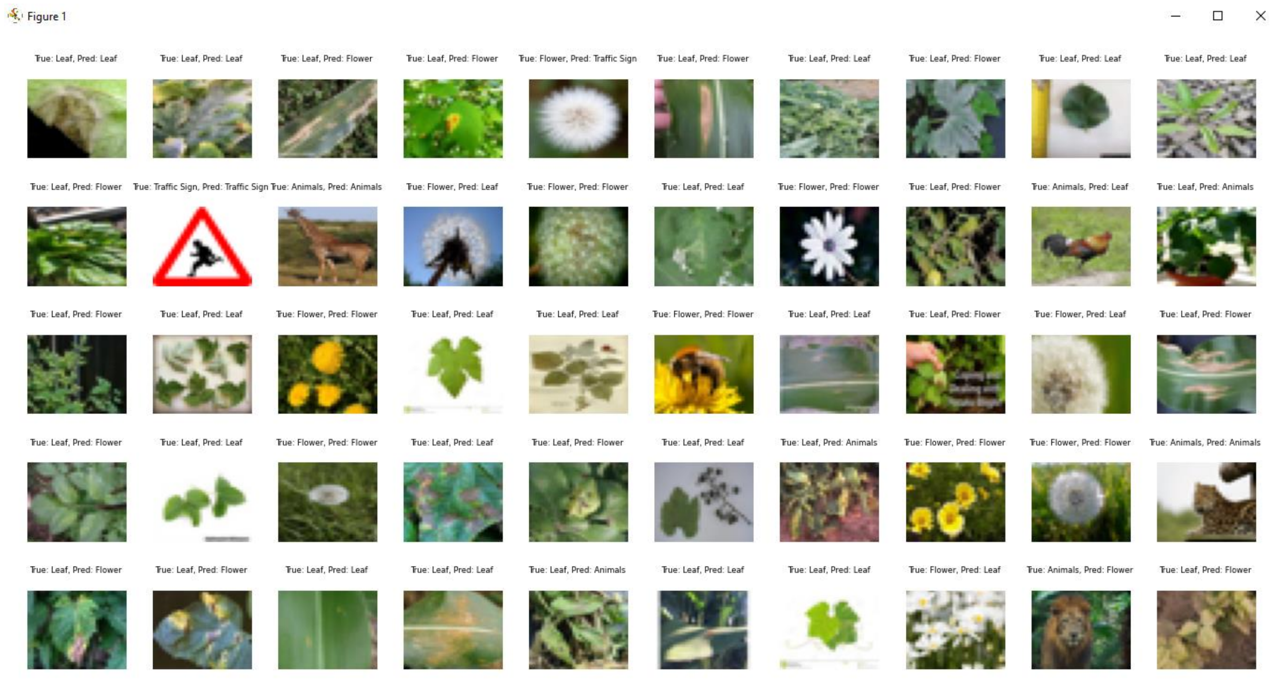


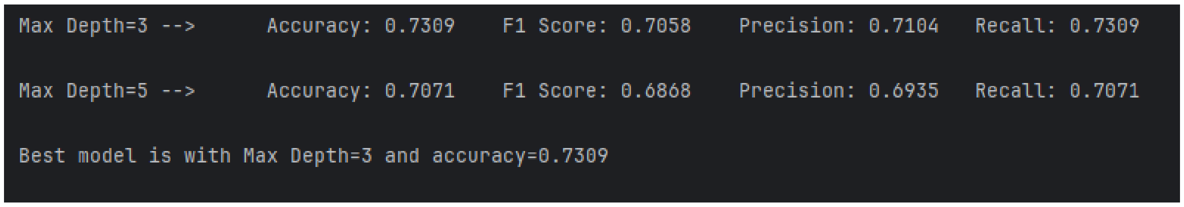
Figure 4:Test images for NB

To better understand the model's predictions, a sample of test for 50 images was displayed alongside their **true labels** and **predicted classes**. This visualization helps identify patterns in misclassification and observe how well the model distinguishes between different image categories.

