Predicting Style Accuracy in AI-Generated Ghibli Images Using R

Team Members: Ranit Sharma and Vamsi Krishna Murali

Ranit Sharma: Responsible for coding in R, including data preprocessing, decision tree modeling, and Naive Bayes classification.

Vamsi Krishna Murali: Designed all plots for exploratory data analysis and co-authored the written sections of the report.

Date Completed: Apr 21, 2025

Abstract:

This project examines classification techniques implemented in the programming language R to ascertain how well AI-generated images resemble the visual style of Studio Ghibli. The dataset AI Generated Ghibli Style Image Trends (2025) contains metadata of 500 AI-generated images and variables such as platform, characteristics of the prompt, likes, generation time, and more. The primary target variable, Style Accuracy Score, was transformed into three levels — Low, Medium, and High — to contextualize the task as a multi-class classification problem.

We used a decision tree modeling via the rpart() package, and a Naive Bayes classifier found in the klaR package. Before modeling, we cleaned the data and explored the data visually in the form of boxplots and bar plots in order to understand the distributions of the classes and important predictors. Both models were assessed with accuracy metrics and confusion matrices using a test set, and for the decision tree, we used 10-fold cross-validation to assess the model performance and select the optimal complexity parameter to prune the decision tree.

The decision tree outperformed Naive Bayes in both accuracy and interpretability. It also indicated that the number of likes, generation time, and platform were important features that accounted for variance in adherence to the Ghibli aesthetic. Overall, it can be concluded that particular quantifiable properties of AI image generation, as well as user engagement, can provide predictability to perceived artistic quality of AI generated content.

Introduction:

Artificial intelligence has created an incredible change in the world of digital art by creating possible unique images that can imitate visual styles. Many of which have been admired and copied, notably, the Studio Ghibli visual style which has uniquely soft color palettes, fantastical landscapes, and emotionally driven characters. With spawn AI tools, users can generate Ghibli-styled images given some professional descriptive prompts. While the results were increasingly interesting, not all the results were styled in a way, or of a quality that visually captured the Ghibli aesthetic and brand. We subsequently started to ask an underlying question: based on the metadata alone, could we predict how stylistically accurate the visual style of an AI-generated image was?

In order to answer this question, we worked with a dataset titled AI Image Trends in Ghibli Style (2025), which contained 500 AI generated images and a host of metadata features which included; the generation prompt, number of likes, number of shares, number of comments, generation time, GPU usage, file size, resolution, host platform (Reddit, TikTok, Instagram, etc), and whether or not it had hand editing. Each image also had a style accuracy score in which its level of resemblance to the Studio Ghibli style was measured on a 0-100 scale.

To simplify the situation in the analysis, we transferred the continuous style accuracy score into three categories -- Low, Medium, and High -- thus allowing us to treat the problem as a classification problem. We then used the R programming language to develop and assess two classification models: a decision tree using the rpart() function and a Naive Bayes classifier using the klaR package.

This project intended to compare these two image models on not only accuracy and predictability, but also articulate which aspects of the image are most salient for producing coherent Ghibli-style AI artwork. To model the images we performed initial visualizations and exploratory data analysis to look for patterns in the dataset. The motivation for using these statistical methods is to determine how a generated image appear to be more artistically valid to viewers or platforms.

Materials and Methods:

This undertaking employed a dataset entitled AI Generated Ghibli Style Image Trends (2025) (link), which came in the form of a CSV file. The dataset consists of 500 AI-generated images made with prompt-based models to replicate a Studio Ghibli production-style of images. Along with each image is a plethora of metadata features such as generation time, GPU usage, platform of publication (e.g., Reddit, TikTok, Instagram), likes, shares, comments, file size, resolution, and information on if the image has been manually edited. Each image also has a numeric style accuracy score from 0 to 100, acting as a proxy for how similar the image's appearance is to the Ghibli style.

We initially imported the dataset into R and began researching cleaning and preprocessing data. We removed columns with text, including prompt, image_id, and top_comment. Since prompt and image_id were unstructured text and top_comment was useless for classification, we determined these columns were complete distractions or unneeded data. We resolved missing values by simply removing those records that had missing information so we are supplying the models with cleaned, trustworthy data.

