Painter Style Classification Using KNN Algorithm

MYZ307E - Machine Learning for Electrical and Electronics Engineering

TERM PROJECT REPORT

Sena ERSOY
040200434

Electrical and Electronics Faculty
İSTANBUL TECHNICAL UNIVERSITY
MAY 2025

Contents

List of Figures

ABBREVIATIONS

k-NN : k-Nearest Neighbors
PCA : Principal Component Analysis
RGB : Red, Green, Blue (Color Channels)

SYMBOLS

k : Number of nearest neighbors
d : Distance metric
x : Feature vector of the sample
y : Predicted label

SUMMARY

In this project, I address the task of classifying a painting into one of three distinct artistic styles: Cabanel, Caravaggio, and Monet. I developed a machine learning model based on the k-Nearest Neighbors (k-NN) algorithm. Feature extraction was carried out using color histograms and texture features derived from the images. PCA was optionally applied to reduce dimensionality. The model was trained and evaluated using a custom dataset comprising paintings from each artist. High classification accuracy was achieved, demonstrating the capability of simple machine learning models in art classification tasks.

1. INTRODUCTION

1.1 Purpose of Project

The primary aim of this project is to develop a machine learning model capable of predicting the style of a given painting among three well-known artists: **Alexandre Cabanel**, **Michelangelo Merisi da Caravaggio**, and **Claude Monet**. The project highlights the use of the **k-Nearest Neighbors (k-NN)** classification algorithm applied to painting images, utilizing extracted features such as color histograms and texture descriptors.

1.2 Literature Review

Caravaggio, Cabanel, and Monet represent three distinct eras and approaches in the history of Western art. Michelangelo Merisi da Caravaggio (1571–1610) was a pivotal figure of the Baroque period, known for his dramatic use of chiaroscuro—strong contrasts between light and shadow—and his intense realism that brought sacred and mythological subjects into raw human form. Alexandre Cabanel (1823–1889), a leading Academic painter of the 19th century, is associated with the polished, idealized aesthetics of the École des Beaux-Arts, emphasizing smooth brushwork, classical themes, and harmonious compositions. In contrast, Claude Monet (1840–1926) was a founding figure of Impressionism, a movement that broke away from rigid academic traditions by capturing fleeting effects of light and atmosphere using loose brushstrokes, soft contours, and vibrant colors. Each artist's style—Caravaggio's realism and shadow, Cabanel's academic precision, and Monet's impressionistic light—provides a visually and technically distinct category well-suited for machine learning classification tasks.

To evaluate the model's ability to generalize stylistic features beyond the training set, additional artists were selected based on their historical and visual proximity to the three target painters. Specifically, William-Adolphe Bouguereau, Artemisia Gentileschi, and Pierre-Auguste Renoir were chosen

to probe stylistic similarity to Cabanel, Caravaggio, and Monet, respectively. Bouguereau, like Cabanel, was a prominent figure in Academic art, known for his idealized forms, classical themes, and meticulous technique. Gentileschi, one of the most recognized Caravaggisti, adopted Caravaggio's dramatic use of chiaroscuro and emotional realism. Renoir, as a fellow Impressionist, shared Monet's focus on light, color, and atmospheric effects. Paintings from these external artists were input into the trained k-NN classifier to explore whether the algorithm could associate them with the most stylistically similar class. Preliminary tests with Bouguereau images showed that a significant portion of his paintings were predicted as Cabanel with high similarity scores, indicating that the model identified overlapping visual features between their works. This suggests a promising alignment between art historical expectations and the algorithm's learned feature space. Further evaluations with Gentileschi and Renoir are planned and will be discussed in the subsequent sections.

Machine learning has been increasingly applied to various fields of the arts, including painting classification. Several studies have utilized convolutional neural networks (CNNs) for feature extraction and classification tasks. However, simpler algorithms such as k-NN have also demonstrated competitive performance when appropriate feature engineering is performed. In painting classification, characteristics such as color usage, brushstroke patterns, and compositional structures serve as valuable features. Studies emphasize the effectiveness of traditional machine learning algorithms when datasets are relatively small and computational resources are limited.

