Mapping Crop-types using Available Satellite RS Data and VGI In-situ Data in R

Subin Shrestha

2023-11-06

Executice Summary

During the project work we have came across many R packages and studied the function and uses of them. We also imported different data sets into R then pre-processed the data by removing unnecessary columns and omitting NaNs. We also combined some data frames as per requirement and visualized them along with shape file of the districts of Madhesh Pradesh using leaflet map. We downloaded Landsat 8 image and calculated the NDVI from it and merged with in-situ data in order to compare actual crop type and classification result of crop type. Then it is divided into train(70%) and test(30%) subset and classified using Random Forest Model. A confusion matrix is used to evaluate the performance of a classification algorithm. From the confusion matrix, various evaluation metrics such as Accuracy, Precision, Recall, F1-Score is calculated. Despite the limitation of data size and quality of Remote Sensing Data 71.87% of overall accuracy is achieved which is satisfactory. Result for Paddyrice is found to be most accurate and poor for the Sugercane. Most of Sugercane was predicted as Paddyrice and Orchid in this model.

Table of Content

Introduction

This project is focused on mapping the crop types in the Madesh province of Nepal using a combination of available in-situ data collected through volunteer geographic information (VGI) and satellite remote sensing data. This approach enhances Land Use and Land Cover (LULC) mapping by improving accuracy, offering finer details, validating information, providing seasonal insights, mitigating data gaps due to cloud cover, and allowing for more precise classification.

The in-situ data is collected from October 23, 2021, to January 6, 2022, and the spatial coverave is the Madhesh Province in Nepal. Around 600 VGI sample data is collect and then geo-referenced with the WGS84 providing latitude and longitude coordinates. For satellite remote sensing data the Landsat 8 images were used.

Objective

The main objective of this project is to explore and visualize the in-situ data collected in Madhesh Provice of Nepal with VGI method and satellite remote sensing data, process and filter the data as requirement, apply machine learning algorithms to train models using collected data, classify crop types and evaluate the classification performance in R. It is essential for informed decision-making in agriculture, environmental conservation, disaster response, and policy formulation. It empowers stakeholders with accurate, timely, and spatially explicit information, contributing significantly to sustainable development and food security.The main objectives of this report are as follows:

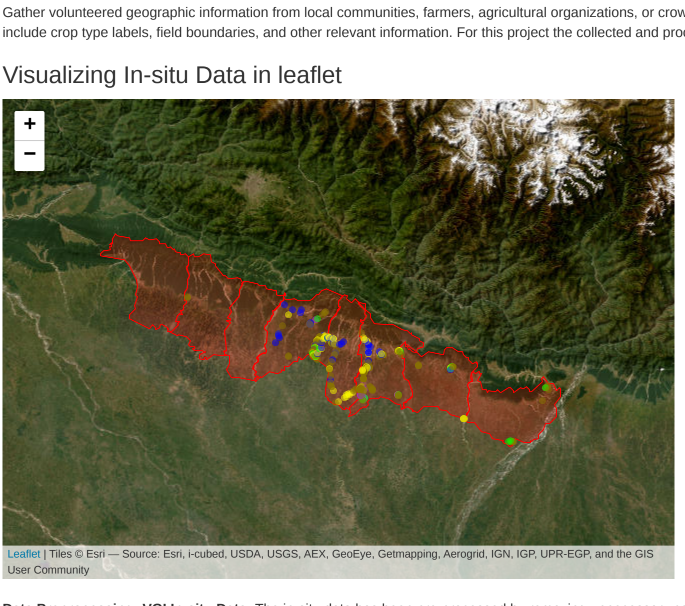
- To analyze the feasibility and effectiveness of classifying crop types using satellite remote sensing data and VGI.
- To visualize the in-situ data over the map and compare it with the NVDI value obtained from the satellite data.
- To classify the crop type using Random Forest Classification model.
- To assess the accuracy and performance of the classification model with confusion matrix, overall accuracy and individual class performance.