## ❖ Decision Tree Model.

A decision tree is a flowchart-like tree structure where an internal node represents a feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. In this project, images were converted into numerical feature vectors and passed to the model for training. In Scikit-learn, **optimization of decision tree** classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning, so I use a hyperparameters called **max_depth_values** = [3,5] with two values of max depth and upon to the result we decide which depth is best. The reason why I choose theses two values for depth is to make the model simpler and easier to interpret, avoid the overfitting if the value is higher and avoid the underfitting if the value is very low. Other than pre-pruning parameters, I also try other attribute selection measure such as **entropy**.

- **Results**



```
Max Depth=3 -->      Accuracy: 0.7309    F1 Score: 0.7058    Precision: 0.7104    Recall: 0.7309

Max Depth=5 -->      Accuracy: 0.7071    F1 Score: 0.6868    Precision: 0.6935    Recall: 0.7071

Best model is with Max Depth=3 and accuracy=0.7309
```

Figure5: Performance for DT

As shown in figure 5 that at max depth=3 give the best accuracy due to reduced overfitting. Simpler models often generalize better. A smaller depth helps the model focus on the core patterns, not the noise.

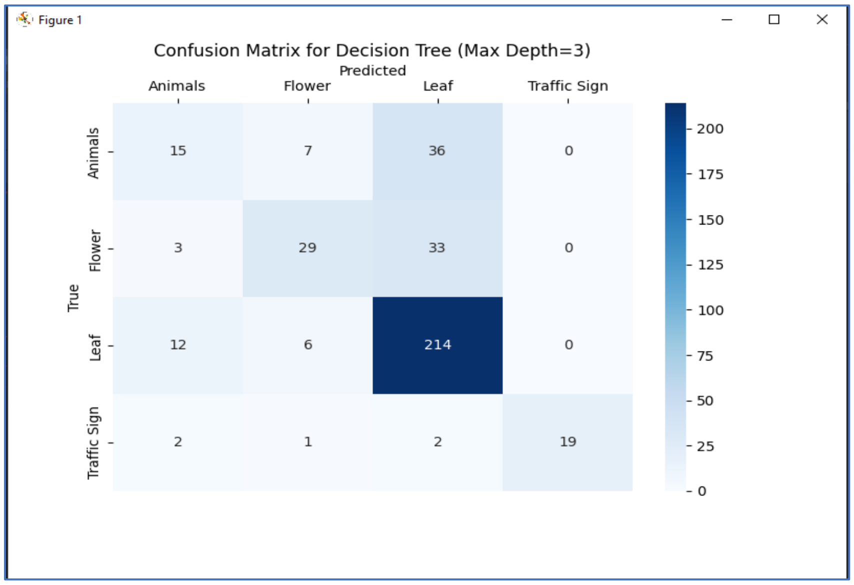


Figure 6: Confusion matrix for DT

The image shows that the model performs well on leaf class with **214** correctly predicted as Leaf, it confuses Animals (true) and leaf (predicted) classes with each other (misclassification) with **36** misclassified and also it confuse Animals (predicted) and traffic sign (true) with each other (misclassification) with **2** misclassified and so on. The zeros indicate no images classified.

- **Visualizing Test Images.**

As shown below in figure 7, The images in the red box are a **<u>sample</u>** of how the decision tree classified them and if you compare them with naïve bayes you will see a different.