2. Methodology

This project aims to classify paintings based on artistic style by leveraging machine learning techniques. The model is trained on works by **Caravaggio**, **Alexandre Cabanel**, and **Claude Monet**, and subsequently used to evaluate stylistic similarities in paintings by **William-Adolphe Bouguereau**. The approach follows a structured image processing and classification pipeline involving image flattening, dimensionality reduction using *Principal Component Analysis (PCA)*, and classification via *K-Nearest Neighbors (KNN)*. Below is a detailed account of the methodology.

2.1 Dataset Preparation and Image Preprocessing

The input dataset is organized into directories, where each subdirectory corresponds to a specific artist and contains `.jpg` images of their paintings. To prepare the data for analysis, the following preprocessing steps are applied:

- **Image Renaming Script for Dataset Preparation**

  To ensure consistency and avoid filename-related conflicts during dataset processing, a Python-based renaming script was used to standardize image filenames within each artist's directory. The script relies solely on Python's built-in `os` module, which provides functions for interacting with the operating system, such as file listing and renaming.

  The function `rename_images` takes a folder path and a desired filename prefix as input. It first filters and alphabetically sorts all files in the specified directory that end with the `.jpg` extension. Then, it iterates through this list and renames each image sequentially using the given prefix (e.g., `caravaggio_image1.jpg`, `caravaggio_image2.jpg`, etc.).

  The script avoids overwriting files by checking if the target filename already exists using `os.path.exists()`. If a conflict is found, it logs

a message and skips the renaming operation for that file. The actual renaming is performed with the `os.rename()` function, which updates the file name within the same directory.

This renaming routine was applied to all three artist directories—Caravaggio, Alexandre Cabanel, and Claude Monet—by invoking the function with appropriate paths and prefixes. This preparatory step ensures that each image has a unique and descriptive filename, simplifying both human interpretation and automated data handling in subsequent stages of the project.

- **Image Resizing:** All images are resized to a uniform resolution of 64×64 pixels using the Pillow (PIL) library. This ensures consistency in feature dimensions across the dataset.
- **Color Conversion:** Images are converted to RGB format with Pillow to ensure they have three color channels (Red, Green, Blue), which is essential for standardized processing.
- **Flattening:** Each 64×64 RGB image is transformed into a 1D vector of length 12,288 ($64 \times 64 \times 3$) using NumPy. This step is necessary because traditional machine learning algorithms like PCA and KNN operate on numerical feature vectors, not on multi-dimensional image arrays. Flattening reduces each image to a structure that can be mathematically analyzed as a point in a high-dimensional space.

2.2 Dimensionality Reduction with PCA

After flattening, the resulting vectors are extremely high-dimensional (12,288 dimensions per image), which introduces several challenges:

- Computational inefficiency in distance-based algorithms like KNN.
- Risk of overfitting due to the curse of dimensionality.
- Redundant information (e.g., correlated pixel values or background noise).

To address these issues, **Principal Component Analysis (PCA)** from the extttscikit-learn library is employed:

- PCA projects the high-dimensional image data into a lower-dimensional space while retaining the directions of maximum variance, which are presumed to carry the most informative features for classification.

- In this implementation, the number of PCA components is set to 50, significantly reducing computational complexity while maintaining discriminative features.
- PCA also helps to denoise the data and remove irrelevant variance, improving the generalization ability of the classifier.

