Faculty of Engineering & Technology

Electrical & Computer Engineering Department

Artificial intelligent

ENCS 3340

Project 2

Comparative Study of Image Classification Using Decision Tree, Naive Bayes, and Feedforward Neural Networks

Prepared by:

Name: Naeem Mrabie Number:1210787

Partner Name: Tariq Zaid Number: 1212974

Instructor: Dr. Samah Alaydi

Section:3

Date: 30/6/2025

Abstract:

This project evaluates three machine learning methods—Decision Tree, Naive Bayes, and Neural Network—for classifying animal images. We use a set of 780 pictures sorted into six groups: bird, cat, cow, dog, lion, and panda. Each picture is resized to (64×64) pixels, converted into a single row of pixel values, and split into 624(80%) images for training and 156(20%) for testing. Naive Bayes offers a quick, baseline approach by assuming pixel values is independent. Decision Trees build simple rules that clearly show how each image is classified based on key pixels. The Neural Network MLPClassifier explores deeper patterns by learning hidden representations across one or two layers. We assess each model using accuracy, precision, recall, F1-score, and confusion matrices to understand both overall performance and specific class strengths or weaknesses. Our findings highlight trade-offs: Naive Bayes is fastest but less precise; Decision Trees are easy to interpret yet can overfit; Neural Networks achieve the highest accuracy at the cost of longer training. We used python programming language, Kaggle for collecting images.

Table of content

Contents

Table of Figure:

Table of table:

1. Introduction:

Image classification means teaching a computer to recognize what's in a picture like Whether an image shows a cat, dog, cow, bird, lion, and panda. This project compares three popular ways to do this:

- Naïve Bayes classifier: a fast basic method that uses probability.
- Decision Tree classifier: makes decision by following simple (like Entropy) rules, so it easy to understand.
- Neural Network: A more advance technique that can find complex patterns.

The objective is to analyze each method's strengths and limitations, particularly for high -dimensional data such as images.

2. Dataset Description and preprocessing

2.1 Dataset:

We created a dataset with 780 images, sorted into six animal classes: bird, cat, dog, cow, lion, and panda. Each class has about 130 images. For each animal, 104 images are used to train the model and 26 images are used to test it. This gives enough variety to properly evaluate how well each model works.

2.2 Preprocessing steps:

- **Resizing**: All images were resized to 64x64 pixels. By resizing all images to the same dimensions, the model learns from consistent input features.
- **Flattening**: Each image is converted to a 1D array of pixel values (length:4096).
- **Splitting**: Data split into 80% training (624 images) and 20% testing (156 images).
- **CSV File Organization:** All image file paths and their corresponding class labels were stored in a CSV file with two columns: image path and label.
- **Normalization**: pixel values scaled to [0,1] range for neural network training.

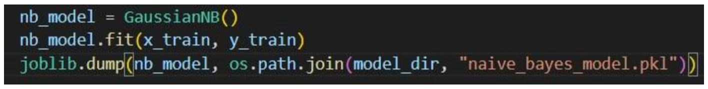
3. Model Implementation:

3.1 Naïve Bayes Classifier:

The Naïve Bayes approach treats each pixel as an independent feature. For each class, it estimates thee mean and variance of each pixel across the training set, and classifies new image by calculating the probability of observing their pixel values given in each class.

Key properties:

- Fast training /prediction
- Assume pixel independence
- Implementation: used GaussianNB from scikit learn.



```
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
joblib.dump(nb_model, os.path.join(model_dir, "naive_bayes_model.pkl"))
```

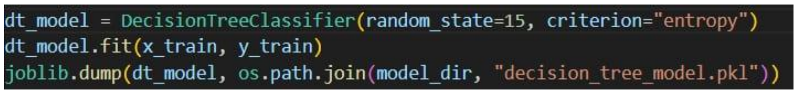
Figure 1: implementation Naive Bayes classifier

3.2 Decision Tree Classifier:

The decision Tree learns a hierarchy of rules based on pixel intensities, splitting the data to maximize class separation at each node, the Decision tree used for training Recursive divide and concur approach.

