Machine Learning Project

Painting-to-Artist Image Classification

Student: Barak Finkel 206491714

Abstract:

The "Painting-To-Artist Image Classification Project"{1} investigates attempts for classification of paintings to their respective artists using machine learning techniques. Through preprocessing and model tuning, the project aims to uncover patterns and nuances in artwork styles. Results showcase varying scores across different models, reflecting the complexity of the classification task.

Introduction:

Art study, particularly in the realm of paintings, has long been a subject of fascination. Understanding the unique styles and techniques employed by different artists not only enriches our appreciation of art but also presents intriguing challenges for computational analysis.

Driven by a personal passion for art and a curiosity about its underlying patterns, this project seeks to attempt developing an image classification system capable of identifying the artists behind various paintings. Leveraging machine learning techniques and image processing algorithms, I aimed to create a model that can discern subtle artistic nuances and attributes, ultimately assigning each artwork to its correct creator.

By employing a systematic approach that involves data preprocessing, feature extraction, and model tuning, I was informed of the complexities of trying to encapsulate artistic expression within numerical data. This project is the first personal step in uncovering efficient ways to help machine learning models connect the dots between artistic creativities and unique styles and the person behind them.

Dataset:

The dataset in this project is a subset of images taken from Kaggle's "Best Artworks of All Time"{2}

The dataset utilized in this project comprises a collection of paintings sourced from various renowned artists spanning different periods and styles. The dataset encompasses artworks attributed to 10 prominent Artists: Andy Warhol, Frida Kahlo, Henri Matisse, Joan Miro, Leonardo da Vinci, Marc Chagall, Pablo Picasso, Rembrandt, Salvador Dali, and Vincent van Gogh.

All the artists above have rather different artistic styles. Moreover, most of them belonged to different artistic movements, making the dataset rich with different attributes to analyze.

Pre-Processing:

Pre-processing images plays a crucial role towards making the ML models of our choice have a better understanding of them. In the project, several preprocessing steps were made in attempts to enhance the readabilty and quality of the output data for the models to train on.

The steps taken comprise of the following pipeline:

1) **Sample**: 100 images were sampled from each artist in our dataset (the minimal ammount of images one of the artists has), summing up to 1000 images in total.
2) **Resize**: Each image was reduced to a 128x128x3 scale, preserving it's color. This was in order to preserve the artistic attributes of the images, such as brush strokes, contours and overall composition of each artpiece as well as decrease dimensionality as part of the goal to ease the process of data fitting.
3) **Normalization**: Each image's pixel values were normalized to bring them to a scale of [0,1], using the min-max normalization over each pixel in the image:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

4) **Feature Extraction**: 5 different methods of feature extraction were attempted during the corresponding process:
   A) **Pixel Values** – The raw pixel values of the images were considered as features. This basic approach captures the intensity of each pixel and color distribution.
   B) **Histogram of Oriented Gradients (HOG)** – HOG computes the distribution of local gradient orientations in the image, providing information over directionality of blocks in the image. It works by dividing the image into small, overlapping cells, computing gradient information within each cell, and then aggregating gradient orientations into a histogram. This process captures information about the shape and structure of objects in the image.
   C) **Haralick Features** - Haralick features are computed from the gray-level co-occurrence matrix (GLCM), which quantifies the spatial relationships between groups of pixels with a certain intensity value and their neighbors. These features capture textural information by analyzing the distribution of gray-level pairs in the image.
   D) **Gabor Filters** – Gabor filters are sinusoidal gratings modulated by a Gaussian envelope. They are characterized by their frequency, orientation, and spatial extent. Gabor filters are applied at multiple scales and orientations to analyze texture information in the image. They are particularly effective at capturing localized texture patterns across different spatial frequencies and orientations.

E) **Sobel Edges** - The Sobel filter is a convolution-based edge detection technique. It computes the gradient magnitude of the image by convolving it with a pair of M x M convolution kernels, one for detecting horizontal changes and the other for vertical changes in intensity. The resulting gradient magnitude highlights edges and boundaries in the image.

The inspiration for the attempts over gabor and sobel filters came from Sreenivas B.'s video{3} in which he covers basic fundamentals in data pre-processing.