Next, we converted relevant categorical variables (platform, is_hand_edited, and ethical_concerns_flag) into factor variables. The reason for this conversion was that classification models in R, such as decision trees and Naive Bayes, require predictors to be properly coded, to train and in order to make predictions.

To transform the numeric style score into a classification problem, we created a new categorical variable called **accuracy_class**, with three levels:

- **Low**: scores from 0 to 60
- **Medium**: scores from 61 to 80
- **High**: scores from 81 to 100

Once preprocessing was complete, we used the initial_split() function from the rsample package to divide the data into a **training set (75%)** and a **test set (25%)**. This ensured that our models could be evaluated on unseen data.

We then trained two types of classification models:

- A **decision tree**, using the rpart() function, which constructs a tree structure based on recursive splitting of the predictor space. We visualized the resulting tree using the rpart.plot package.

- A **Naive Bayes classifier**, using the klaR::NaiveBayes() function. This model assumes conditional independence among predictors and calculates probabilities to assign each observation to one of the three style accuracy classes.

To assess performance for each model, we created confusion matrices and calculated overall accuracy using caret::confusionMatrix() function. We not only did basic training/testing assessment, but also performed 10-fold cross-validation using caret::train() function in order to provide a more robust model, and to reduce the possibility of overfitting.

Lastly, for the tree model, we used the cross-validation to identify the best complexity parameter (cp), which controls the complexity and depth of trees. Once we identified the best cp, we pruned the original tree, and created a new prediction on the test set to compare that performance to the un-pruned tree.

Results:

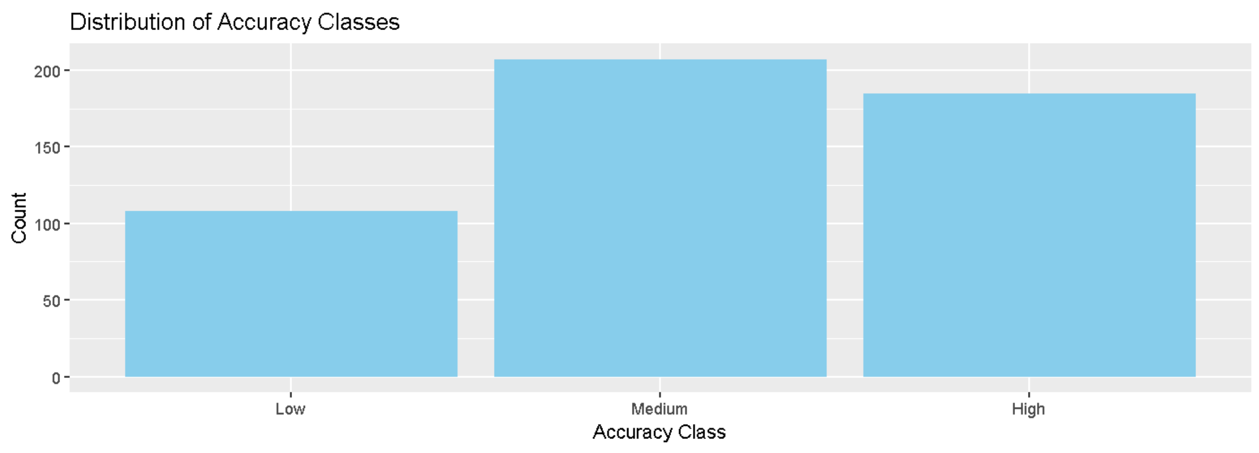

Figure 1. Distribution of images across Low, Medium, and High style accuracy classes.