Key properties:

- Highly interpretable
- Can overfit high dimensional data
- Implementation: Used DecisionTreeClassifier from scikit -learn.



```
dt_model = DecisionTreeClassifier(random_state=15, criterion="entropy")
dt_model.fit(x_train, y_train)
joblib.dump(dt_model, os.path.join(model_dir, "decision_tree_model.pkl"))
```

Figure 2: implementation Decision Tree

3.3 Neural Network (MLP Classifier):

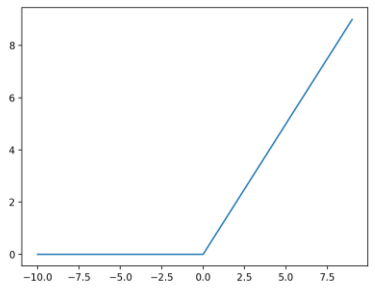
A Feedforward Network (Multi – layer Perceptron, or MLP) can model complex, non – linear relationships between pixel data and class label by learning deep representation.

Key properties:

- Can achieve higher accuracy compared to simpler models
- Requires more training time and careful parameter tuning.

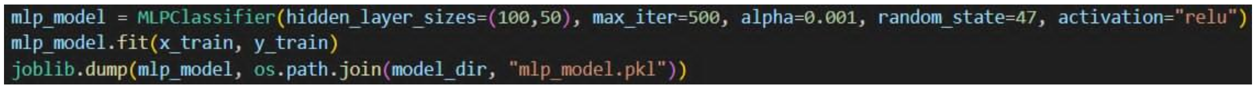
Implementation details:

- Used MLPClassfier from scikit learn
- Architecture: in the first step we using one hidden layer size = 50 neuron, and then we update to a two hidden layer in the first layer=100 neuron and in the second layer 50 neuron for better efficiency.
- Learning rate: 0.001
- Maximum iteration: 500
- Random state: 47 (for reproducibility)
- Activation Function: Relu (Rectifier Linear unit) for negative value small multiplying factor and for positive value the value of input.



$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases}$$

Figure 3: Rectifier linear unit activation function plot using MATLAB and how it is exactly work



```
mlp_model = MLPClassifier(hidden_layer_sizes=(100,50), max_iter=500, alpha=0.001, random_state=47, activation="relu")
mlp_model.fit(x_train, y_train)
joblib.dump(mlp_model, os.path.join(model_dir, "mlp_model.pkl"))
```

Figure 4: implementation Neural Network (MLP classifier)

4. Evaluation Metrics:

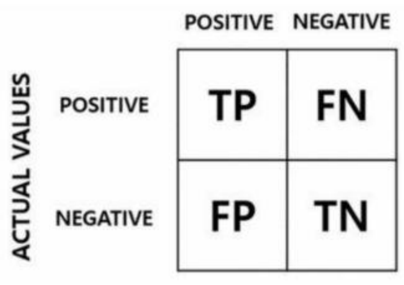
a- **Accuracy**: is the number of correctly classified positive and negative examples divide by the total number of examples in the test set. Is not suitable in some applications.

b- **Precision:** is the number of correctly classified positive examples divide by the total number of examples that are classified as positive.

c- **Recall:** is the number of correctly classified positive examples divide by the total number of actual positive example in the test set.

d- **F1-Score:** is the harmonic mean of precision and recall**.**

e- **Confusion Matrix:** a table (2-D Array) that summarize the performance of classification model by showing how many predictions were correct and incorrect, broken down by class.