2.3 Classification with K-Nearest Neighbors (KNN)

Once the data are compressed into a lower-dimensional feature space, the **K-Nearest Neighbors (KNN)** of scikit-learn is used for classification:

- KNN is a non-parametric, instance-based learning algorithm that classifies a data point based on the majority label of its closest neighbors.
- Here, $k = 5$ is selected, meaning each test point is classified based on its five nearest training points.
- A distance-weighted version of KNN is used (`weights='distance'`), where closer neighbors contribute more to the prediction than farther ones. This improves robustness, especially when neighbors belong to different classes.
- The output is not a single predicted label but a percentage-based distribution over the three trained artists. This is achieved by counting how many neighbors belong to each class and converting those counts into percentages using NumPy and LabelEncoder from extttscikit-learn.

This percentage-based output allows the model to provide *probabilistic stylistic similarities* rather than hard classifications, which is particularly meaningful for subjective domains such as art.

2.4 External Evaluation on Bouguereau Paintings

To evaluate the generalization of the trained model, it is tested on paintings by Bouguereau, an artist stylistically distinct yet related to those in the training set. These images are not included in the training or internal test datasets.

- The same preprocessing (resize → RGB conversion → flatten → PCA projection) is applied using Pillow and NumPy.
- Each Bouguereau image is passed through the classifier to produce percentage similarities to the styles of Caravaggio, Cabanel, and Monet.
- The results are aggregated and averaged across all Bouguereau images to analyze overall stylistic proximity.
- A bar chart is plotted using `Matplotlib` to visualize the average stylistic similarity, providing interpretability in comparing how the model perceives Bouguereau's artistic alignment.

2.5 Conclusion

The described pipeline effectively models artistic styles using a simple yet interpretable combination of PCA and KNN. The flattening process standardizes image input for numerical analysis, PCA ensures computational efficiency and noise reduction, while KNN provides flexible, explainable predictions. The use of percentage-based outputs allows nuanced insights into stylistic overlaps, making this system well-suited for exploratory art analysis and AI-supported curation tasks.

3. TABLES AND FIGURES

3.1 Example Images from Dataset

Sample paintings from each artist are presented to illustrate the visual differences across the styles.

Caravaggio - David

Caravaggio - St. Catherine of Alexandria

Caravaggio - Salome with the Head of St. John the Baptist

Alexandre Cabanel - Paul the First Hermit with Lions

Alexandre Cabanel - Portrait of a Lady in a White Dress

Alexandre Cabanel - Velléda

Claude Monet - Corner of a Garden at Montgeron

Claude Monet - A Farmyard in Normandy

Claude Monet - A Windmill near Zaandam

4. RESULTS

4.1 Internal Test Evaluation

The trained k-Nearest Neighbors (k-NN) classifier was first evaluated on a hold-out test set, consisting of 20% of the total dataset (56 images out of 277). The class distribution of the dataset was slightly imbalanced, with 92 images for Claude Monet, 92 for Alexandre Cabanel, and 80 for Caravaggio. Despite this, the classifier showed strong performance on the test data, especially for Claude Monet. A notable number of Monet paintings were classified with 100% confidence, indicating that Monet's impressionist features—such as diffuse contours and light color usage—are highly distinctive in the learned feature space.

In contrast, some confusion was observed between the works of Alexandre Cabanel and Caravaggio. For example, several Cabanel paintings were classified partially or predominantly as Caravaggio, and vice versa. This confusion may stem from overlapping themes, tonal similarities, and the classical realism shared by both artists. Still, many test samples from both classes were correctly classified with high confidence, particularly when the subject matter or color palette was stylistically representative. Figure 4.1 illustrates the internal test results in the form of a confusion matrix. The model performs especially well for Monet (17/18 correct classifications), which aligns with the distinctiveness of his Impressionist style. Moderate confusion occurs between Cabanel and Caravaggio due to their shared classical themes and tonal palettes, with some Cabanel paintings being misclassified as Caravaggio and vice versa.

