**Astronomical Image Classification Using R**

**Introduction**

The analysis of astronomical images involves processing and interpreting data to understand the nature and behavior of celestial bodies. This project uses R to analyze a dataset of astronomical images obtained from Kaggle (Melcore., 2018), focusing on preprocessing, feature extraction, and classification. The project aims to classify images into three categories: Galaxy, Planet, and Space. Inspired by methodologies similar to those discussed in the context of Python-based astronomical data analysis (Ivezić et al., 2014) this document details the problem, the solution implemented, and the various visualizations used to interpret the data. Additionally, this program can be further modified to create a web UI, making it an interactive educational tool for young children to learn about space.

**Program Interface**

The program is executed in the R programming environment. To run the program, ensure that R and RStudio are installed on your system. The necessary packages include imager, ggplot2, dplyr, randomForest, and magick. The program is executed by sourcing the main script file, which contains all the necessary functions and model training code. The user can start the analysis by entering the image name and can do it for any number of images. To terminate the program, simply close the RStudio session.

**Program Execution**

**Loading the Dataset**

The dataset of astronomical images is loaded from from Kaggle (Melcore, 2018). Each image is preprocessed to a uniform size and converted to grayscale. The labels are mapped to three categories: Galaxy, Planet, and Space.

**Training the Model**

A random forest model is trained on the preprocessed images using the mapped labels. The model is evaluated using a confusion matrix and ROC curves to assess its performance.

**Analyzing New Images**

User can analyze an image by running the program and then entering the image name once prompted. The function outputs analysis, including color intensity histograms, edge detection, and the predicted category of the image. New images can be analyzed without having to re-train the data.

**Input and Output**

**1. Input: Training Data**

* **Image Files:** The program accepts image files in various formats (e.g., JPG, PNG) from a specified directory for training (path specified in the script and can be modified).
* **Labels:** A CSV file containing the image filenames and their corresponding labels.

**2. Input: User Input**

* The user is prompted to input the image name they wish to analyze.
* After the end of the first analysis, the user is given the option of analyzing a new image.

**1. Output: Training Data**

* **Visualizations for Analysis:** The program generates several graphs, confusion matrix, ROC curves, model accuracy over time, and category distribution.

**2. Output: For the User**

* **Predictions:** The program outputs the predicted category of new images.
* **Plot:** Color Intensity Histogram and Edge Detection Plot.
* **General Image Analysis:** Image dimensions and number of color channels

**Program Structure**

**Preprocessing**

• **Function:** preprocess\_image

• **Description:** Resizes images to 64x64 pixels and converts them to grayscale.

**Model Training**

• **Function:** train\_model

• **Description:** Trains a random forest classifier on the preprocessed images.

**Image Analysis**

• **Function:** analyze\_image

• **Description:** Analyzes a new image, generates visualizations, and predicts the category.

**Repeat Analysis**

• **Function:** repeat\_analysis

• **Description:** Function to repeatedly ask for new image names and analyze them

**Examples**

**Example 1: User wants to analyze the image called “nebula.png”**

The user runs the program, and the following happens:

**Preprocessing the data:** For the purpose of training the data, the program converts the images into a standardized format that can be used for training and analysis. The images are resized to 64x64 pixels and converted to grayscale to reduce complexity and focus on essential features. Labels are also mapped to three categories: Galaxy, Planet, and Space. The following graphs provide a visual understanding of the dataset distribution and the model’s performance on this preprocessed data.

**A graph with different colored squares

Description automatically generated**

Figure 1: This graph shows the distribution of the dataset across three categories: Galaxy, Planet, and Space. It highlights the imbalance in the dataset, with a higher number of images classified under the ‘Space’ category.

**A screenshot of a graph

Description automatically generated**

Figure 2: This confusion matrix visualizes the performance of the classification model by comparing actual versus predicted categories. It helps identify where the model is performing well and where it might be making errors.

**Model accuracy:** Once the data is preprocessed, the next step is to train and evaluate the model. A random forest classifier is used to predict the categories of the images. The model’s accuracy is assessed using various metrics, including ROC curves and the Out-Of-Bag (OOB) error rate. These evaluations help in understanding how well the model can distinguish between different categories and its overall performance.

**A graph of a positive curve

Description automatically generated**

Figure 3: The ROC curve (Receiver Operating Characteristic curve) evaluates the model’s performance in distinguishing between different categories by plotting the true positive rate against the false positive rate.