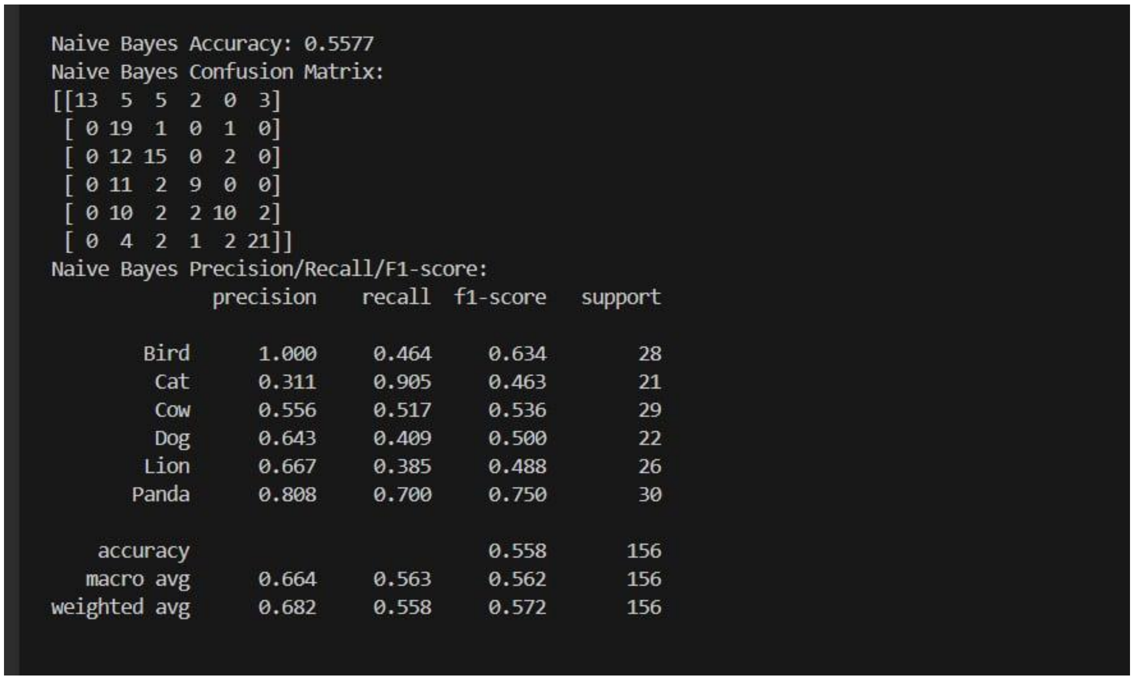
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Figure 5: Evaluation Metric Summary

4.1 Accuracy, precision, Recall, f1-score (Naïve Bayes)



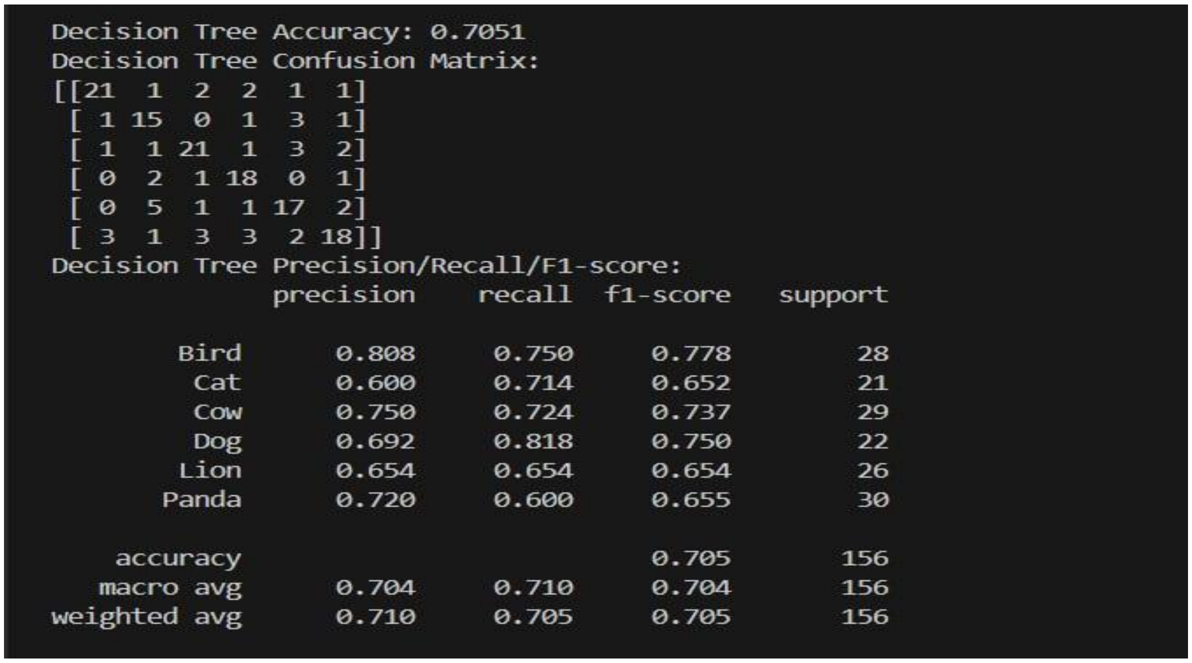
```
Naive Bayes Accuracy: 0.5577
Naive Bayes Confusion Matrix:
[[13  5  5  2  0  3]
 [ 0 19  1  0  1  0]
 [ 0 12 15  0  2  0]
 [ 0 11  2  9  0  0]
 [ 0 10  2  2 10  2]
 [ 0  4  2  1  2 21]]
Naive Bayes Precision/Recall/F1-score:
              precision    recall  f1-score   support

        Bird      1.000     0.464     0.634        28
         Cat      0.311     0.905     0.463        21
         Cow      0.556     0.517     0.536        29
         Dog      0.643     0.409     0.500        22
        Lion      0.667     0.385     0.488        26
       Panda      0.808     0.700     0.750        30

    accuracy                          0.558       156
   macro avg      0.664     0.563     0.562       156
weighted avg      0.682     0.558     0.572       156
```