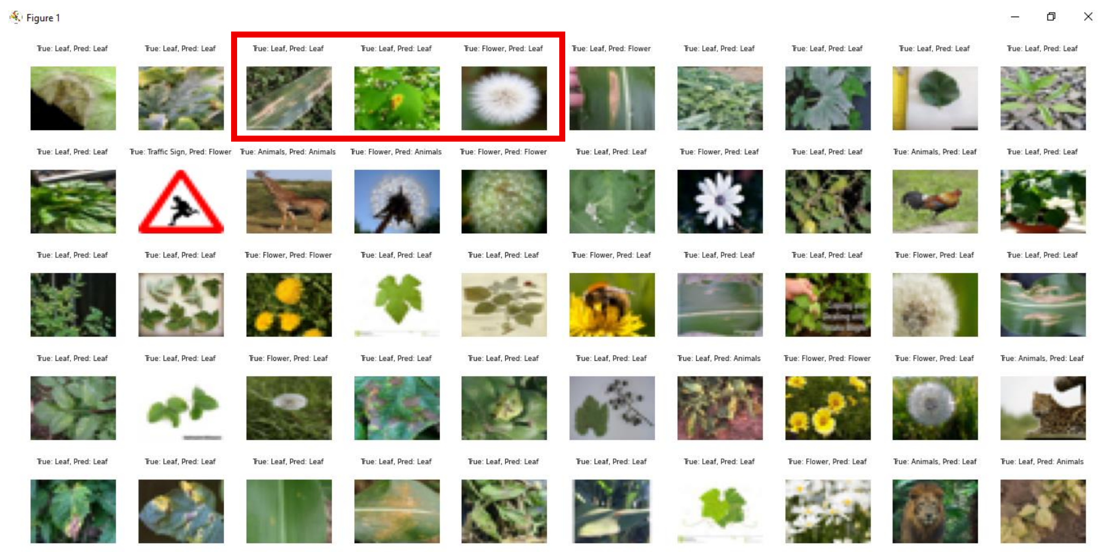


Figure 7:Test images for DT

- **Visualizing tree path.**

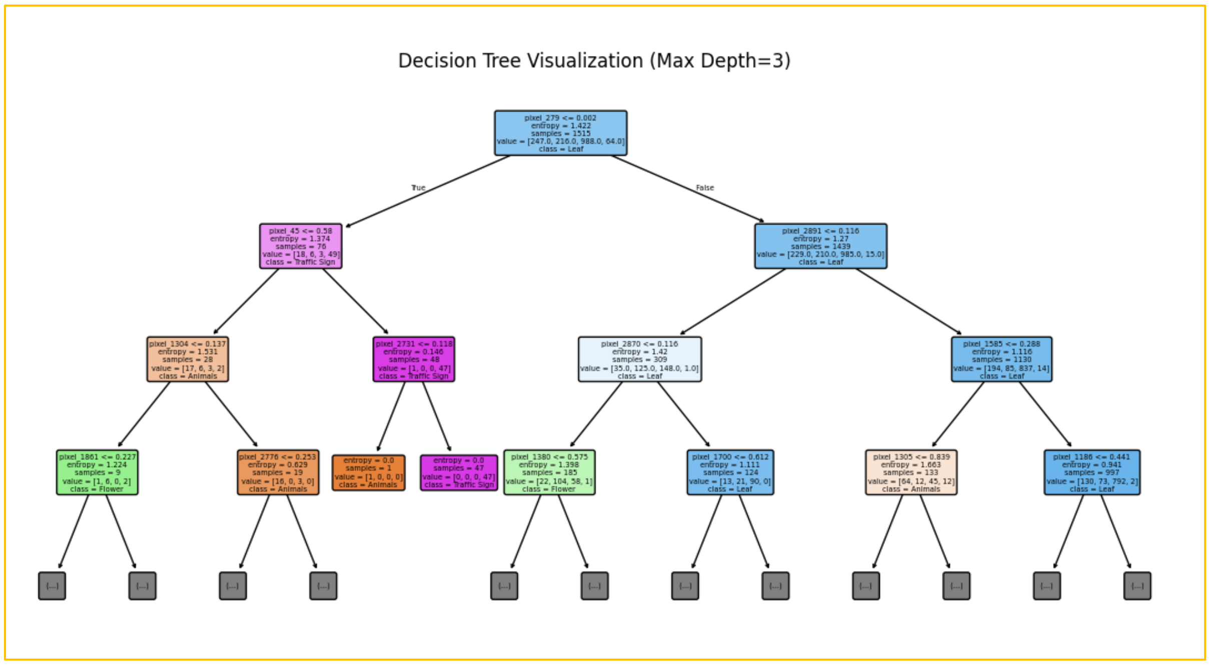


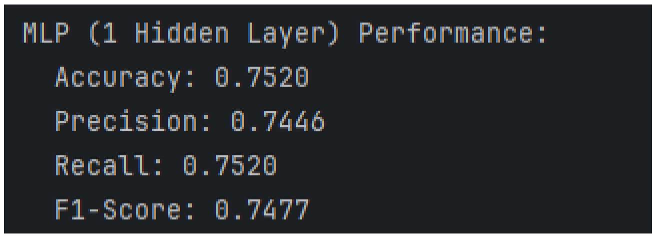
Figure 8:path for DT

As you can see, this pruned model is less complex, more explainable, and easier to understand. Each internal node represents a decision based on a specific pixel's intensity value, while each leaf node represents a final class prediction. The decision tree splits the dataset by evaluating whether the pixel intensity is below or above certain thresholds, The visualization also displays the entropy (a measure of impurity), the number of samples, and the distribution of class labels at each node.

## ❖ MLP Classifier (1-Hidden layer).

In this project, a Multi-Layer Perceptron (MLPClassifier) was used as a feedforward neural network model for image classification. The network was configured with **one hidden layer** containing **100** neurons, using the **ReLU** activation function and the **Adam** optimizer for training with max_iter=300. The input to the MLP consists of flattened image vectors, allowing the model to learn from pixel intensities. MLP is capable of learning non-linear patterns and more complex relationships in the data. Although training requires more computational time and careful tuning. the MLP achieved higher performance metrics across accuracy, precision, recall, and F1-score, indicating its ability to generalize better on unseen data.

- **Results**



```
MLP (1 Hidden Layer) Performance:
  Accuracy: 0.7520
  Precision: 0.7446
  Recall: 0.7520
  F1-Score: 0.7477
```