**A graph showing a number of trees

Description automatically generated**

Figure 4: This graph displays the model’s Out-Of-Bag (OOB) error rate as the number of trees in the random forest increases. It shows how the model’s accuracy improves with more trees.

**Analyzing the image**: After the training is complete, the user is prompted to enter an image name. The user inputs, “nebula.png”. The program runs the analysis and outputs: image dimensions and color channels; saves intensity histogram and edge detection plot (and prints the path they are saved at). Then the program predicts the category the image belongs to, in this case it is “space” and is accurately identified (since it is neither a galaxy nor a planet). The user is prompted to enter another image if they wish to analyze it. The user enters “no” The program prints “Your analysis is complete” to indicate that the analysis is finished.

A blue and white nebula

Description automatically generated with medium confidence

Figure 5: Image of a Nebula input by User

A black background with white lines

Description automatically generated

Figure 6: The edge detection graph highlights the edges within the analyzed image, helping to identify the shapes and boundaries of celestial bodies for deeper understanding of the mathematical properties of an the object.

A graph of different colored lines

Description automatically generated with medium confidence

Figure 7: This histogram output displays the distribution of pixel intensities for each color channel (red, green, and blue), providing insights into the color composition of the image.

A screenshot of a computer program

Description automatically generated

Figure 8: This output shows user input, the detailed analysis of a new image, and prompt to input another image.

**Example 2: User enters “yes” to analyze another image.**

The user wishes to analyze another image named “planet.png”. When prompted “Do you want to analyze another image? (yes/no)” The user enters yes, this time the data is not retrained, only the image analysis runs. However, in this scenario, the model is not doing a good job on predicting the planet category, improvements for this inaccuracy have been discussed in the next section. The following outputs are obtained.

**A planet with a black background

Description automatically generated**

Figure 9: User inputs the image name "planet.jpg"

**A graph of different colored lines

Description automatically generated with medium confidenceA black and white image of a planet

Description automatically generated**

Figure 10: Edge Detection and Histogram for the planet

**A screenshot of a computer program

Description automatically generated**

Figure 11: This time the analysis inaccurately predicts the category, but other analysis is accurate.

**Improvements and Extensions**

Future improvements could include expanding the category beyond the three: Space, Planet and Galaxy, where the user can obtain analysis of more complex images that may not be obvious otherwise. Using more advanced models like convolutional neural networks (CNNs) along with larger datasets can help improve accuracy of the model. Additionally, incorporating metadata such as image capture date and location could provide more context for the analysis. Further extensions could involve real-time image processing and integration with astronomical databases. Moreover, this project could be extended to develop an educational tool aimed at young children, helping them learn about astronomical data in an interactive way. By creating a user-friendly web UI, the program can become more accessible, engaging, and beneficial for users, providing a hands-on learning experience about space.

**Difficulties Encountered**

The primary difficulties included handling images with varying dimensions and formats, ensuring consistent preprocessing, and managing computational resources for training the model. Additionally, fine-tuning the random forest model to achieve optimal performance required extensive experimentation.

**Conclusion**

This project demonstrates the application of R for analyzing and classifying astronomical images. The use of various visualizations and machine learning techniques provides valuable insights into the dataset. The project highlights the potential of R for handling complex image processing tasks and lays the groundwork for future enhancements. Additionally, by focusing on educational applications, this project aims to make astronomical data analysis more accessible and engaging for young children, providing an interactive and user-friendly platform that benefits users by enhancing their understanding of space.

**References**

Encyclopædia Britannica, Inc. (n.d.). Nebula. In Encyclopædia Britannica. Retrieved July 15, 2024, from https://www.britannica.com/science/nebula

Ivezić, Ž., Connolly, A. J., VanderPlas, J. T., & Gray, A. (2014). Statistics, data mining, and machine learning in astronomy: A practical Python guide for the analysis of survey data. Princeton University Press.

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. R News, 2(3), 18-22. URL https://CRAN.R-project.org/doc/Rnews/.

Melcore. (2018). Astronomy Picture of the Day dataset. Kaggle. <https://www.kaggle.com/datasets/melcore/astronomy-picture-of-the-day?select=APOC64>

R Core Team. (2024). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. URL https://ggplot2.tidyverse.org.