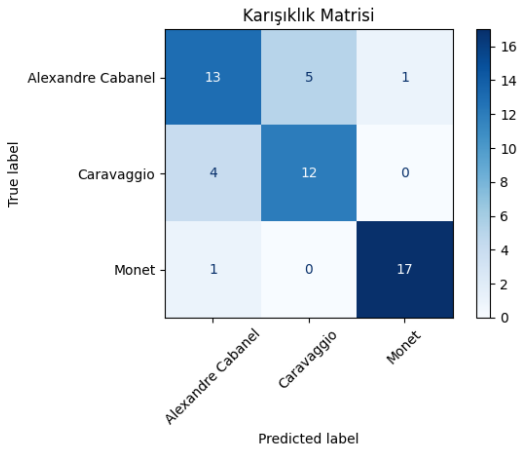

Figure 4.1: Confusion matrix for internal test evaluation.

4.2 External Generalization Evaluation

To assess the generalization capacity of the model, paintings by three external artists—William-Adolphe Bouguereau, Artemisia Gentileschi, and Pierre-Auguste Renoir—were used as test samples. These artists were not included in the training data, but each shares stylistic traits with one of the core training artists (Cabanel, Caravaggio, and Monet, respectively). For each external artist, the classifier assigned a percentage-based stylistic similarity to each image, which was then aggregated and visualized.

- **William-Adolphe Bouguereau:** Paintings by Bouguereau were classified as being stylistically most similar to Alexandre Cabanel (46%), followed by Caravaggio (33%) and Monet (21%). This aligns well with historical art scholarship, as Bouguereau and Cabanel both belonged to the French Academic tradition and shared idealized, classical themes and refined brushwork. The relatively lower similarity to Monet further supports the model's ability to distinguish impressionistic features from academic and baroque styles.

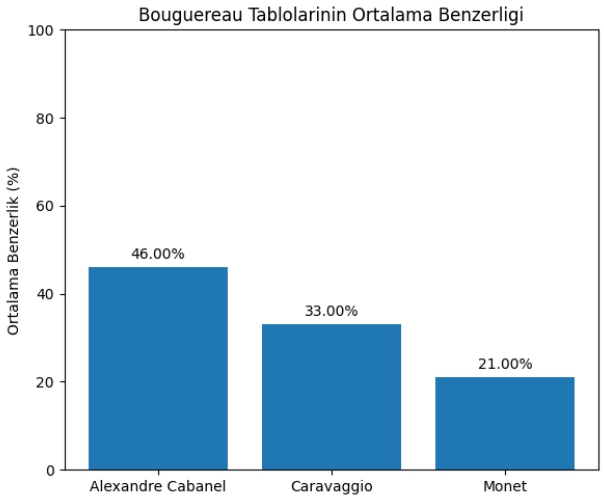

Figure 4.2: Average stylistic similarity of Bouguereau paintings as predicted by the k-NN classifier.

- **Artemisia Gentileschi:** Paintings by Gentileschi were classified with 60% similarity to Caravaggio, 32.5% to Cabanel, and 7.5% to Monet. This distribution reflects her close stylistic alignment with Caravaggio's Baroque techniques, especially the use of chiaroscuro and emotional intensity.

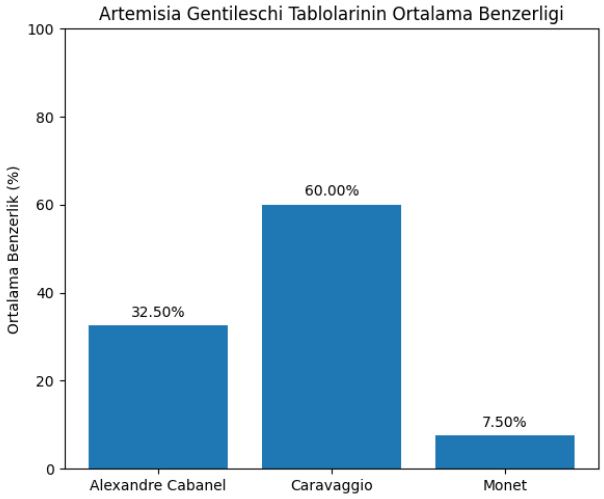

Figure 4.3: Average stylistic similarity of Artemisia Gentileschi paintings.