Although, after experimenting on the dataset given in this project, HOG and Haralick features made more promising results in terms of computation time and in efficiency over dimensionality reduction.

In addition, they cover the same grounds more or less in terms of calculating gradients and recognizing textural patterns in each image.

Thus, the chosen combination of features were Pixel Values, HOG and Harilick features. Below are examples of the outputs the three, rescaled back to provide visual presentation.

Henri Matisse, 1905 – "The Green Stripe"

5) **<u>Principal Component Analysis (PCA)</u>**: Principal Component Analysis (PCA) is a widely-used technique for dimensionality reduction in machine learning and data analysis. It involves transforming the original data into a new coordinate system where the first principal component captures the greatest variance, followed by subsequent components. This transformation is achieved by finding the eigenvectors and eigenvalues of the data covariance matrix.

For context, Eigenvectors are vectors within a vector space that maintain their direction under a linear transformation, while eigenvalues represent the scaling factor of the eigenvectors. In PCA, eigenvectors correspond to the directions of maximum variance, and eigenvalues indicate the magnitude of variance along those directions. Together, they are crucial for identifying the principal components that capture the most significant sources of variability in the dataset.

In the project, several methods of applying PCA were conducted:

A) **First Attempt**: PCA performed individually on each picture after resizing, with a fixed number of components applied. However, this approach proved to be too rigid, as using a fixed number of components often led to the loss of important information from the images.
B) **Second Attempt**: PCA applied over the entire dataset after concatenating every filtered version together. In this approach, PCA is performed on the concatenated feature vectors obtained from all filtered versions of the images combined. While this approach seemed comprehensive, this type of reduction both omitted important values and its results were inconsistent over different samplings.
C) **Chosen Approach**: PCA applied over each set of the dataset's images for an individual filter. This method involves applying PCA separately to the feature vectors obtained from each filtered version of the images, enabling a more focused analysis of the variance within each specific filter's feature space.

6) **Dataframe Concatenation:** Following the application of each filter, resulting in 2D lists where rows represent individual images and columns represent the filter's features, these arrays were concatenated into a unified dataframe. This dataframe served as input for the selected machine learning models.

7) **Train-Test Split**: After preprocessing, the data is split for training and testing according to a chosen fraction from the dataset. In this project, we've chosen a generic 80/20 split.

Machine Learning Models:

In this project, several machine learning models were employed, each with its hyperparameters fine-tuned to optimize performance. The models selected were K-Nearest Neighbors (KNN), Random Forest, AdaBoost, and Support Vector Machine (SVM). Each model was constructed utilizing Scikit-Learn's implementation.

Each model offers unique strengths and characteristics that make them suitable for testing in this project. Each model received sets of hyperparameters, and utilized a cross validation grid search in order to find the best vector of parameters over the train data. During this process, the training data was split into five folds, where four folds were used for training and one for validation iteratively. This allowed each model to explore various combinations of hyperparameters and identify the most effective set for maximizing performance on the validation data.

The following explains of choice for every model, and the hyperparameters chosen for tuning:

- **K-Nearest Neighbors (KNN)**:
  KNN is a simple yet effective algorithm that doesn't make strong assumptions about the underlying data distribution. It's been found to be generally suitable for multi-class classification tasks and can handle both linear and non-linear decision boundaries.
  The hyperparameters tuned for this experiment were the number of neighbors, the type of weights, and the type of metric for the calculation of the distance (when chosen as the weight).

- **Random Forest**:
  Random Forest is an ensemble learning method known for its robustness and ability to handle high-dimensional data. It's also known for being less prone to overfitting compared to individual decision trees and can provide insights into feature importance.
  The hyperparameters tuned for this experiment were the number of estimators, the maximum number of features to consider when splitting, the maximum depth of the trees, and the criterion for the quality of the split.

- **AdaBoost**:
  AdaBoost is a boosting algorithm that sequentially improves the performance of weak learners by focusing on misclassified samples. It's been known to be effective in handling imbalanced datasets and can adapt well to different data distributions.
  The hyperparameters tuned for this experiment were the number of estimators, the learning rate, and the algorithm used for boosting. The chosen type of estimator was a decision tree with depth = 1.