Xavier, T. (2019). Using Python to analyze astronomy data: TESS exoplanet search. Medium. <https://xavnet2.medium.com/using-python-to-analyze-astronomy-data-tess-exoplanet-search-b858e23dfa49>

Xie, Y., Cheng, J., Tan, X., & Allaire, J. J. (2024). imager: Image Processing Library Based on CImg. R package version 1.5.3. URL https://cran.r-project.org/package=imager.

**Appendices**

**Appendix A: Source Code**

# Load necessary libraries

library(imager)

library(dplyr)

library(caret)

library(randomForest)

library(magick)

library(ggplot2)

library(pROC)

library(plotly)

library(colorspace)

# Function to preprocess images

preprocess\_image <- function(file\_path) {

img <- load.image(file\_path) %>%

resize(64, 64) %>% # Resize to 64x64

grayscale() # Convert to grayscale

as.vector(img) # Flatten the image

}

# Load the dataset

data\_dir <- "/Users/shreya/Desktop/archive/APOC64"

image\_files <- list.files(data\_dir, pattern = "\*.jpg", full.names = TRUE)

# Load labels with proper separator

labels <- read.csv("/Users/shreya/Desktop/archive/infos.csv", stringsAsFactors = FALSE, sep = ";", header = FALSE, fill = TRUE, skip = 1)

colnames(labels) <- c("Filename", "Title")

# Define the categories

planets <- c("Earth", "Mars", "Jupiter", "Saturn", "Venus", "Mercury", "Neptune", "Uranus")

# Function to map specific labels to general categories

map\_labels <- function(title) {

if (grepl("galaxy", title, ignore.case = TRUE)) {

return("Galaxy")

} else if (any(sapply(planets, function(planet) grepl(planet, title, ignore.case = TRUE)))) {

return("Planet")

} else {

return("Space")

}

}

# Apply mapping to create a new category column

labels$category <- sapply(labels$Title, map\_labels)

# Check the category distribution (Graph 1)

category\_dist <- ggplot(labels, aes(x = category, fill = category)) +

geom\_bar() +

labs(title = "Category Distribution", x = "Category", y = "Count") +

theme\_minimal()

print(category\_dist)

# Preprocess all images

image\_data <- lapply(image\_files, preprocess\_image)

image\_matrix <- do.call(rbind, image\_data)

# Combine image data with new labels

image\_df <- data.frame(image\_matrix)

image\_df$category <- factor(labels$category)

# Split the data into training and testing sets

set.seed(123)

train\_index <- createDataPartition(image\_df$category, p = 0.8, list = FALSE)

train\_data <- image\_df[train\_index, ]

test\_data <- image\_df[-train\_index, ]

# Train a random forest model

rf\_model <- randomForest(category ~ ., data = train\_data, ntree = 100)

# Make predictions

predictions <- predict(rf\_model, newdata = test\_data)

# Confusion matrix (Graph 2)

conf\_matrix <- confusionMatrix(predictions, test\_data$category)

conf\_matrix\_plot <- as.data.frame(conf\_matrix$table)

conf\_matrix\_heatmap <- ggplot(conf\_matrix\_plot, aes(x = Reference, y = Prediction, fill = Freq)) +

geom\_tile() +

scale\_fill\_gradient(low = "white", high = "blue") +

labs(title = "Confusion Matrix", x = "Actual", y = "Predicted") +

theme\_minimal()

print(conf\_matrix\_heatmap)

# ROC curve (Graph 3)

roc\_curves <- multiclass.roc(test\_data$category, as.numeric(predictions))

roc\_list <- roc\_curves$rocs

# Plot ROC curves for each class

roc\_plot <- ggplot() +

labs(title = "ROC Curves", x = "False Positive Rate", y = "True Positive Rate") +

theme\_minimal()

for (roc in roc\_list) {

roc\_df <- data.frame(

specificity = 1 - roc$specificities,

sensitivity = roc$sensitivities

)

roc\_plot <- roc\_plot +

geom\_line(data = roc\_df, aes(x = specificity, y = sensitivity))

}

print(roc\_plot)

# Accuracy over time (Graph 4)

model\_accuracy <- rf\_model$err.rate

accuracy\_df <- data.frame(Trees = 1:nrow(model\_accuracy), Error = model\_accuracy[, "OOB"])

accuracy\_plot <- ggplot(accuracy\_df, aes(x = Trees, y = Error)) +

geom\_line(color = "blue") +

labs(title = "Model Accuracy Over Time", x = "Number of Trees", y = "OOB Error Rate") +

theme\_minimal()

print(accuracy\_plot)

