Faculty of Engineering & Technology

Department of Electrical and Computer Engineering

ENCS3340: Artificial Intelligence

Project Report

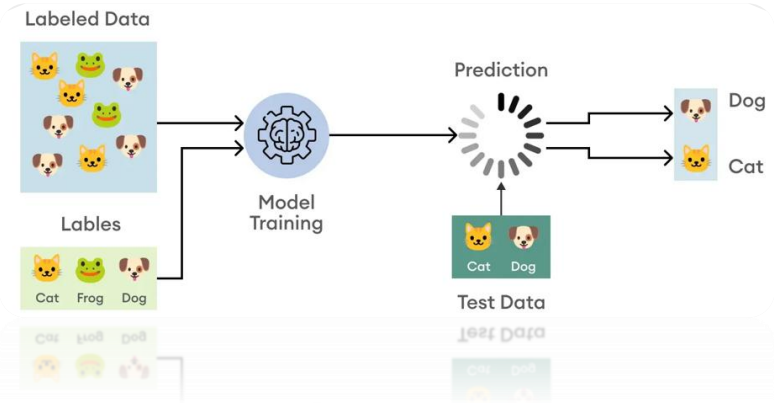
Comparative Study of Image Classification Using Decision Tree, Naive Bayes, and Feedforward Neural Networks

---

Name: Noureddin Etkaidek **ID:** 1220162

Name: Fadi Bassous **ID:** 1221005

Instructor: Yazan Abu Farha



Section: 2.

Date: 28/1/2025.

Introduction

Image classification is a fundamental task in computer vision that involves assigning a category label to an image based on its visual content. It plays a critical role in many real-world applications such as object recognition, autonomous driving, medical imaging, and security systems. With the rise of machine learning, various models have been developed to tackle this problem, each with different strengths in terms of performance, complexity, and interpretability.

This report presents a comparative study of three classification models — Naive Bayes, Decision Tree, and Multi-Layer Perceptron (MLPClassifier) — applied to a subset of the Natural Images dataset. The selected dataset includes three visually distinct classes: airplane, car, and motorbike. All models were implemented using Scikit-learn and evaluated on the same preprocessed data using accuracy, precision, recall, F1-score, and confusion matrices.

By analyzing each model's performance, the report aims to highlight their respective strengths, limitations, and suitability for image classification tasks. This comparison provides insight into how model complexity and design influence classification accuracy and generalization.

Table of Contents

Table of figures

List of tables

Dataset Description

The dataset used in this study is the "Natural Images" dataset, available on Kaggle[1]. It consists of over 6,000 colored images classified into eight different categories such as airplane, car, motorbike, flower, fruit, and more. For this project, we focused exclusively on three classes: airplane (727 images), car (968 images), and motorbike (788 images). These categories were chosen based on their visual distinguishability and balanced representation in the dataset.

The dataset contains images of varying sizes and resolutions. Each image represents a real-world object photographed under different lighting, angles, and backgrounds. This introduces natural variability, making it a practical testbed for evaluating the generalization capabilities of classification algorithms. The images were originally labeled and categorized into respective folders, simplifying the process of extracting labels during preprocessing. The selection of a multiclass, RGB image dataset reflects the complexity typically encountered in real-world image classification problems.

Preprocessing

Effective preprocessing is essential for achieving good model performance in image classification. The first step involved resizing all images to a uniform size of 64x64 pixels. This dimensional standardization ensures consistency in feature vectors while significantly reducing computational complexity. Instead of converting the images to grayscale, we retained the RGB color format to preserve color-based features, resulting in $64 \times 64 \times 3 = 12{,}288$ features per image.

The images were then flattened from 3D matrices into 1D arrays, making them suitable for input into scikit-learn models, which expect 2D feature matrices. After flattening, pixel values were normalized by dividing by 255.0 to scale all values to the [0,1] range. For the Multi-Layer Perceptron, feature standardization was applied using StandardScaler to achieve zero mean and unit variance, which enhances convergence and stability during training.