- **Support Vector Machine (SVM)**:
  SVM is a powerful classifier capable of constructing non-linear decision boundaries through the use of kernel functions. It's known to work well in high-dimensional spaces and is effective even when the number of features exceeds the number of samples.
  The hyperparameters tuned for this experiment were the regularization parameter (C), the kernel type, and the kernel coefficient for 'rbf', 'poly', and 'sigmoid' kernels.

Results and Analysis:

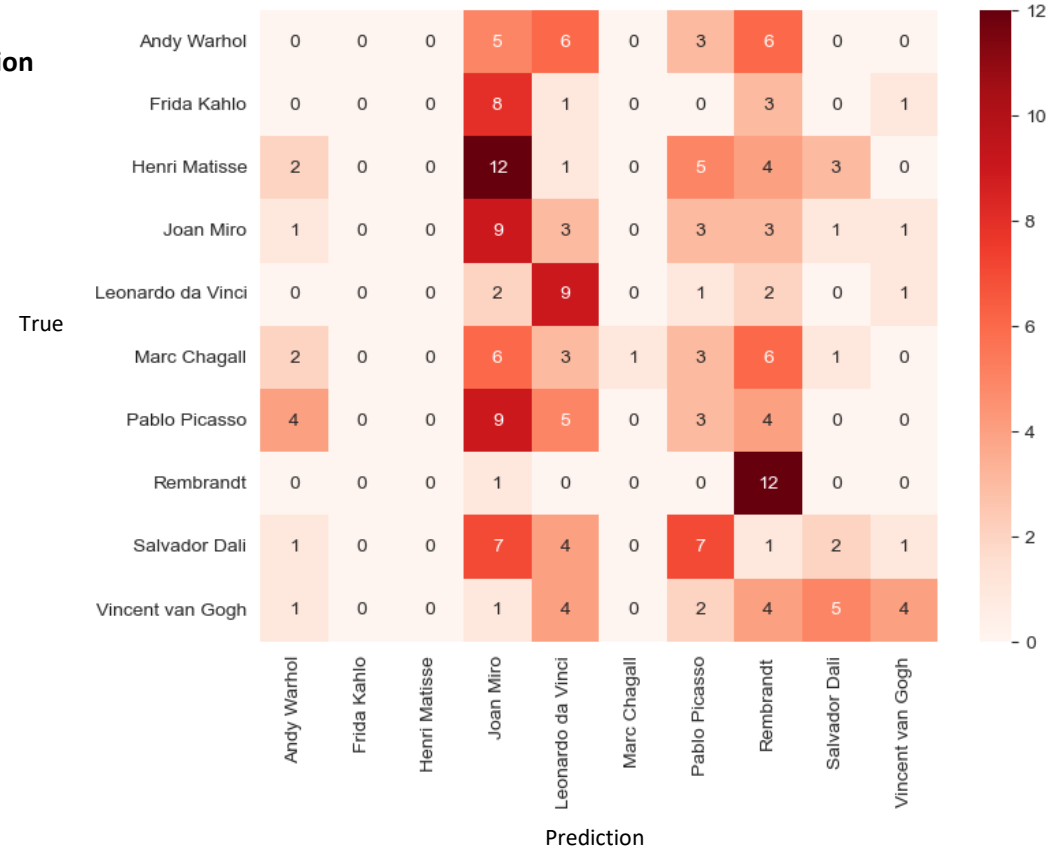
In this section, we present the results of the machine learning experiments conducted on the painting-to-artist image classification project. We begin by outlining the performance metrics of each model and discussing their respective accuracies. Subsequently, we delve into a detailed analysis of the classification outcomes, examining the strengths and limitations of each model in accurately predicting the artists from the input images. Finally, we explore the implications of these findings and provide insights into potential avenues for further research and improvement.