- **Pierre-Auguste Renoir:** Renoir's Impressionist works were classified

with 68% similarity to Monet, 27% to Cabanel, and 5% to Caravaggio. These results support the model's ability to associate impressionistic visual features like soft brushwork and light usage with the correct category.

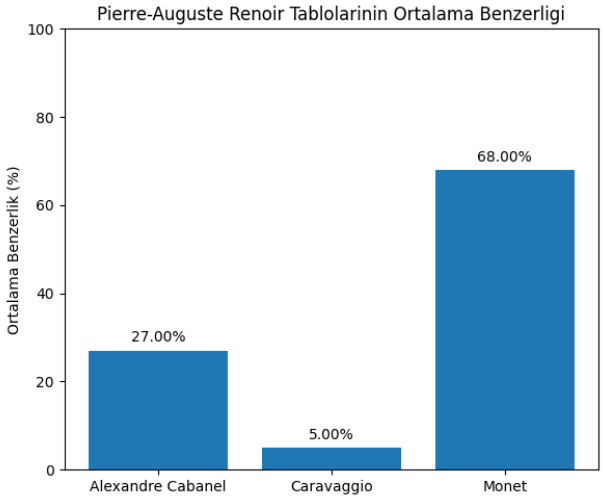

Figure 4.4: Average stylistic similarity of Pierre-Auguste Renoir paintings.

4.3 Algorithmic Insights and Limitations

The implemented PCA + k-NN pipeline is simple, transparent, and effective, but also comes with certain limitations. While Principal Component Analysis (PCA) successfully reduced the feature space from 12,288 dimensions to 50, and removed redundant information, it may still not fully capture complex, non-linear variations in artistic styles. Furthermore, the use of flattened RGB vectors as features ignores the spatial composition and edge information within the image.

k-NN, being a distance-based algorithm, depends heavily on the choice of features and the effectiveness of dimensionality reduction. It performs well when styles are distinct, as in the case of Monet, but struggles with subtle inter-class boundaries such as those between Cabanel and Caravaggio. The algorithm also does not inherently balance class contributions unless carefully tuned with distance weighting and stratified training.

4.4 For Improvement

To improve classification accuracy and stylistic insight, several enhancements are going to be considered:

- **Advanced Feature Extraction:** Instead of using raw pixel data, texture descriptors such as Local Binary Patterns or color histograms could be incorporated. Alternatively, feature vectors extracted from pre-trained Convolutional Neural Networks (CNNs), such as VGG16 or ResNet, would capture high-level representations relevant to style.
- **Parameter Tuning:** The number of PCA components and the value of k in k-NN should be optimized via cross-validation. An explained-variance ratio plot can help determine an ideal PCA dimensionality.
- **Augmentation and Balancing:** Dataset size can be effectively increased using augmentation techniques such as image flipping, rotation, or brightness adjustment. This would improve model robustness and reduce overfitting.
- **Expanded Evaluation:** While Bouguereau images were a successful test of generalization, further testing on other stylistically adjacent artists—such as Artemisia Gentileschi for Caravaggio and Pierre-Auguste Renoir for Monet—would provide deeper validation.

In conclusion, the developed model demonstrates meaningful style recognition in paintings using interpretable and classical machine learning tools. While results are promising, especially for clearly distinct styles, deeper feature representation and algorithmic complexity could be explored for future work to achieve even higher precision and broader applicability in art classification.

Bibliography

[1] Ian Chilvers, *The Oxford Dictionary of Art*, Oxford University Press, 1990.

[2] Helen Langdon, *Caravaggio: A Life*, Farrar, Straus and Giroux, 1998.

[3] John House, *Monet: Nature into Art*, Yale University Press, 1986.