Models Implemnation

1. Naive Bayes Classifier

The Naive Bayes classifier is based on Bayes' Theorem and assumes that the features are conditionally independent given the class label. In our implementation, we used the Gaussian Naive Bayes model from scikit-learn's sklearn.naive_bayes module. This version assumes that the input features follow a normal (Gaussian) distribution. The classifier was trained on the normalized pixel intensity values of the flattened images. Despite its simplicity, the model is computationally efficient and surprisingly effective as a baseline. However, its independence assumption is often violated in image data, which can limit its performance.

Implementation Parameters:

- GaussianNB(): We used the default constructor without modification. This model doesn't require hyperparameter tuning, making it a good first step in any classification task.

2. Decision Tree Classifier

Decision Trees work by recursively splitting the input space based on feature values to maximize information gain, typically measured using entropy or Gini impurity. For our study, we employed scikit-learn's DecisionTreeClassifierwith the 'entropy' criterion. This model is particularly interpretable, as the decision-making process can be visualized as a tree structure. In our implementation, we limited the maximum depth to 10 to prevent overfitting and to keep the tree human-readable. The trained model was visualized using plot_tree(), which helped in understanding the pixel intensity thresholds that guide classification decisions.

Implementation Parameters:

- criterion='entropy': Specifies that the model should use entropy to measure the quality of splits instead of the default Gini impurity.
- random_state=42: Sets a seed for the random number generator to ensure reproducibility of results.

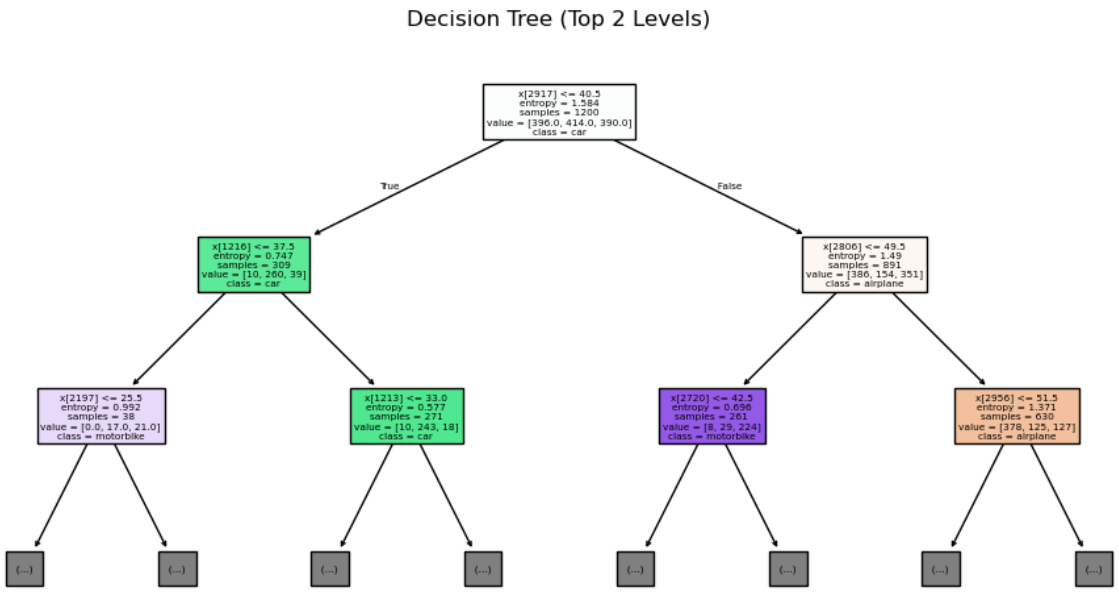


Figure 1: Decision Tree (Top 2 levels)

3. Multi-Layer Perceptron (MLPClassifier)

The Multi-Layer Perceptron is a class of feedforward artificial neural networks composed of input, hidden, and output layers. Each neuron in one layer is connected to every neuron in the subsequent layer. MLPs use non-linear activation functions, allowing them to learn complex decision boundaries. We used scikit-learn's MLPClassifier, configured with two hidden layers of 256 and 128 neurons, respectively. The activation function used was ReLU (Rectified Linear Unit), and the optimizer was Adam, which is well-suited for large-scale data. Standardized feature inputs improved convergence during training. The MLP model required significantly more training time but offered superior accuracy and robustness.