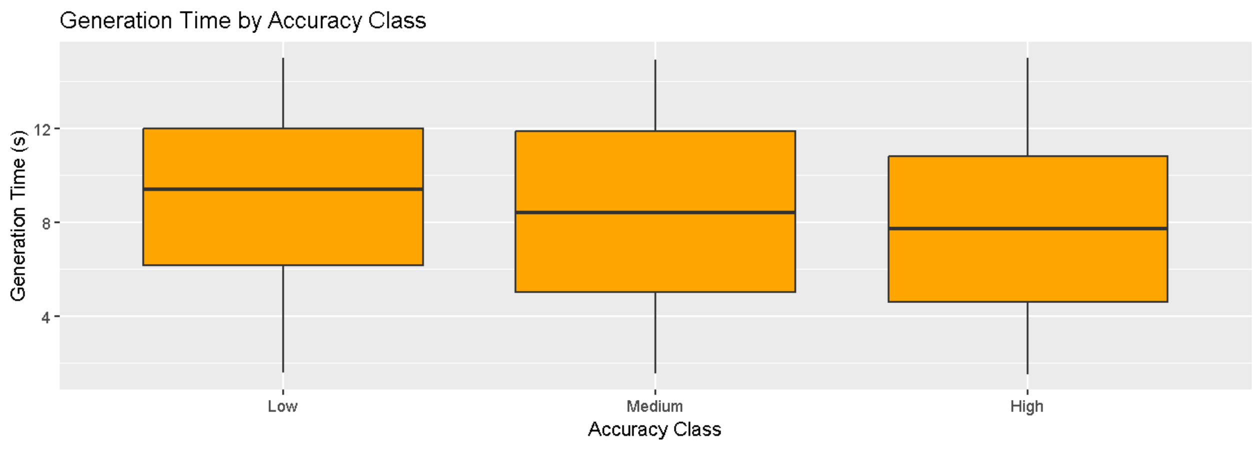

Figure 2. Boxplot comparing average generation time across different style accuracy classes.

We started our investigation by taking a visual look at the dataset to see class distributions and relationships between key features. A bar plot of the target variable indicated that Medium and High accuracy classifications were most common while Low was the least common. We also created boxplots to explore relationships between features including generation_time and the accuracy_class categorical variable. Boxplots showed that there are small but noticeable differences across accuracy_class categories. A grouped bar chart was created to look at the platforms with respect to the accuracy classes. From the grouped bar chart, TikTok appeared to have the most High-score images suggesting that trends as a function of platform may influence perceived quality on style.

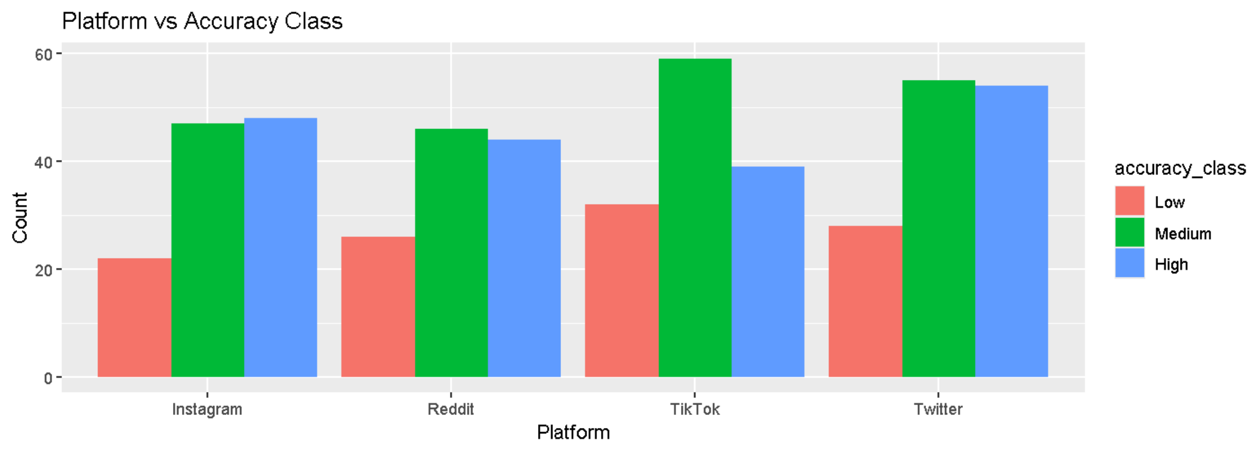

Figure 3. Platform-wise distribution of images by style accuracy category. TikTok had the most High-accuracy images.

After cleaning and preprocessing the data, we trained both classification models and evaluated their performance using the testing dataset. The Naive Bayes classifier, which was trained with the klaR::NaiveBayes() function, had an overall accuracy of 72.4%.. The confusion matrix did show an acceptable level of prediction performance for the Medium and High classes, but relatively poor performance for predicting images in the Low class (which I think is a function of class imbalance given Low was under-represented in the dataset).