Methodology

Data Collection:

Satellite RS Data: Obtain Landsat 8 satellite imagery from reliable sources, ensuring it covers the Madesh province of Nepal. **VGI In-situ Data:** Gather volunteered geographic information from local communities, farmers, agricultural organizations, or crowdsourced platforms. This data can include crop type labels, field boundaries, and other relevant information. For this project the collected and process data has been provided.

Visualizing In-situ Data in leaflet



Data Preprocessing: VGI In-situ Data: The in-situ data has been pre-processed by removing unnecessary columns and omitting NaNs. We also combined some data frames as per requirement.

Integration of Satellite and In-situ Data: Spatially overlay the in-situ data onto the satellite imagery. This integration links the ground truth information with corresponding satellite pixels. Then merge crop type from in-situ data with corresponding satellite pixels based on spatial location.

Feature Extraction: Red band and NIR band from the Landsat 8 image is used to calculate NDVI. NDVI helps quantify vegetation health and is crucial for crop classification.

Data Splitting: Split the integrated dataset into 70% training and 30% testing subsets. The training set will be used to train the machine learning model, while the testing set will be used to evaluate its performance.

Machine Learning Model Development: The splitted data is then used to train the Random Forest model, utilizing features NDVI extracted parameters as input, and crop types as the target variable.

Model Evaluation and Optimization: Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score on the testing subset.

Results

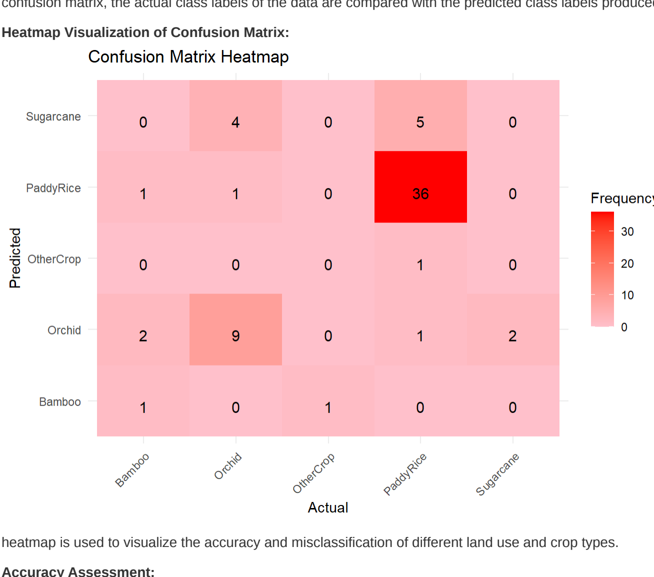
Classification Results:

The primary objective of this study was to classify crop types using the integrated remote sensing and in-situ data. The Random Forest classification model used in this study achieved the following results:

Overall Accuracy: The overall accuracy of the classification model was approximately 71.87%.

Confusion Matrix: Confusion Matrix is particularly useful for tasks where the model's output can be categorized into two or more classes. In a confusion matrix, the actual class labels of the data are compared with the predicted class labels produced by the classification algorithm.

Heatmap Visualization of Confusion Matrix:



heatmap is used to visualize the accuracy and misclassification of different land use and crop types.

Accuracy Assessment:

Precision: Precision measures the accuracy of the positive predictions made by the model. It calculates the ratio of true positive (TP) predictions to the total number of positive predictions made by the model (including both true positives and false positives).

Precision = True Positives (TP)/True Positives (TP) + False Positives (FP)

Precision provides insight into how many of the predicted positive instances were actually positive. A higher precision indicates fewer false positives, which means the model is better at not misclassifying negative instances as positive.

Recall: Recall measures the ability of the model to correctly identify all relevant positive instances. It calculates the ratio of true positive (TP) predictions to the total number of actual positive instances in the dataset (including both true positives and false negatives).

Recall = True Positives (TP)/True Positives (TP) + False Negatives (FN)

Recall provides insight into how many of the actual positive instances were correctly predicted by the model. A higher recall indicates that the model is good at capturing positive instances, minimizing false negatives.