Implementation Parameters:

- hidden_layer_sizes=(256, 128): Defines two hidden layers, the first with 256 neurons and the second with 128 neurons. This structure allows the model to learn increasingly abstract representations.
- activation='relu': Applies the ReLU activation function, which introduces non-linearity and helps prevent the vanishing gradient problem.

- solver='adam': Uses the Adam optimization algorithm, which combines the advantages of both AdaGrad and RMSProp.
- max_iter=500: Sets the maximum number of iterations (epochs) for training, giving the model ample time to converge.
- random_state=42: Ensures that results are reproducible by initializing the random number generator with a fixed seed.

Evaluation Metrics

The models were evaluated using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

We used an 80/20 train-test split with stratification to ensure class balance across both subsets.

Results

Accuracy Summary

Table 1: Accuracy results for each model

MODEL	ACCURACY
NAIVE BAYES	73%
DECISION TREE	79%
MLPCLASSIFIER	94%

Confusion Matrices

Each confusion matrix visualizes how actual labels compare with predicted labels for the three models.

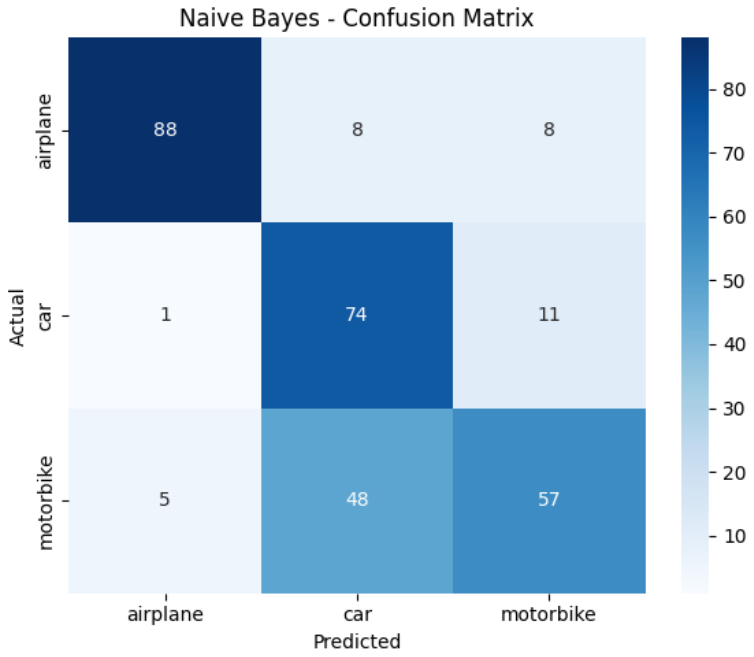


Figure 2: Naive Bayes Confusion Matrix

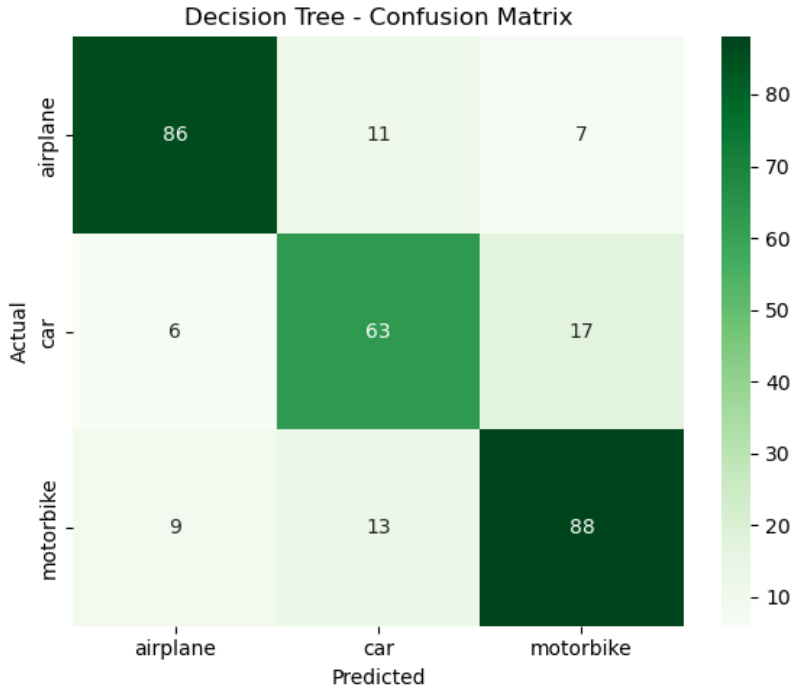


Figure 3: Decision Tree Confusion Matrix

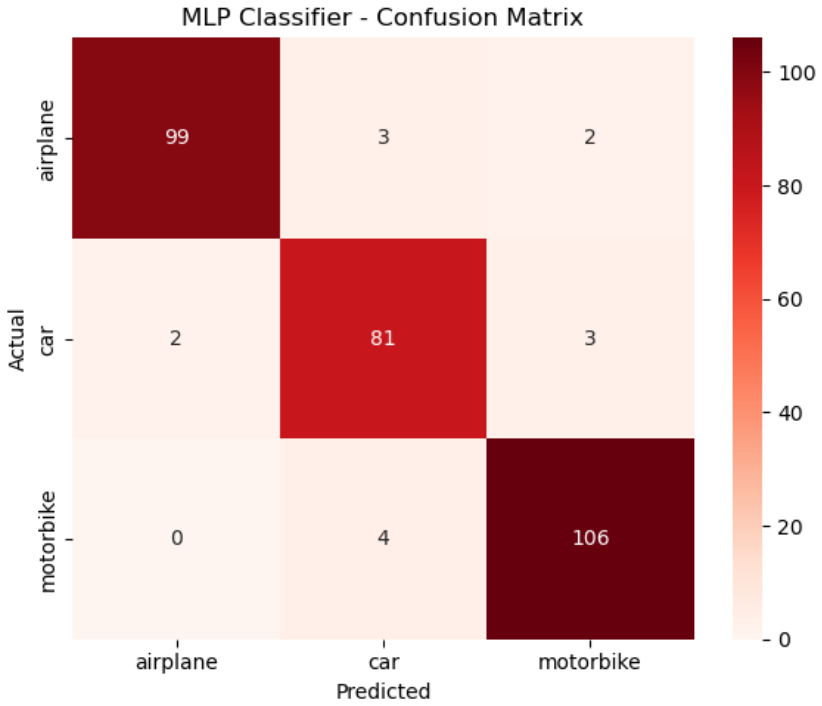


Figure 4: MLPClassifier Confusion Matrix

Classification Reports

```
=== Naive Bayes ===
Accuracy: 0.7766666666666666
Report:
              precision    recall  f1-score   support

           0       0.94      0.90      0.92       104
           1       0.62      0.88      0.73        86
           2       0.81      0.57      0.67       110

    accuracy                           0.78       300
   macro avg       0.79      0.79      0.77       300
weighted avg       0.80      0.78      0.77       300
```

Figure 5: Naive Bayes Classification Report

```
=== Decision Tree ===
Accuracy: 0.8366666666666667
Report:
              precision    recall  f1-score   support

           0       0.88      0.89      0.89       104
           1       0.83      0.78      0.80        86
           2       0.81      0.83      0.82       110

    accuracy                           0.84       300
   macro avg       0.84      0.83      0.83       300
weighted avg       0.84      0.84      0.84       300
```