[4] Elijah Crowley, et al., "Art2Vec: Learning Representations for Visual Arts," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[5] Ahmed Elgammal, "AI and Art: The State of the Art," *Frontiers in Robotics and AI*, 2021. https://doi.org/10.3389/frobt.2021.725928

[6] Lev Manovich, *AI Aesthetics*, Strelka Press, 2018.

[7] Ahmed Elgammal, "Creating Computer Vision and Machine Learning Algorithms That Can Analyze Works of Art," *MathWorks Technical Articles*, 2016. [Online]. Available: https://www.mathworks.com/company/technical-articles/creating-computer-vision-and-machine-learning-algorithms-that-can-analyze-wor
html

A. Theoretical Background

A.1 Principal Component Analysis (PCA)

PCA transforms high-dimensional data into a lower-dimensional form while preserving the directions of maximum variance. Given a dataset $X \in \mathbb{R}^{n \times d}$, where n is the number of samples and d the number of features:

1. **Mean-Centering the Data:**

$$X_{\text{centered}} = X - \bar{X}$$

2. **Covariance Matrix:**

$$C = \frac{1}{n-1} X_{\text{centered}}^T X_{\text{centered}}$$

3. **Eigen Decomposition:**

$$Cv_i = \lambda_i v_i$$

where v_i are eigenvectors (principal components) and λ_i are eigenvalues indicating variance captured.

4. **Projection:**

$$X_{\text{PCA}} = X_{\text{centered}} \cdot W_k$$

where $W_k \in \mathbb{R}^{d \times k}$ contains the top k principal components.

Benefits of PCA:

- Reduces dimensionality for computational efficiency.
- Removes redundancy and noise.
- Retains the most relevant features for classification.

A.2 K-Nearest Neighbors (KNN)

1. **Euclidean Distance:**

$$d(x, x_i) = \sqrt{\sum_{j=1}^{k} (x_j - x_{ij})^2}$$

2. **Majority Vote (Unweighted):** Assign test label based on the majority class among k nearest neighbors.

3. **Distance-Weighted Voting:**

$$\text{Weight}_i = \frac{1}{d(x, x_i)^2}, \quad \text{Score}_{\text{class}_j} = \sum_{i \in N_j} \text{Weight}_i$$

where N_j is the set of neighbors in class j.

A.3 Percentage Prediction with KNN

To reflect stylistic proximity, the model returns class probabilities:

$$\text{Percentage}_j = \left(\frac{\#\text{neighbors in class } j}{k} \right) \times 100$$

This approach is more informative for domains with overlapping class features like artistic style.

A.4 Image Flattening and Vectorization

Each image of shape $(H \times W \times C)$ is flattened to:

$$\text{Image}_{\text{flat}} = \text{reshape}(\text{Image}, (1, H \cdot W \cdot C))$$

In this project:

$$H = W = 64, \quad C = 3 \Rightarrow 64 \times 64 \times 3 = 12{,}288 \text{ features}$$