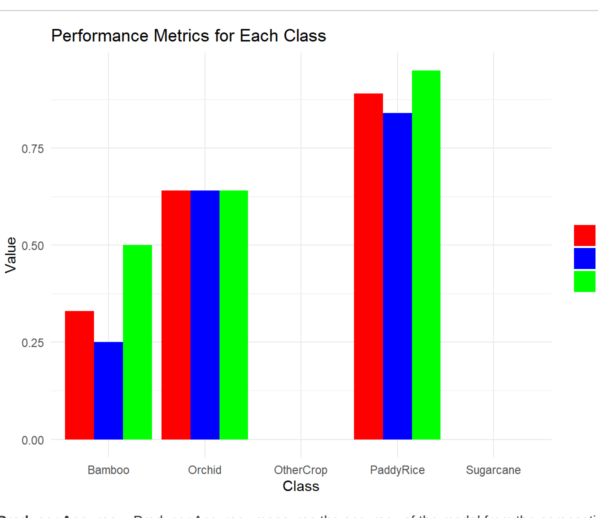
F1-Score: The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, taking both false positives and false negatives into account. F1 score is especially useful when the class distribution is imbalanced because it gives equal weight to precision and recall.

F1 Score = 2×(Precision×Recall)/Precision+Recall

F1 score ranges from 0 to 1, where 1 indicates a perfect balance between precision and recall. A higher F1 score suggests a better trade-off between precision and recall.

Bar Chart For Precision, recall and F1 Score

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```



Producer Accuracy: Producer Accuracy measures the accuracy of the model from the perspective of the data it was trained on. Specifically, it calculates the proportion of correctly classified pixels (or areas) for a specific land cover class relative to the total number of reference pixels (or areas) of that class in the real world.

Producer Accuracy (PA) = (Number of Correctly Classified class/Total Number of Reference Class)×100%

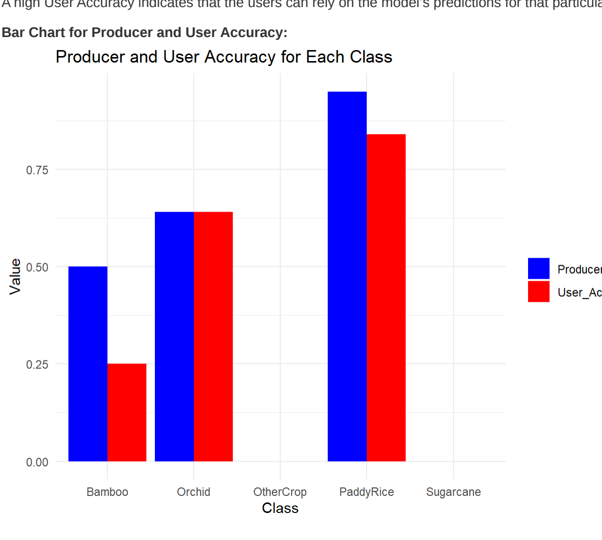
A high Producer Accuracy indicates that the classifier is effective in identifying the class it was trained to recognize.

User Accuracy: User Accuracy measures the accuracy of the model from the perspective of the user of the classification results. It calculates the proportion of correctly classified pixels (or areas) for a specific land cover class relative to the total number of pixels (or areas) classified as that class by the model.

User Accuracy (UA) = Number of Correctly Classified Class/Total Number Classified as that Class×100%

A high User Accuracy indicates that the users can rely on the model's predictions for that particular class.

Bar Chart for Producer and User Accuracy:



Discussion

The overall accuracy in this model is 71.87% means the performance in good even though some classes are hard to distinguish from other. Precision, Recall and F1 Score of PaddyRice is high among all, while most of Sugercane is classified into PaddyRice and Orchid. The data collection is not randomly distributed in whole part of Madhesh Pradesh. It is concentrated only in few districts while many districts have only few data collection points. Upon providing high accuracy data the model seems to perform good in future. Also instead of using only one Vegetation Index multiple Vegetaion Index would do better prediction for different crop type.