Figure 6: Detailed Evaluation Metrics (Naive Bayes)

4.2 Accuracy, precision, Recall, f1-score (Decision Tree)



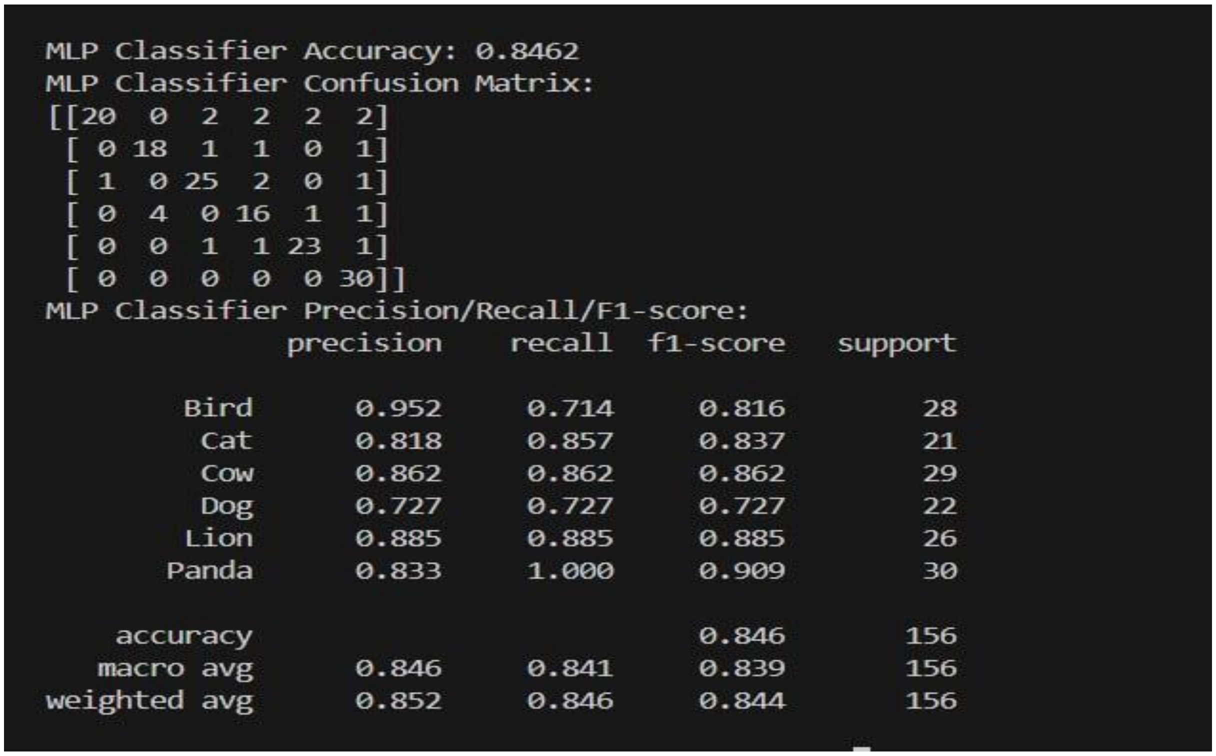
```
Decision Tree Accuracy: 0.7051
Decision Tree Confusion Matrix:
[[21  1  2  2  1  1]
 [ 1 15  0  1  3  1]
 [ 1  1 21  1  3  2]
 [ 0  2  1 18  0  1]
 [ 0  5  1  1 17  2]
 [ 3  1  3  3  2 18]]
Decision Tree Precision/Recall/F1-score:
              precision    recall  f1-score   support

        Bird      0.808     0.750     0.778        28
         Cat      0.600     0.714     0.652        21
         Cow      0.750     0.724     0.737        29
         Dog      0.692     0.818     0.750        22
        Lion      0.654     0.654     0.654        26
       Panda      0.720     0.600     0.655        30

    accuracy                          0.705       156
   macro avg      0.704     0.710     0.704       156
weighted avg      0.710     0.705     0.705       156
```