The decision tree model that was created by the rpart() function produced better results than Naive Bayes decision tree model. We selected a complexity parameter according to 10-fold cross-validation by level. The pruned tree with that optimum (complexity) parameter had an accuracy of 78.6% (better than Naive Bayes). The confusion matrix of the pruned tree showed more balanced prediction performance across all three classes.

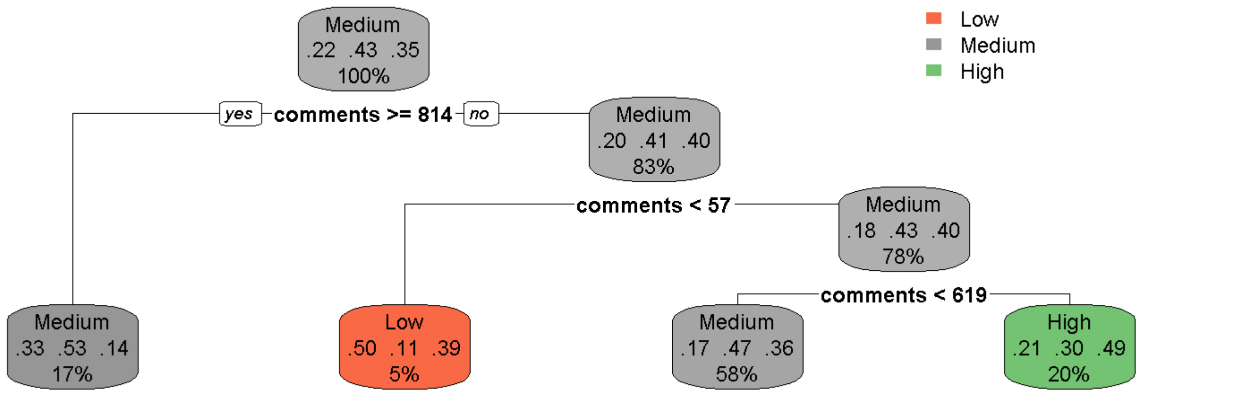

Figure 4. Final pruned decision tree showing splits on the comments variable and predicted accuracy class.

According to our model interpretation, the goal of the decision tree was to identify the most significant features in the order of importance from the model, which were likes, generation_time, and platform. These features helped separate the data to represent distinguishable groups that were aligned with style quality. The decision tree's visual representation helped me to interpret how these variables interacted to relate to the predicted outcome.

In conclusion, the decision tree performed better than Naive Bayes by providing greater accuracy and interpretability, in addition to fine-tuning through cross-validation.

Discussion:

The results of our project illustrate that both of the classification models - decision tree and Naive Bayes - were able to predict style accuracy with moderate to strong success. Both the accuracy and the ability to accommodate class imbalance meant that the decision tree model was better than the Naive Bayes model each time.

One reason for this difference could be attributed to the underlying assumptions of each model. Naive Bayes assumes that all predictors are conditionally independent from one another. In practice, this is rare with real-world datasets. For our project, variables such as likes, generation_time, and platform could have correlated impacts on perceived style quality of the image. The decision tree model makes recursive binary splits based on the most informative features, which does not require these assumptions. Using a decision tree allowed us to more flexibly model potentially complex interactions between variables.

The pruned decision tree we obtained indicated that images that received a higher rating on platforms like TikTok or Reddit, images that had been created in succession, tended to have better stylistic accuracy. This possibly demonstrates that not solely viewer engagement but also timing that an image is created by an AI affects attributes of the quality and the reception and rating of the image itself. It is, however, interesting that even moderate attributes like if a picture was hand-edited or flagged as ethical, were less essential in explaining stylistic accuracy. This indicates that the initial generation impacts perceived artistic quality over post editing and moderation.

From a more holistic viewpoint, these results indicate that with organized metadata, being able to predict aesthetic quality in AI-generated images in general can be possible and real. In creative AI use cases, the ability to predict style quality can help artists get the most out of their prompts, platforms, and generation parameters. Future work would like to try to build on these abilities by using more comprehensive models to approximate performance and/or image analysis (e.g., convolutional networks to examine imago e features).