Conclusion

The identification and mapping of crops and their characteristics using remote sensing data has been widely used in developed countries in recent years. Crop mapping plays a vital role in enhancing agricultural productivity, optimizing resource utilization, supporting sustainable practices, and ensuring food security, making it an invaluable tool for agricultural and environmental management.This study is only focused on describing the methodology, with the help of finely collected data and advanced machine learning algorithms, precise classification of any part of World can be done in future.

References

Volunteered In-Situ Data for Agriculture Crop Mapping: A Case Study of Nepal By U. S. Panday , A. K. Pratihast , J. Aryal , R. Kayastha

R Code Chunks

```r
#loading libraries
library(sf)
library(leaflet)
library(rgdal)
library(raster)
library(terra)
library(dplyr)
library(ggplot2)
library(rgeos)
library(viridis)
library(rasterVis)
library(randomForest)
library(knitr)
library(tidyr)

#setting working directory

setwd("D:\\Geoinformatics\\Scientific Geo-Computing\\ProjectWork")

#loading in-situ data
in_situ_data <- st_read("LULC_Crop-Types_In-SituData2021_USP/LULC_Crop-Types_In-SituData2021_USP.shp")
```

```
## Reading layer `LULC_Crop-Types_In-SituData2021_USP' from data source
##   `D:\Geoinformatics\Scientific Geo-Computing\ProjectWork\LULC_Crop-Types_In-SituData2021_USP\LULC_Crop-Types_In-SituData2021_USP.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 603 features and 14 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 84.96197 ymin: 26.44534 xmax: 86.94405 ymax: 27.19818
## Geodetic CRS:  WGS 84
```

```r
#removing unwanted columns from in-situ data
in_situ_data <- in_situ_data %>%
  select(-data_Other, -data_Photo, -data_UserN, -data_Phone, -data_Subsc, -data_Email, -data_EndTi, -data_meta_, -F17, -F18)

#filtering data_LULC 'agriculture' row
in_situ_filtered <- in_situ_data %>% filter(data_LULC == 'Agriculture')

head(in_situ_filtered)
```

```
## Simple feature collection with 6 features and 4 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 85.70018 ymin: 26.96397 xmax: 85.70624 ymax: 26.96834
## Geodetic CRS:  WGS 84
##   data_Start   data_LULC data_CropT ClassType                  geometry
## 1 2021-10-24 Agriculture  Sugarcane        SC POINT (85.70624 26.96397)
## 2 2021-10-24 Agriculture  PaddyRice        PD POINT (85.70592 26.96418)
## 3 2021-10-24 Agriculture  Sugarcane        SC POINT (85.70403 26.96514)
## 4 2021-10-24 Agriculture  PaddyRice        PD POINT (85.70291 26.96515)
## 5 2021-10-24 Agriculture  PaddyRice        PD POINT (85.70018 26.96597)
## 6 2021-10-24 Agriculture  Sugarcane        SC POINT (85.70308 26.96834)
```

```r
# Load the district shapefile
district_shp <- st_read("districts/districts.shp")
```