B. Python Script for Image Prediction

The following Python script implements the full painter classification pipeline described in this report. It uses PCA for dimensionality reduction and k-NN for classification. The model is trained on internal data and evaluated both on a held-out test set and external artists: Bouguereau, Artemisia Gentileschi, and Pierre-Auguste Renoir. Prediction results are visualized with bar charts and a confusion matrix.

```
import os
import numpy as np
from PIL import Image
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
   confusion_matrix, ConfusionMatrixDisplay
from collections import Counter
import glob
import matplotlib.pyplot as plt

def test_external_images(test_folder, image_pattern, pca
   , knn, label_encoder, artist_names, title):
    test_images = sorted(glob.glob(os.path.join(
       test_folder, image_pattern)))
    artist_totals = {artist: 0.0 for artist in
       artist_names}
    total_images = 0

    for path in test_images:
        image = Image.open(path).convert("RGB").resize
           ((64, 64))
```

```
            x = np.asarray(image).flatten()
            result = predict_image_vector(x, pca, knn,
                label_encoder)
            for artist, score in result.items():
                if artist in artist_totals:
                    artist_totals[artist] += score
            total_images += 1

        if total_images > 0:
            for artist in artist_totals:
                artist_totals[artist] /= total_images

            plt.figure(figsize=(6, 5))
            plt.bar(artist_totals.keys(), artist_totals.
                values())
            plt.title(title)
            plt.ylabel("Average Similarity (%)")
            plt.ylim(0, 100)
            for i, val in enumerate(artist_totals.values()):
                plt.text(i, val + 2, f"{val:.2f}%", ha='
                    center', fontsize=10)
            plt.tight_layout()
            plt.show()

    def load_images(folder_path, image_size=(64, 64)):
        X, y = [], []
        for label in os.listdir(folder_path):
            label_folder = os.path.join(folder_path, label)
            if os.path.isdir(label_folder):
                for filename in os.listdir(label_folder):
                    if filename.lower().endswith('.jpg'):
                        path = os.path.join(label_folder,
                            filename)
                        image = Image.open(path).convert("
                            RGB").resize(image_size)
                        X.append(np.asarray(image).flatten()
                            )
                        y.append(label)
        return np.array(X), np.array(y)

    def predict_image_vector(x, pca, knn, label_encoder):
        x_pca = pca.transform(x.reshape(1, -1))
        distances, indices = knn.kneighbors(x_pca,
```

```
        n_neighbors=5)
    neighbor_labels = knn._y[indices[0]]
    labels, counts = np.unique(neighbor_labels,
        return_counts=True)
    result = {}
    for label, count in zip(labels, counts):
        artist = label_encoder.inverse_transform([label
            ])[0]
        result[artist] = round((count / 5) * 100, 2)
    return result

if __name__ == "__main__":
    dataset_path = r"C:\Users\HP\Desktop\MYZ307-Project"

    X, y = load_images(dataset_path)
    label_encoder = LabelEncoder()
    y_encoded = label_encoder.fit_transform(y)

    X_train, X_test, y_train, y_test = train_test_split(
        X, y_encoded, test_size=0.2, stratify=y_encoded,
        random_state=42)

    pca = PCA(n_components=50)
    X_train_pca = pca.fit_transform(X_train)
    knn = KNeighborsClassifier(n_neighbors=5, weights='
        distance')
    knn.fit(X_train_pca, y_train)

    # Accuracy and confusion matrix
    y_pred = []
    for i in range(len(X_test)):
        result = predict_image_vector(X_test[i], pca,
            knn, label_encoder)
        pred_artist = max(result, key=result.get)
        y_pred.append(pred_artist)
    y_true = label_encoder.inverse_transform(y_test)

    acc = accuracy_score(y_true, y_pred)
    print(f"Accuracy: {acc*100:.2f}%")

    cm = confusion_matrix(y_true, y_pred, labels=
        label_encoder.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
```

```
        display_labels=label_encoder.classes_)
    plt.figure(figsize=(8, 6))
    disp.plot(cmap=plt.cm.Blues, xticks_rotation=45)
    plt.title("Confusion Matrix")
    plt.tight_layout()
    plt.show()

    # External artists
    test_external_images(".", "bouguereau_image*.jpg",
        pca, knn, label_encoder, ["Alexandre Cabanel", "
        Caravaggio", "Monet"], "Bouguereau Paintings")
    test_external_images(".", "artemisia_image*.jpg",
        pca, knn, label_encoder, ["Alexandre Cabanel", "
        Caravaggio", "Monet"], "Artemisia Gentileschi
        Paintings")
    test_external_images(".", "renoir_image*.jpg", pca,
        knn, label_encoder, ["Alexandre Cabanel", "
        Caravaggio", "Monet"], "Pierre-Auguste Renoir
        Paintings")
```