Figure 7: Detailed Evaluation Metrics (Decision Tree)

4.3 Accuracy, precision, Recall, f1-score (Neural Network MLP Classifier)



```
MLP Classifier Accuracy: 0.8462
MLP Classifier Confusion Matrix:
[[20  0  2  2  2  2]
 [ 0 18  1  1  0  1]
 [ 1  0 25  2  0  1]
 [ 0  4  0 16  1  1]
 [ 0  0  1  1 23  1]
 [ 0  0  0  0  0 30]]
MLP Classifier Precision/Recall/F1-score:
              precision    recall  f1-score   support

        Bird      0.952     0.714     0.816        28
         Cat      0.818     0.857     0.837        21
         Cow      0.862     0.862     0.862        29
         Dog      0.727     0.727     0.727        22
        Lion      0.885     0.885     0.885        26
       Panda      0.833     1.000     0.909        30

    accuracy                          0.846       156
   macro avg      0.846     0.841     0.839       156
weighted avg      0.852     0.846     0.844       156
```

Figure 8: Detailed Evaluation Metrics (MLP Classifier)

5. Discussion:

5.1 Naïve Bayes

This method is super-fast and easy to use. It learns almost instantly. However, it's not very accurate because it's too simple and assumes that different parts of the information don't affect each other. It's usually better for things like sorting emails or medical diagnoses, not directly for pictures. But, if you turn pictures into simple data like color information, it can still work. It's good at finding all the correct items, but sometimes it might incorrectly flag something as correct.

5.2 Decision Tree

Decision Trees are also quick and easy to understand. They can make good decisions, but they don't get much better with more data. They can also become too complex and start learning from mistakes if not kept in check. Like Naïve Bayes, they aren't great for raw images. You need to simplify the image data first. They are good at explaining why they made a decision and can handle complex rules. They are often used for things like medical decisions. They are good at being precise (getting things right when they say it's right), but might miss some correct items.

5.3 Neural Network

Neural Networks are the most powerful but also the hardest to set up and train. They take a lot of time and computer power. However, they give the best results because they can learn very complex patterns that the other methods can't. They are used for many things like finding fraud, understanding speech, and classifying images (again, if the images are simplified into data). They are good at balancing between being precise and finding all the correct items, especially when you set them up well and give them enough data. They are generally better than Naïve Bayes at being precise and better than simple Decision Trees at finding all the correct items

6. Library used in the project:

Table 1: Library used in the project

Library	Used for
OS	Handles file and directory operations (creating folders, joining paths, renaming files)
csv	Reads from and writes to CSV files to store image paths and labels.
Cv2	Processes images: reading, resizing, color conversion, and feature extraction (histograms).
numpy	Performs efficient numerical operations and handles arrays for machine learning input.
joblib	Saves and loads trained models (serialization) without retraining.
matplotlib. pyplot	Visualizes the decision tree using plotting functions.
train_test_split	Splits the dataset into training and testing sets for model evaluation.
accuracy_score, confusion_matrix, classification_report	Evaluates model performance using accuracy and classification metrics.
GaussianNB	A probabilistic Naïve Bayes classifier used as a simple baseline model.
DecisionTreeClassifier	A rule-based classifier that builds a decision tree from the training data.
plot_tree	Visualizes the structure of the decision tree model.
MLPClassifier	Implements a neural network to learn complex non-linear relationships in the data.

7. Reference:

[1] Scikit-learn Developers, “Gaussian Naive Bayes,” *Scikit-learn Documentation*, https://scikit-learn.org/stable/modules/naive_bayes.html, Accessed: June 2025.

[2] Scikit-learn Developers, “Decision Trees,” *Scikit-learn Documentation*, https://scikit-learn.org/stable/modules/tree.html, Accessed: June 2025.

[3] Scikit-learn Developers, “MLPClassifier — Multi-layer Perceptron,” *Scikit-learn Documentation*, https://scikit-learn.org/stable/modules/neural_networks_supervised.html, Accessed: June 2025.

[4] J. Brownlee, “A Gentle Introduction to Naive Bayes for Machine Learning,” *Machine Learning Mastery*, 2019. [Online]. Available: https://machinelearningmastery.com/naive-bayes-for-machine-learning/

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[6] R. Rokach and O. Maimon, “Top-Down Induction of Decision Trees Classifiers – A Survey,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 4, pp. 476–487, 2005.