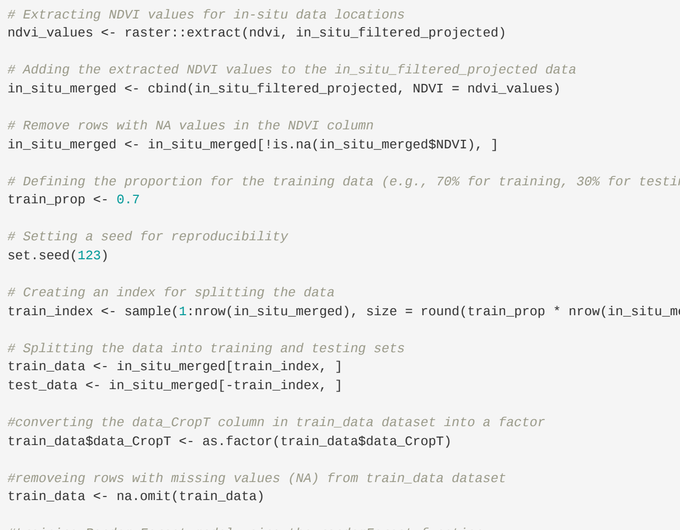
```
## Reading layer `districts' from data source
##   `D:\Geoinformatics\Scientific Geo-Computing\ProjectWork\districts\districts.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 77 features and 4 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 80.05847 ymin: 26.34776 xmax: 88.20155 ymax: 30.47297
## Geodetic CRS:  WGS 84
```

```r
# Define the target districts
target_districts <- c("SAPTARI", "SIRAHA", "DHANUSHA","MAHOTTARI", "SARLAHI", "RAUTAHAT","BARA", "PARSA")

# Filter the shapefile to include only the target districts
madhesh_districts <- district_shp[district_shp$DISTRICT %in% target_districts, ]


# Defining a custom color palette for the specific class types
color_palette <- colorFactor(
  palette = c("green", "yellow", "purple", "gold4", "blue"),  # Define colors for each class type
  domain = c("Sugarcane", "PaddyRice", "Orchid","Bamboo", "OtherCrop")  # Define class types
)

# Creating a leaflet map
leaflet() %>%
  addProviderTiles("Esri.WorldImagery") %>%
  addPolygons(data = madhesh_districts, color = "red", weight = 1, opacity = 1, fillOpacity = 0.2) %>%
  addCircleMarkers(data = in_situ_filtered,
                   lng = ~st_coordinates(geometry)[, 1],
                   lat = ~st_coordinates(geometry)[, 2],
                   popup = ~data_CropT,
                   color = ~color_palette(data_CropT),
                   radius = 1)%>%
  setView(lng = 85.793, lat = 26.953, zoom = 8.0)
```

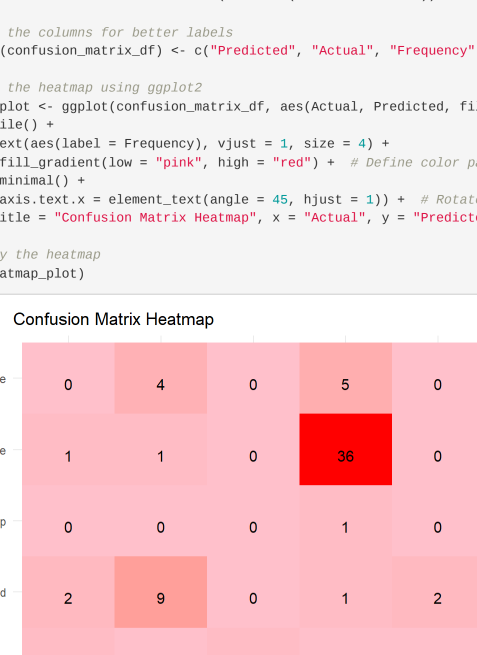



```r
#importing landsat 8 image data
red_band <- raster("LC08_L2SP_140041_20211219_20211223_02_T1_SR_B4.tif")
nir_band <- raster("LC08_L2SP_140041_20211219_20211223_02_T1_SR_B5.tif")

# Calculation NDVI
ndvi_funtion <- function(red_band, nir_band) {
  ndvi <- (nir_band - red_band) / (nir_band + red_band)
  return(ndvi)
}

ndvi <- ndvi_funtion(red_band , nir_band)

#plotting NDVI
gplot(ndvi) +
  geom_raster(aes(x = x, y = y, fill = value)) +
  coord_quickmap() +
  ggtitle("NDVI") +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=15),
        axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
#creating target coordinate reference system
crs_target <- st_crs("+init=EPSG:32645")

#projecting coordinate systerm
in_situ_filtered_projected <- st_transform(in_situ_filtered, crs_target)

# Extracting NDVI values for in-situ data locations
ndvi_values <- raster::extract(ndvi, in_situ_filtered_projected)

# Adding the extracted NDVI values to the in_situ_filtered_projected data
in_situ_merged <- cbind(in_situ_filtered_projected, NDVI = ndvi_values)

# Remove rows with NA values in the NDVI column
in_situ_merged <- in_situ_merged[!is.na(in_situ_merged$NDVI), ]

# Defining the proportion for the training data (e.g., 70% for training, 30% for testing)
train_prop <- 0.7

# Setting a seed for reproducibility
set.seed(123)

# Creating an index for splitting the data
train_index <- sample(1:nrow(in_situ_merged), size = round(train_prop * nrow(in_situ_merged)))

# Splitting the data into training and testing sets
train_data <- in_situ_merged[train_index, ]
test_data <- in_situ_merged[-train_index, ]

#converting the data_CropT column in train_data dataset into a factor
train_data$data_CropT <- as.factor(train_data$data_CropT)

#removeing rows with missing values (NA) from train_data dataset
train_data <- na.omit(train_data)

#training Random Forest model using the randomForest function
rf_model <- randomForest(data_CropT ~ NDVI, data = train_data, ntree = 500)

# Making predictions on the test data
rf_predictions <- predict(rf_model, test_data)

confusion_matrix <- table(Actual = test_data$data_CropT, Predicted = rf_predictions)
confusion_matrix
```