Figure 6: Decision Tree Classification Report

```
=== MLP Classifier ===
Accuracy: 0.95
Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       104
           1       0.91      0.95      0.93        86
           2       0.95      0.94      0.94       110

    accuracy                           0.95       300
   macro avg       0.95      0.95      0.95       300
weighted avg       0.95      0.95      0.95       300
```

Figure 7: MLPClassifier Classification Report

Analysis and Comparison

The three models evaluated Naive Bayes, Decision Tree, and MLPClassifier show clear differences in their classification accuracy and confusion matrix results, reflecting their underlying learning mechanisms and suitability for high-dimensional image data.

Accuracy Overview

As shown in **Table 1**, the Naive Bayes classifier achieved an accuracy of **73%**, the Decision Tree classifier improved upon that with **79%**, while the MLPClassifier delivered the highest accuracy of **94%**. This progression highlights how model complexity and capacity affect classification performance.

Confusion Matrix Analysis

The **confusion matrix for Naïve Bayes** as shown in Fig.1 shows that Naive Bayes struggled particularly with distinguishing **motorbikes from cars**, misclassifying 48 motorbike

images as cars. This high misclassification rate stems from the model's assumption of feature independence, which does not hold in image data where spatial relationships and texture features matter. Nonetheless, it showed reasonable performance in identifying airplanes.

The **confusion matrix for Decision Tree** as shown in Fig.2 shows that The Decision Tree model demonstrated improved classification across all three categories. However, it still had noticeable misclassifications, such as:

- 11 airplane images predicted as cars,
- 17 car images predicted as motorbikes.

The tree's structure allowed it to form rules from pixel intensities, but without capturing complex spatial hierarchies, its performance remained moderate.

The **confusion matrix for MLPClassifier** as shown in Fig.3 shows that The MLPClassifier showed strong classification performance across all classes:

- It correctly identified **99 out of 104 airplanes**,
- **81 out of 86 cars**, and
- **106 out of 110 motorbikes**.

Very few misclassifications occurred, demonstrating the MLP's ability to learn abstract features from the input data through multiple hidden layers and nonlinear activation functions. Its confusion matrix was the most diagonally dominant, indicating high precision and recall across all classes.

Overall Discussion

From these results, it is clear that:

- **Naive Bayes** is best suited as a baseline for quick experiments or very simple tasks. Its low computational cost is an advantage, but its performance drops when features are interdependent.
- **Decision Trees** strike a balance between interpretability and performance. While they capture non-linear decision boundaries better than Naive Bayes, they are prone to overfitting unless pruned or depth-limited.

- **MLPClassifier** outperforms both models due to its deep learning architecture. Although it requires more preprocessing (e.g., feature scaling) and longer training time, the significant gain in accuracy makes it ideal for real-world image classification tasks.

Conclusion

In this report, we compared Naive Bayes, Decision Tree, and MLPClassifier for the task of image classification using a subset of the Natural Images dataset. After preprocessing and training on three categories — airplane, car, and motorbike — we evaluated each model based on accuracy, confusion matrices, and classification reports.

Naive Bayes offered fast, baseline performance but struggled with complex patterns. Decision Tree improved accuracy and provided interpretability but was prone to overfitting. The MLPClassifier achieved the highest accuracy (94%) and strongest class-wise performance, demonstrating its ability to model non-linear relationships effectively.

Overall, while simpler models are useful for fast and interpretable classification, the MLPClassifier proved to be the most effective. Future improvements could include using convolutional neural networks (CNNs) for deeper feature extraction and better performance.

References

[1] Dataset: https://www.kaggle.com/datasets/prasunroy/natural-images

[2] Scikit-learn documentation: https://scikit-learn.org/stable/supervised_learning.html