Figure9: Performance for MLP1

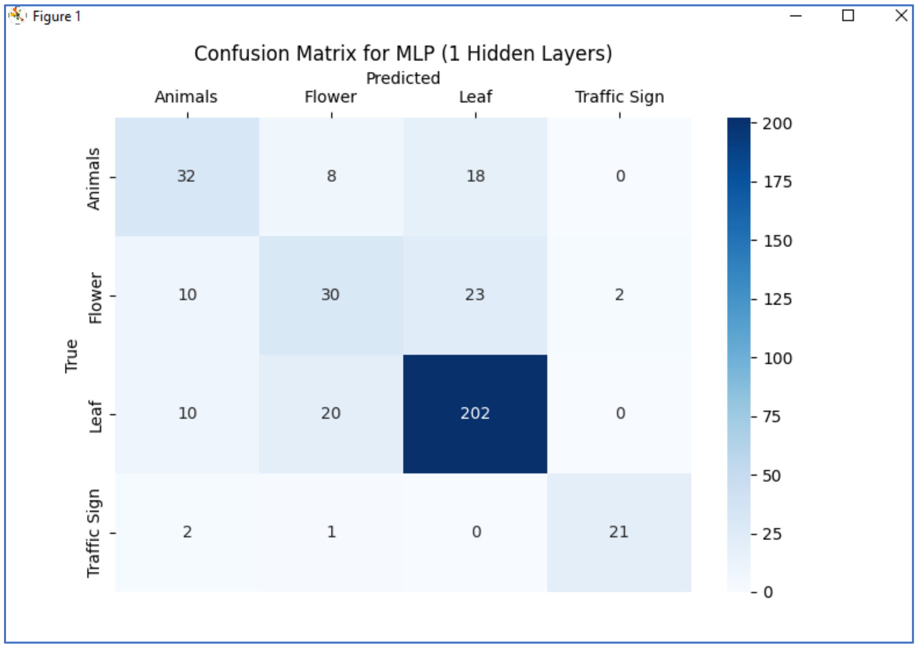


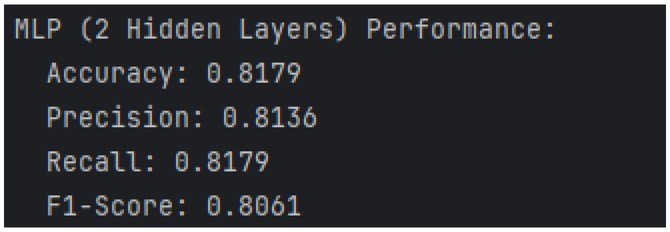
Figure 10: Confusion matrix for MLP1

As shown above in figure 10 , model performs well on leaf class with **202** correctly predicted as Leaf, performs well on Animals class with **32** correctly predicted as Animals, performs well on Traffic Sign class with **21** correctly predicted as Traffic Sign and performs well on Flowers class with **30** correctly predicted as Flowers and so on.

## ❖ MLP Classifier (2-Hidden layers).

In addition to the single-layer model, a Multi-Layer Perceptron (MLPClassifier) with **two hidden layers** was implemented to explore deeper neural network architectures. The model was configured with hidden layer sizes of **(100, 50)** neurons. This deeper structure allows the model to capture complex patterns that simpler models might miss. The **ReLU** activation function and **Adam** optimizer were used to facilitate efficient training with max_iter=300. The two-layer MLP showed improvements in classification accuracy and generalization, particularly on complex.

- **Result**



```
MLP (2 Hidden Layers) Performance:
  Accuracy: 0.8179
  Precision: 0.8136
  Recall: 0.8179
  F1-Score: 0.8061
```

Figure11: Performance for MLP2

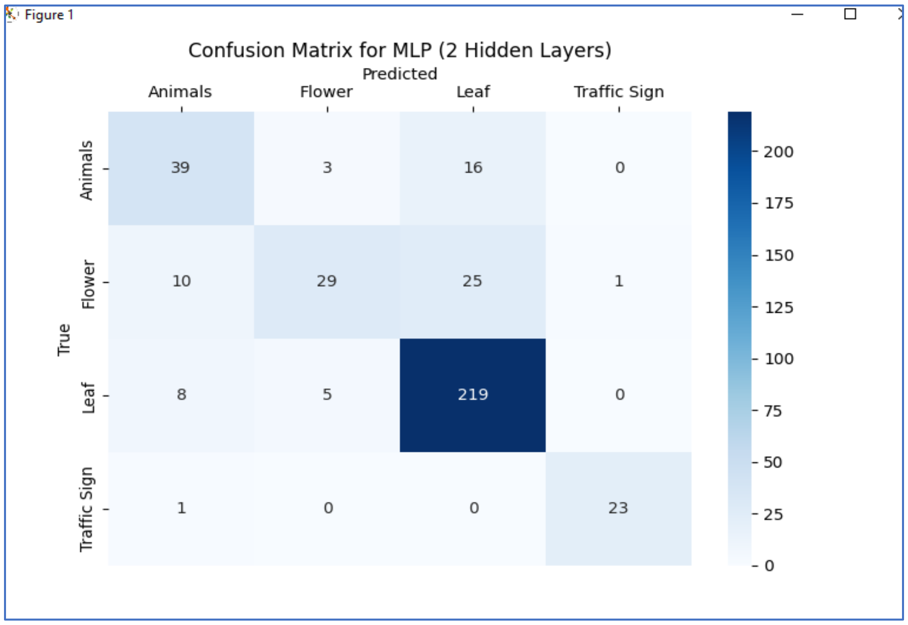


Figure 12: Confusion matrix for MLP2

As shown above in figure 12 , model performs well on leaf class with **219** correctly predicted as Leaf, performs well on Animals class with **39** correctly predicted as Animals, it confuses Flowers (predicted) and leaf classes (true) with each other (misclassification) with **5** misclassified and also it confuse traffic sign (predicted) and Flowers (true) with each other (misclassification) with **1** misclassified and so on.