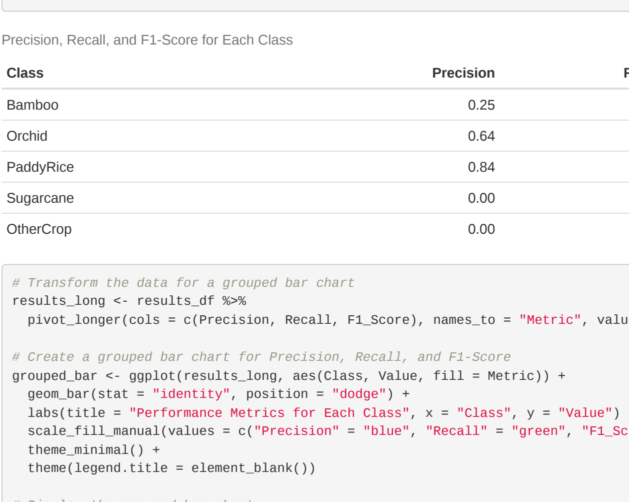
```
##             Predicted
## Actual       Bamboo Orchid OtherCrop PaddyRice Sugarcane
##   Bamboo          1      0         1         0         0
##   Orchid          2      9         0         1         2
##   OtherCrop       0      0         0         1         0
##   PaddyRice       1      1         0        36         0
##   Sugarcane       0      4         0         5         0
```

```r
# Convert the confusion matrix to a data frame
confusion_matrix_df <- as.data.frame(as.table(confusion_matrix))

# Rename the columns for better labels
colnames(confusion_matrix_df) <- c("Predicted", "Actual", "Frequency")

# Create the heatmap using ggplot2
heatmap_plot <- ggplot(confusion_matrix_df, aes(Actual, Predicted, fill = Frequency)) +
  geom_tile() +
  geom_text(aes(label = Frequency), vjust = 1, size = 4) +
  scale_fill_gradient(low = "pink", high = "red") +  # Define color palette
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +  # Rotate x-axis labels
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")

# Display the heatmap
print(heatmap_plot)
```




```r
# Define class names
class_names <- c("Bamboo", "Orchid", "PaddyRice", "Sugarcane", "OtherCrop")

# Initialize vectors to store precision, recall, and F1 values
precision_values <- numeric(length(class_names))
recall_values <- numeric(length(class_names))
f1_values <- numeric(length(class_names))


for (i in 1:length(class_names)) {
  class_name <- class_names[i]
  TP <- confusion_matrix[class_name, class_name]
  FP <- sum(confusion_matrix[, class_name]) - TP
  FN <- sum(confusion_matrix[class_name, ]) - TP
  precision <- TP / (TP + FP)
  recall <- TP / (TP + FN)
  f1_value <- 2 * (precision * recall) / (precision + recall)



  # Store values in respective vectors
  precision_values[i] <- precision
  recall_values[i] <- recall
  f1_values[i] <- f1_value
}

# Create a data frame with  results
results_df <- data.frame(
  Class = class_names,
  Precision = round(precision_values, 2),
  Recall = round(recall_values, 2),
  F1_Score = round(f1_values, 2)
)

# Use kable to format the results as a table
kable(results_df, caption = "Precision, Recall, and F1-Score for Each Class")
```