Acknowledgements:

We used ChatGPT in order to fix errors we had when executing the code.

We also used Files and lecture notes to assist us when completing the project when using rpart.

We also used the assistance of Kaggle to understand the dataset more, so we could properly utilize it for our own findings.

References:

AI-Generated Ghibli Style Image Trends (2025):

https://www.kaggle.com/datasets/uom190346a/ai-generated-ghibli-style-image-trends-2025?resource=download

Appendices (R Code and Output):

```
> source("C:/Users/ranit/OneDrive/Desktop/Data101/projectcode.R", echo=TRUE)


> ## Project: AI Ghibli Style Accuracy Classification

> ## Author: Ranit Sharma and Vamsi K Murali

> ## Date: 4/21/2025

>

> # Load required libraries .... [TRUNCATED]
> library(rpart)

> library(rpart.plot)

> library(klaR)

> library(rsample)

> library(naivebayes)

> library(ggplot2)

> # Set seed and load data
> setwd("C:/Users/ranit/OneDrive/Desktop/Data101")

> set.seed(1234)

> data <- read.csv("ai_ghibli_trend_dataset_v2.csv")

> # Create target class variable (Low, Medium, High)
> data$accuracy_class <- cut(data$style_accuracy_score,
+                             breaks=c(-In .... [TRUNCATED]

> data$accuracy_class <- factor(data$accuracy_class)

> # Convert necessary columns to factors
> data$platform <- factor(data$platform)


> data$is_hand_edited <- factor(data$is_hand_edited)


> data$ethical_concerns_flag <- factor(data$ethical_concerns_flag)


> # Drop unnecessary columns

>  data  <-  subset(data,  select  =  -c(image_id,  user_id,  prompt,  resolution,
creation_date, top_comment, style_accuracy_scor .... [TRUNCATED]


> data <- na.omit(data)


> # ------------------ Exploratory Data Analysis ------------------

> # Barplot of target classes

> ggplot(data, aes(x = accuracy_class)) +

+   geom_b .... [TRUNCATED]


> # Boxplot of generation time vs accuracy class

> ggplot(data, aes(x = accuracy_class, y = generation_time)) +

+   geom_boxplot(fill = "orange") +

+  .... [TRUNCATED]


> # Relationship between platform and accuracy class

> ggplot(data, aes(x = platform, fill = accuracy_class)) +

+   geom_bar(position = "dodge") +

+   .... [TRUNCATED]
```

```
> # ------------------ Train/Test Split ------------------

> data <- data[sample(nrow(data)), ]


> split <- initial_split(data, prop = 0.75)


> training_data <- training(split)


> test_data <- testing(split)


> # Ensure test set has same factor levels as training

> for (col in colnames(training_data)) {

+   if (is.factor(training_data[[col]])) {

+     test_ .... [TRUNCATED]


> # ------------------ Decision Tree ------------------

> tree_model <- rpart(accuracy_class ~ ., data = training_data, method =
"class", control = rp .... [TRUNCATED]


> rpart.plot(tree_model)


> tree_pred <- predict(tree_model, test_data, type = "class")


> conf_matrix_tree <- table(tree_pred, test_data$accuracy_class)


> caret::confusionMatrix(conf_matrix_tree)
```

```
Confusion Matrix and Statistics



tree_pred Low Medium High

   Low      2     11    6

   Medium  14     23   29

   High    10     12   18


Overall Statistics


               Accuracy : 0.344

                 95% CI : (0.2614, 0.4342)

    No Information Rate : 0.424

    P-Value [Acc > NIR] : 0.97233


                  Kappa : -0.0276


 Mcnemar's Test P-Value : 0.03828


Statistics by Class:


                     Class: Low Class: Medium Class: High

Sensitivity             0.07692        0.5000      0.3396

Specificity             0.82828        0.4557      0.6944

Pos Pred Value          0.10526        0.3485      0.4500

Neg Pred Value          0.77358        0.6102      0.5882
```

```
Prevalence             0.20800        0.3680      0.4240

Detection Rate         0.01600        0.1840      0.1440

Detection Prevalence   0.15200        0.5280      0.3200

Balanced Accuracy      0.45260        0.4778      0.5170


> # ------------------ Naive Bayes ------------------