## ❖ Compare Models Performance.

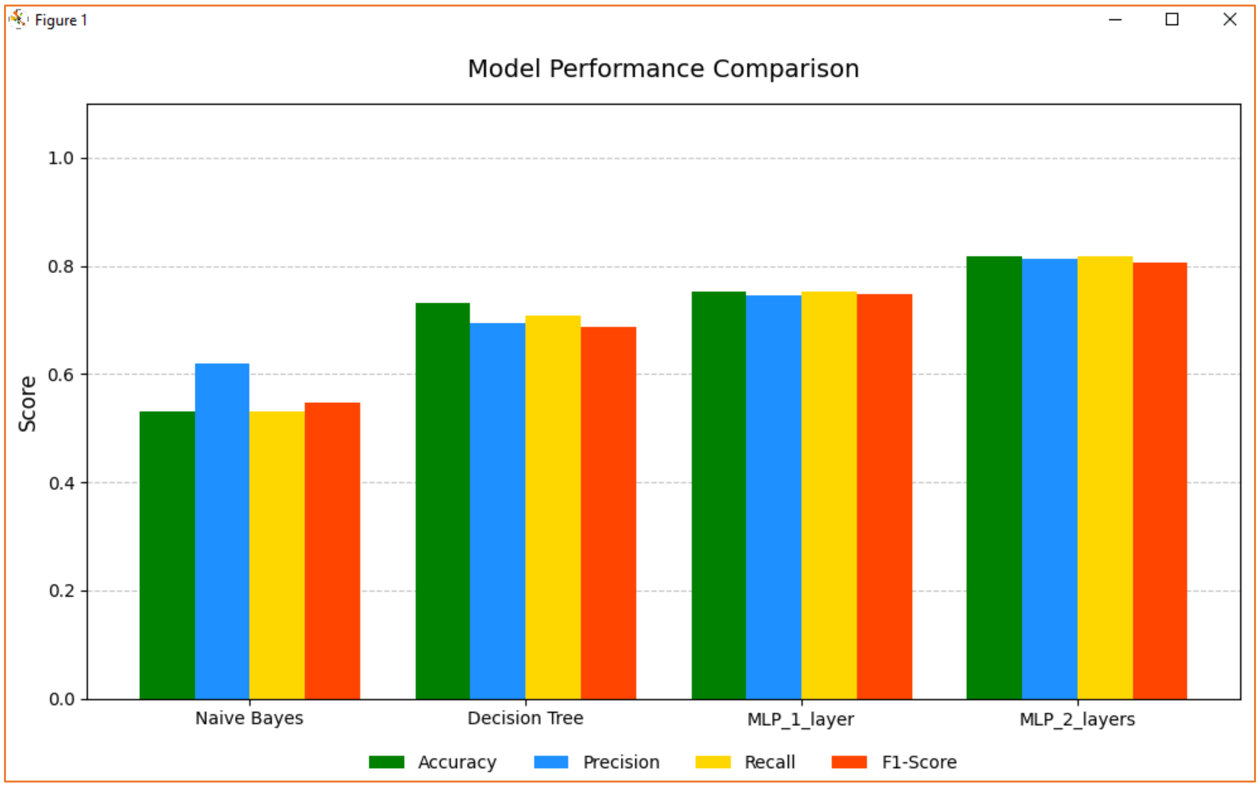


Figure 13: compare performance

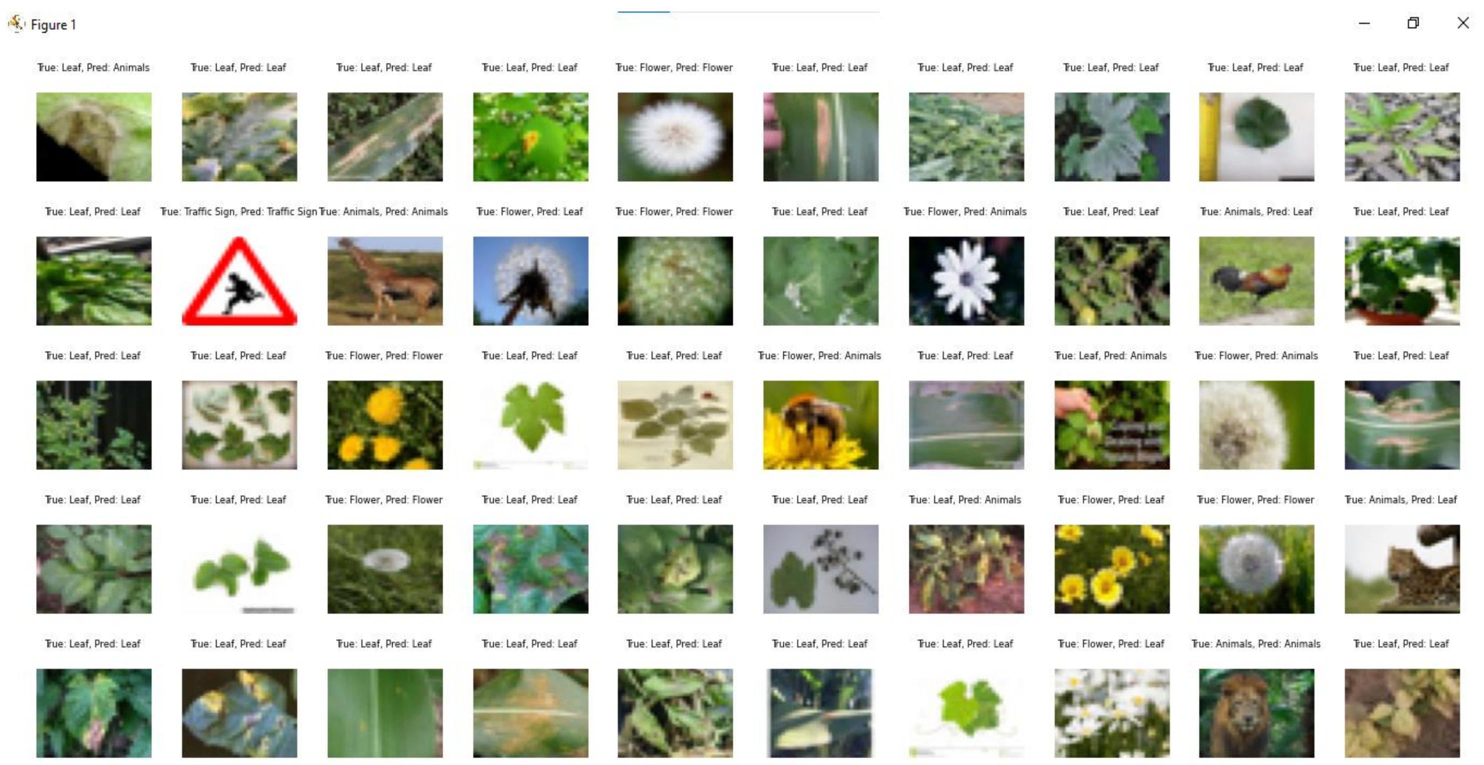
As shown above in figure 13:

- **Naive Bayes:** showed the lowest performance across all metrics. This is expected, as it makes strong independence assumptions and uses simple statistical features.
- **Decision Tree:** outperformed Naive Bayes, particularly in accuracy and recall. However, its performance remained moderate.
- **MLP with 1 hidden layer:** performed better than both Naive Bayes and Decision Tree models. It was able to capture nonlinear relationships within the pixel data, leading to improvements in all metrics.
- **MLP with 2 hidden layers:** achieved the **highest performance** across all metrics. The deeper architecture enabled the network to learn more abstract and hierarchical features from the input images, improving its generalization to unseen data.

Best Performing Model is **MLP (2 Hidden Layers)** due to it had enough depth to learn complex features and it benefited from backpropagation and the use of nonlinear activations (e.g., ReLU).

## ❖ Appendix

**Visualizing Test Images for MLP with 2 hidden layers.**



THE END 😊