K-Nearest Neighbors (KNN):
Best Params: 'metric': 'Euclidean', 'n neighbors': 20, 'weights': 'distance'
Empirical Accuracy: 1.000, True Accuracy: 0.200

Classification Report:

Class	Precision	Recall	F1-Score	Support
Andy Warhol	0.00	0.00	0.00	20
Frida Kahlo	0.00	0.00	0.00	13
Henri Matisse	0.00	0.00	0.00	27
Joan Miro	0.15	0.43	0.22	21
Leonardo da Vinci	0.25	0.60	0.35	15
Marc Chagall	0.00	0.05	0.09	22
Pablo Picasso	0.11	0.12	0.12	25
Rembrandt	0.27	0.92	0.41	13
Salvador Dali	0.17	0.09	0.11	23
Vincent van Gogh	0.50	0.19	0.28	21
Accuracy			**0.20**	**200**

Confusion Matrix:



- The KNN model achieved an accuracy of 20%, indicating poor performance overall.
- The confusion matrix shows that most classes were misclassified, with low precision, recall, and F1-scores for each class.
- Notably, classes like "Andy Warhol" and "Frida Kahlo" exhibit particularly low precision and recall, suggesting significant challenges in accurately identifying these artists' paintings.

Random Forest:

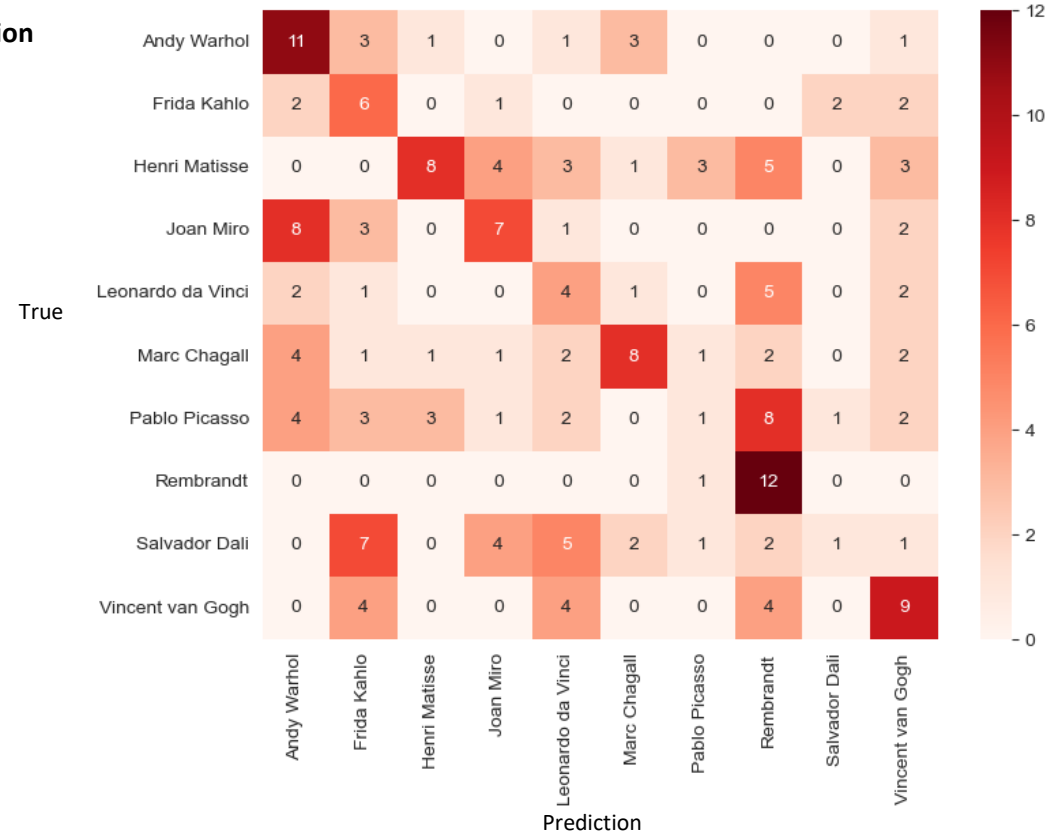
Best Params: {'criterion': 'entropy', 'max depth': 4, 'max features': 'sqrt', 'n estimators': 40}

Empirical Accuracy: 0.821, **True Accuracy**: 0.335

Classification Report:

Class	Precision	Recall	F1-Score	Support
Andy Warhol	0.35	0.55	0.43	20
Frida Kahlo	0.21	0.46	0.29	13
Henri Matisse	0.62	0.30	0.40	27
Joan Miro	0.39	0.33	0.36	21
Leonardo da Vinci	0.18	0.27	0.22	15
Marc Chagall	0.53	0.36	0.43	22
Pablo Picasso	0.14	0.04	0.06	25
Rembrandt	0.32	0.92	0.47	13
Salvador Dali	0.25	0.04	0.07	23
Vincent van Gogh	0.38	0.43	0.40	21
Accuracy			0.34	200

Confusion Matrix:



- The Random Forest model performed slightly better with an accuracy of 34%.
- While some classes such as "Rembrandt" and "Henri Matisse" show relatively high precision and recall, others like "Leonardo da Vinci" and "Salvador" still exhibit low performance metrics.
- Overall, the model struggles with precision and recall for several classes, indicating limitations in correctly classifying images across various artists.

AdaBoost:

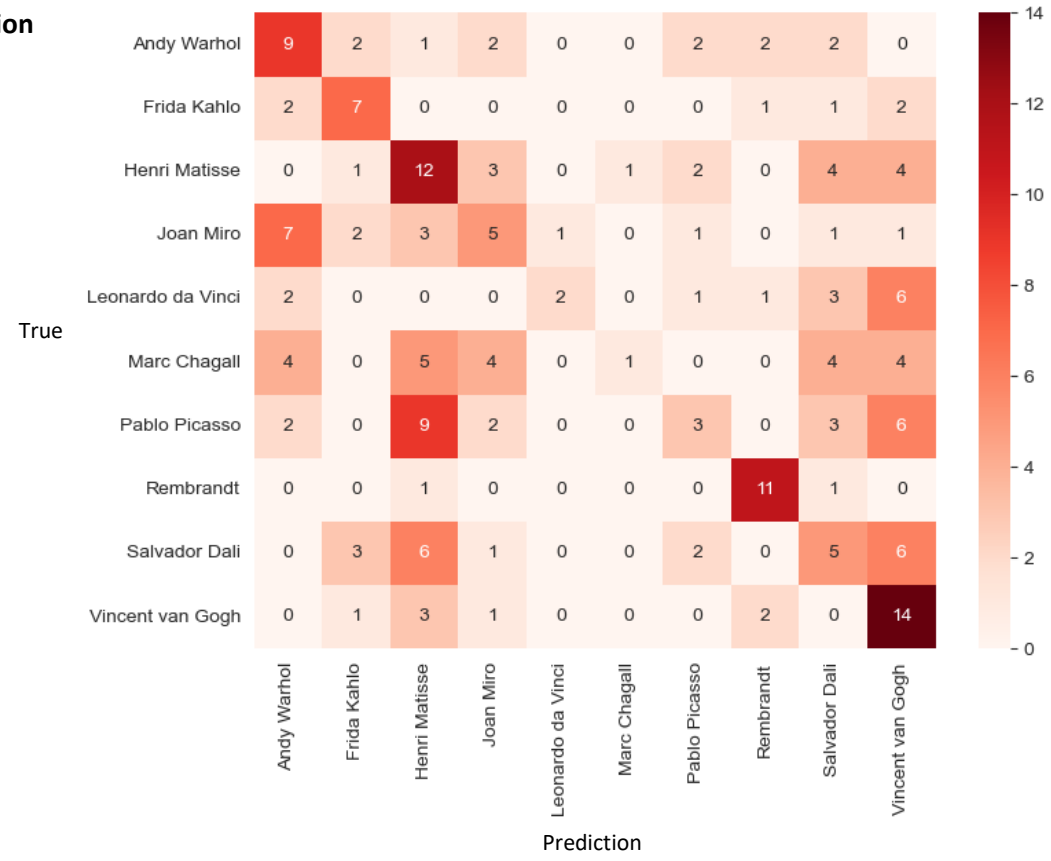
Best Params: 'algorithm': 'SAMME.R', 'learning rate': 0.1, 'n estimators': 40