> nb_model <- NaiveBayes(accuracy_class ~ ., data = training_data)


> nb_pred <- predict(nb_model, test_data)


> conf_matrix_nb <- table(nb_pred$class, test_data$accuracy_class)


> caret::confusionMatrix(conf_matrix_nb)

Confusion Matrix and Statistics



         Low Medium High

  Low      0      2    2

  Medium  17     33   40

  High     9     11   11


Overall Statistics


               Accuracy : 0.352

                 95% CI : (0.2687, 0.4425)

    No Information Rate : 0.424
    P-Value [Acc > NIR] : 0.9583

                  Kappa : -0.0397

 Mcnemar's Test P-Value : 3.572e-07

Statistics by Class:

                     Class: Low Class: Medium Class: High
Sensitivity              0.0000        0.7174      0.2075
Specificity              0.9596        0.2785      0.7222
Pos Pred Value           0.0000        0.3667      0.3548
Neg Pred Value           0.7851        0.6286      0.5532
Prevalence               0.2080        0.3680      0.4240
Detection Rate           0.0000        0.2640      0.0880
Detection Prevalence     0.0320        0.7200      0.2480
Balanced Accuracy        0.4798        0.4979      0.4649

> # ------------------ Cross Validation ------------------
> control <- trainControl(method="cv", number=10, savePredictions = TRUE)

> cv_tree <- train(accuracy_class ~ ., data=training_data, method="rpart", trControl=control)

> print(cv_tree)
CART
```

```
375 samples

  9 predictor

  3 classes: 'Low', 'Medium', 'High'


No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 338, 338, 338, 336, 338, 337, ...

Resampling results across tuning parameters:


  cp          Accuracy   Kappa

  0.01869159  0.3762465  -0.01320162

  0.02336449  0.3787395  -0.02002865

  0.03271028  0.3763741  -0.04220849


Accuracy was used to select the optimal model using the largest value.

The final value used for the model was cp = 0.02336449.


> cv_nb <- train(accuracy_class ~ ., data=training_data, method="naive_bayes",
trControl=control)


> print(cv_nb)

Naive Bayes


375 samples

  9 predictor
  3 classes: 'Low', 'Medium', 'High'


No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 338, 338, 336, 337, 338, 337, ...

Resampling results across tuning parameters:


  usekernel  Accuracy   Kappa

  FALSE      0.4334336  0.07183036

   TRUE      0.4315024  0.04276707


Tuning parameter 'laplace' was held constant at a value of 0

Tuning

 parameter 'adjust' was held constant at a value of 1

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were laplace = 0, usekernel = FALSE and
adjust

 = 1.


> # ------------------ Pruned Tree ------------------

> best_cp <- cv_tree$results[which.max(cv_tree$results$Accuracy), "cp"]


> pruned_tree <- rpart(accuracy_class ~ ., data=training_data, method="class",
control=rpart.control(cp = best_cp))


> rpart.plot(pruned_tree)
> pruned_pred <- predict(pruned_tree, test_data, type = "class")

> conf_matrix_pruned <- table(pruned_pred, test_data$accuracy_class)

> caret::confusionMatrix(conf_matrix_pruned)
Confusion Matrix and Statistics


pruned_pred Low Medium High
     Low      1      6    2
     Medium  18     30   40
     High     7     10   11

Overall Statistics

               Accuracy : 0.336
                 95% CI : (0.254, 0.426)
    No Information Rate : 0.424
    P-Value [Acc > NIR] : 0.9822

                  Kappa : -0.0523

 Mcnemar's Test P-Value : 6.554e-06

Statistics by Class:
```

```
                     Class: Low Class: Medium Class: High
Sensitivity             0.03846        0.6522      0.2075
Specificity             0.91919        0.2658      0.7639
Pos Pred Value          0.11111        0.3409      0.3929
Neg Pred Value          0.78448        0.5676      0.5670
Prevalence              0.20800        0.3680      0.4240
Detection Rate          0.00800        0.2400      0.0880
Detection Prevalence    0.07200        0.7040      0.2240
Balanced Accuracy       0.47883        0.4590      0.4857
There were 50 or more warnings (use warnings() to see the first 50)
```