Precision, Recall, and F1-Score for Each Class

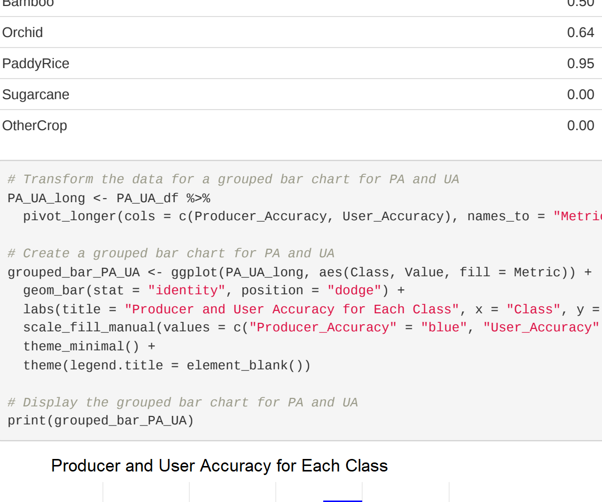
Class	Precision	Recall	F1_Score
Bamboo	0.25	0.50	0.33
Orchid	0.64	0.64	0.64
PaddyRice	0.84	0.95	0.89
Sugarcane	0.00	0.00	NaN
OtherCrop	0.00	0.00	NaN

```r
# Transform the data for a grouped bar chart
results_long <- results_df %>%
  pivot_longer(cols = c(Precision, Recall, F1_Score), names_to = "Metric", values_to = "Value")

# Create a grouped bar chart for Precision, Recall, and F1-Score
grouped_bar <- ggplot(results_long, aes(Class, Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Performance Metrics for Each Class", x = "Class", y = "Value") +
  scale_fill_manual(values = c("Precision" = "blue", "Recall" = "green", "F1_Score" = "red")) +
  theme_minimal() +
  theme(legend.title = element_blank())

# Display the grouped bar chart
print(grouped_bar)
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```




```r
#initializing matrices to store producer accuracy (PA) and user accuracy (UA)
PA_scores <- matrix(0, nrow = length(class_names), ncol = 1)
UA_scores <- matrix(0, nrow = length(class_names), ncol = 1)

#calculating producer accuracy and user accuracy for each class
for (i in 1:length(class_names)){
  class_name <- class_names[i]
  PA <- confusion_matrix[class_name, class_name] / sum(confusion_matrix[class_name, ])
  PA_scores[i] <- PA
  UA <- confusion_matrix[class_name, class_name] / sum(confusion_matrix[, class_name])
  UA_scores[i] <- UA
}

# Create a data frame for  Producer Accuracy and User Accuracy results
PA_UA_df <- data.frame(
  Class = class_names,
  Producer_Accuracy = round(PA_scores, 2),
  User_Accuracy = round(UA_scores, 2))

# Use kable to format the results as a table
kable(PA_UA_df, caption = "Producer and User Accuracy for Each Class")
```

Producer and User Accuracy for Each Class

Class	Producer_Accuracy	User_Accuracy
Bamboo	0.50	0.25
Orchid	0.64	0.64
PaddyRice	0.95	0.84
Sugarcane	0.00	0.00
OtherCrop	0.00	0.00

```r
# Transform the data for a grouped bar chart for PA and UA
PA_UA_long <- PA_UA_df %>%
  pivot_longer(cols = c(Producer_Accuracy, User_Accuracy), names_to = "Metric", values_to = "Value")

# Create a grouped bar chart for PA and UA
grouped_bar_PA_UA <- ggplot(PA_UA_long, aes(Class, Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Producer and User Accuracy for Each Class", x = "Class", y = "Value") +
  scale_fill_manual(values = c("Producer_Accuracy" = "blue", "User_Accuracy" = "red")) +
  theme_minimal() +
  theme(legend.title = element_blank())

# Display the grouped bar chart for PA and UA
print(grouped_bar_PA_UA)
```