# Function to analyze a new image

analyze\_image <- function(image\_path, model) {

# Load the image

img <- image\_read(image\_path)

img\_name <- basename(image\_path)

cat("Analyzing image:", img\_name, "\n")

# Get image dimensions

img\_info <- image\_info(img)

cat("Image dimensions:", img\_info$width, "x", img\_info$height, "\n")

# Convert the image to an imager object for further analysis

imgr <- load.image(image\_path)

# Check the number of color channels

num\_channels <- spectrum(imgr)

cat("Number of color channels:", num\_channels, "\n")

# Remove alpha channel if it exists

if (num\_channels == 4) {

imgr <- imgr[,,1:3,drop=FALSE]

num\_channels <- 3

}

# Ensure the image has exactly three color channels (RGB)

if (num\_channels != 3) {

stop("Error: Image does not have exactly three color channels")

}

# Separate color channels and convert to data frames

red\_df <- as.data.frame(as.vector(imgr[,,1])) %>% mutate(channel = "Red")

green\_df <- as.data.frame(as.vector(imgr[,,2])) %>% mutate(channel = "Green")

blue\_df <- as.data.frame(as.vector(imgr[,,3])) %>% mutate(channel = "Blue")

# Combine data frames

color\_df <- bind\_rows(

red\_df %>% rename(value = `as.vector(imgr[, , 1])`),

green\_df %>% rename(value = `as.vector(imgr[, , 2])`),

blue\_df %>% rename(value = `as.vector(imgr[, , 3])`)

)

# Plot color intensity histograms

color\_histogram <- ggplot(color\_df, aes(x = value, fill = channel)) +

geom\_histogram(bins = 30, alpha = 0.7, position = "identity") +

facet\_wrap(~channel, scales = "free") +

labs(title = "Color Intensity Histograms", x = "Intensity", y = "Frequency") +

theme\_minimal()

# Save the color histogram to the desktop

color\_histogram\_path <- "/Users/shreya/Desktop/color\_intensity\_histogram.png"

ggsave(color\_histogram\_path, plot = color\_histogram, width = 8, height = 6)

cat("Color intensity histogram saved to:", color\_histogram\_path, "\n")

# Edge detection

edges <- imgradient(imgr, "xy") %>% enorm()

# Convert edges to data frame for plotting

edges\_df <- as.data.frame(edges)

# Plot edge detection result

edge\_plot <- ggplot(edges\_df, aes(x = x, y = y, fill = value)) +

geom\_tile() +

scale\_fill\_gradient(low = "black", high = "white") +

labs(title = "Edge Detection", x = "X", y = "Y") +

theme\_minimal()

# Save the edge detection plot to the desktop

edge\_plot\_path <- "/Users/shreya/Desktop/edge\_detection\_plot.png"

ggsave(edge\_plot\_path, plot = edge\_plot, width = 8, height = 6)

cat("Edge detection plot saved to:", edge\_plot\_path, "\n")

# Preprocess the image for prediction

img\_preprocessed <- imgr %>%

resize(64, 64) %>%

grayscale()

# Flatten the preprocessed image if necessary

img\_vector <- as.vector(img\_preprocessed)

# Check the dimensions of the flattened image vector

if (length(img\_vector) != 64\*64) {

stop("Error: Flattened image does not have the expected dimensions")

}

# Create a data frame for prediction

img\_matrix <- as.data.frame(t(img\_vector))

colnames(img\_matrix) <- colnames(train\_data)[-ncol(train\_data)] # Match training data columns

# Predict the category

category\_prediction <- predict(model, newdata = img\_matrix)

# Output the prediction with a sentence

cat("Predicted category:", category\_prediction, "\n")

cat("Your image is a", tolower(category\_prediction), ".\n")

}

# Function to repeatedly ask for image names and analyze them

repeat\_analysis <- function(model) {

repeat {

image\_name <- readline(prompt = "Enter the image name (e.g., space.jpg): ")

image\_path <- file.path("/Users/shreya/Desktop", image\_name)

analyze\_image(image\_path, model)

another <- readline(prompt = "Do you want to analyze another image? (yes/no): ")

if (tolower(another) != "yes") {

break

}

}

cat("Your analysis is complete.\n")

}

# Start the analysis process

repeat\_analysis(rf\_model)