Empirical Accuracy: 0.455, **True Accuracy**: 0.345

Classification Report:

Class	Precision	Recall	F1-Score	Support
Andy Warhol	0.35	0.45	0.39	20
Frida Kahlo	0.44	0.54	0.48	13
Henri Matisse	0.30	0.44	0.36	27
Joan Miro	0.28	0.24	0.26	21
Leonardo da Vinci	0.67	0.13	0.22	15
Marc Chagall	0.50	0.05	0.08	22
Pablo Picasso	0.27	0.12	0.17	25
Rembrandt	0.65	0.85	0.73	13
Salvador Dali	0.21	0.22	0.21	23
Vincent van Gogh	0.33	0.67	0.44	21
Accuracy			**0.34**	**200**

Confusion Matrix:



- The AdaBoost model achieved a similar accuracy of 34%, with performance metrics comparable to the Random Forest model.
- Classes such as "Rembrandt" and "Frida Kahlo" show improved precision and recall compared to other models, but challenges persist in accurately classifying images for certain artists.
- The model's performance remains inconsistent across different classes, highlighting the need for further refinement.

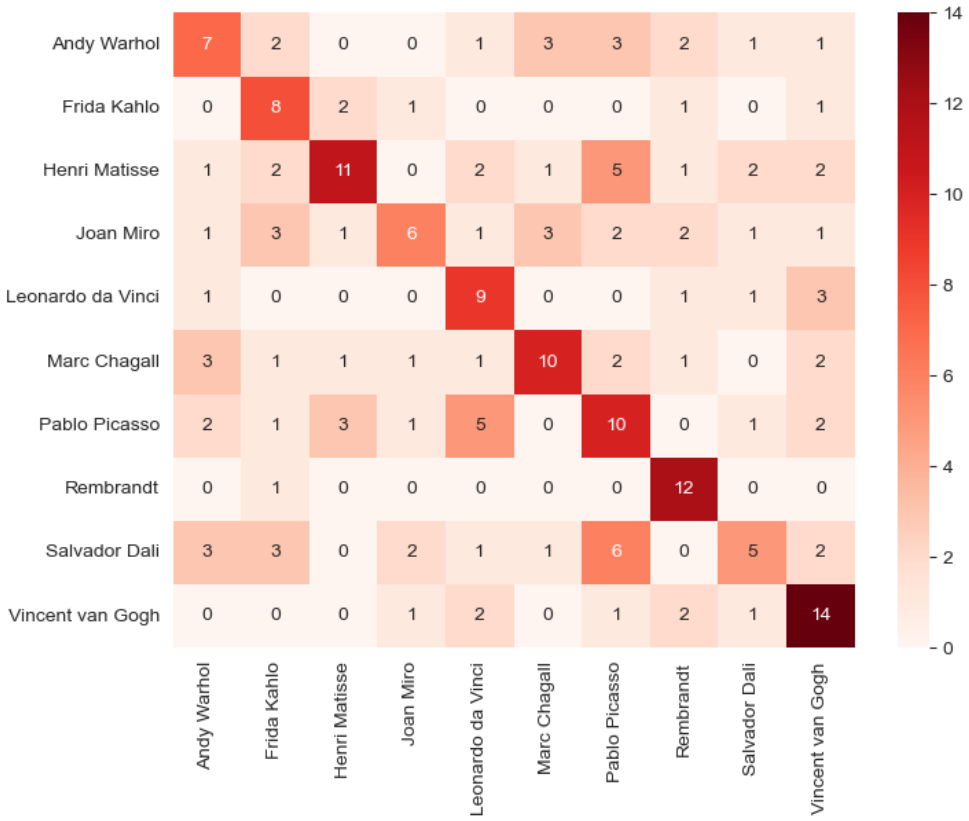
Support Vector Machine (SVM):

Best Params: 'C': 10, 'gamma': 'scale', 'kernel': 'rbf'
Empirical Accuracy: 1.000, **True Accuracy**: 0.460

Classification Report:

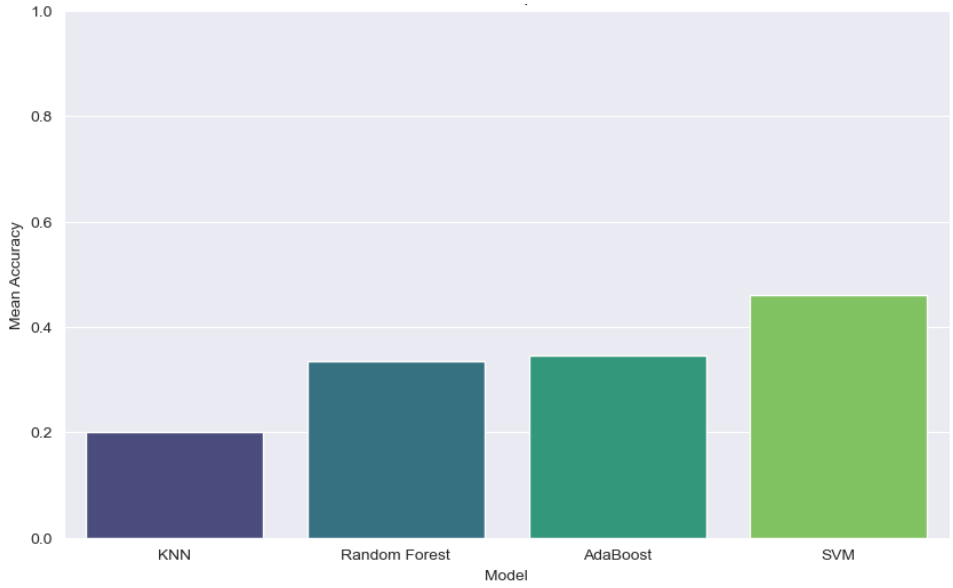
Class	Precision	Recall	F1-Score	Support
Andy Warhol	0.39	0.35	0.37	20
Frida Kahlo	0.38	0.62	0.47	13
Henri Matisse	0.61	0.41	0.49	27
Joan Miro	0.50	0.29	0.36	21
Leonardo da Vinci	0.41	0.60	0.49	15
Marc Chagall	0.56	0.45	0.50	22
Pablo Picasso	0.34	0.40	0.37	25
Rembrandt	0.55	0.92	0.69	13
Salvador Dali	0.42	0.22	0.29	23
Vincent van Gogh	0.50	0.67	0.57	21
Accuracy			**0.46**	**200**

Confusion Matrix:



- The SVM model outperformed the other models with an accuracy of 46%, indicating the highest level of success in correctly classifying images to their respective artists.
- While precision and recall vary across classes, the SVM model shows relatively balanced performance, with notable improvements in classes like "Andy Warhol" and "Frida Kahlo." This could be visually recognized from the confusion matrix tending to be more intense throughout it's diagonal.
- Overall, the SVM model demonstrates the most promising results, suggesting its effectiveness in this task compared to other models tested.

Accuracy Comparison:



- The machine learning models tested in the project varied in their performance, with accuracies ranging from 20% to 46%.
- The SVM model demonstrated the highest accuracy at 46% and exhibited relatively balanced performance across different classes, making it the most effective model for this task.
- While the other models exhibited fine results for certain classes, overall classification accuracy remained relatively low, indicating the complexity of the task.

Conclusion:

The machine learning models tested in this project demonstrated varying degrees of success in classifying paintings to their respective artists based on visual features. While the models exhibited promising performance, particularly the SVM model, overall classification accuracy remained relatively low, suggesting significant challenges in accurately distinguishing between artists' painting styles with the current tools.

Room for improvement exists in several areas, including further exploration of feature engineering, and model architecture. Additionally, incorporating specific knowledge aside images about artists' painting styles and historical context such as year of operation, artistic movement and more could enhance feature extraction and model training processes, potentially leading to more accurate classification results.

In conclusion, while the results of this project provide valuable insights into the application of machine learning in art classification, there is still much to explore and refine.

Bibliography:

1) https://github.com/BarakFinkel/Painting-To-Artist-Classification
2) https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time
3) https://www.youtube.com/watch?v=nK-2k_ENgEc