Project Report Format

# INTRODUCTION

* 1. Project Overview

The **Rice Type Classification** project aims to classify different types of rice grains using deep learning and computer vision. It uses a Convolutional Neural Network (CNN) based on the MobileNetV2 architecture to predict rice types from uploaded images. The goal is to automate rice type identification with high accuracy, helping in agriculture and food industries.

* 1. Purpose

The purpose of this project is to:

* Enable accurate identification of rice types from grain images.
* Assist farmers, distributors, and consumers in differentiating rice varieties.
* Provide a user-friendly web interface for prediction using Streamlit.

# IDEATION PHASE

* 1. Problem Statement

Identifying rice types manually is time-consuming and prone to human error. There is a need for an automated system that can accurately classify rice grain types using image data.

* 1. Empathy Map Canvas

 **Says**: "I need help identifying the rice type."

 **Thinks**: "I hope the system is accurate and easy to use."

 **Feels**: Anxious about misclassification.

 **Does**: Uploads rice images to get predictions from the model.

* 1. Brainstorming

Ideas considered:

* Manual dataset curation and augmentation.
* Use of MobileNetV2 for fast and efficient classification.
* Deploying the model using Streamlit for accessibility.
* Evaluating with different metrics like accuracy and confidence score.

# REQUIREMENT ANALYSIS

* 1. Customer Journey map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stage | Action | |  | | --- | |  |  |  | | --- | | Experience | |
| Awareness | Learns about the tool | Curious |
| Engagement | Uploads rice image | Interested |
| Usage | Receives prediction | Satisfied |
| Feedback | Shares usability feedback | Encouraged to improve |

* 1. Solution Requirement

 Python (TensorFlow, Keras, NumPy, PIL)

 Streamlit for frontend

 .h5 trained CNN model

 Rice image dataset categorized by type

* 1. Data Flow Diagram

[User Uploads Image]

↓

[Image Preprocessed]

↓

[Model Predicts Class]

↓

[Result Displayed in UI]

* 1. Technology Stack

 **Frontend**: Streamlit

 **Backend**: TensorFlow (Model Inference)

 **Language**: Python

 **Libraries**: Keras, PIL, NumPy

# PROJECT DESIGN

* 1. Problem Solution Fit

 The project directly addresses the challenge of visually identifying rice types.

 Solution is low-cost, automated, and accessible through a web interface.

* 1. Proposed Solution

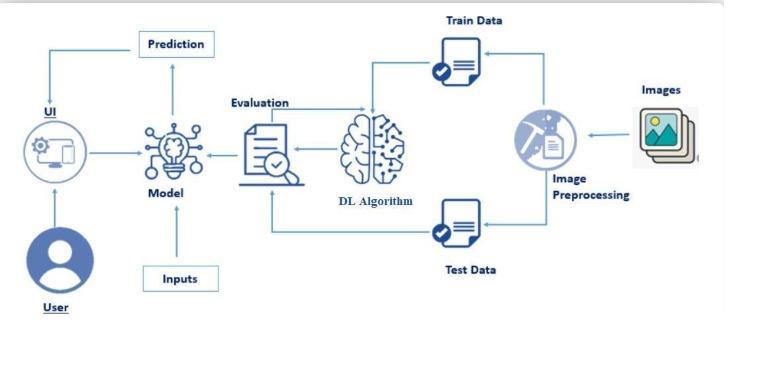
 Train a CNN model on categorized rice images.

 Integrate the model into a Streamlit app.

 Allow users to upload images and receive predictions.

* 1. Solution Architecture

User → Streamlit UI → Image → Preprocessing → CNN Model (MobileNetV2) → Prediction → Output



# PROJECT PLANNING & SCHEDULING

* 1. Project Planning

|  |  |
| --- | --- |
| Week | Task |
| 1 | Data collection, dataset exploration, and cleaning |
| 2 | |  | | --- | |  |  |  | | --- | | Data preprocessing and augmentation | |
| 3 | |  | | --- | |  |  |  | | --- | | Model selection (MobileNetV2), model training | |
| 4 | |  | | --- | |  |  |  | | --- | | Evaluation, optimization, and finalizing model | |
| 5 | Streamlit UI development and integration with the trained model |
| 6 | |  | | --- | |  |  |  | | --- | | Testing, bug fixing, documentation, and final submission | |

# FUNCTIONAL AND PERFORMANCE TESTING

* 1. Performance Testing

To evaluate the performance of our rice classification model, we performed rigorous testing using a validation dataset consisting of 15,000 images (3,000 per rice class). The following metrics were used to measure model effectiveness:

#### Final Accuracy Scores

* **Training Accuracy:** 97.12%
* **Validation Accuracy:** 97.99%
* **Overall Test Accuracy:** **98.05%**

#### Confusion Matrix

The confusion matrix illustrates that the model correctly classified the majority of samples across all five rice varieties with very few misclassifications.

Classification Report

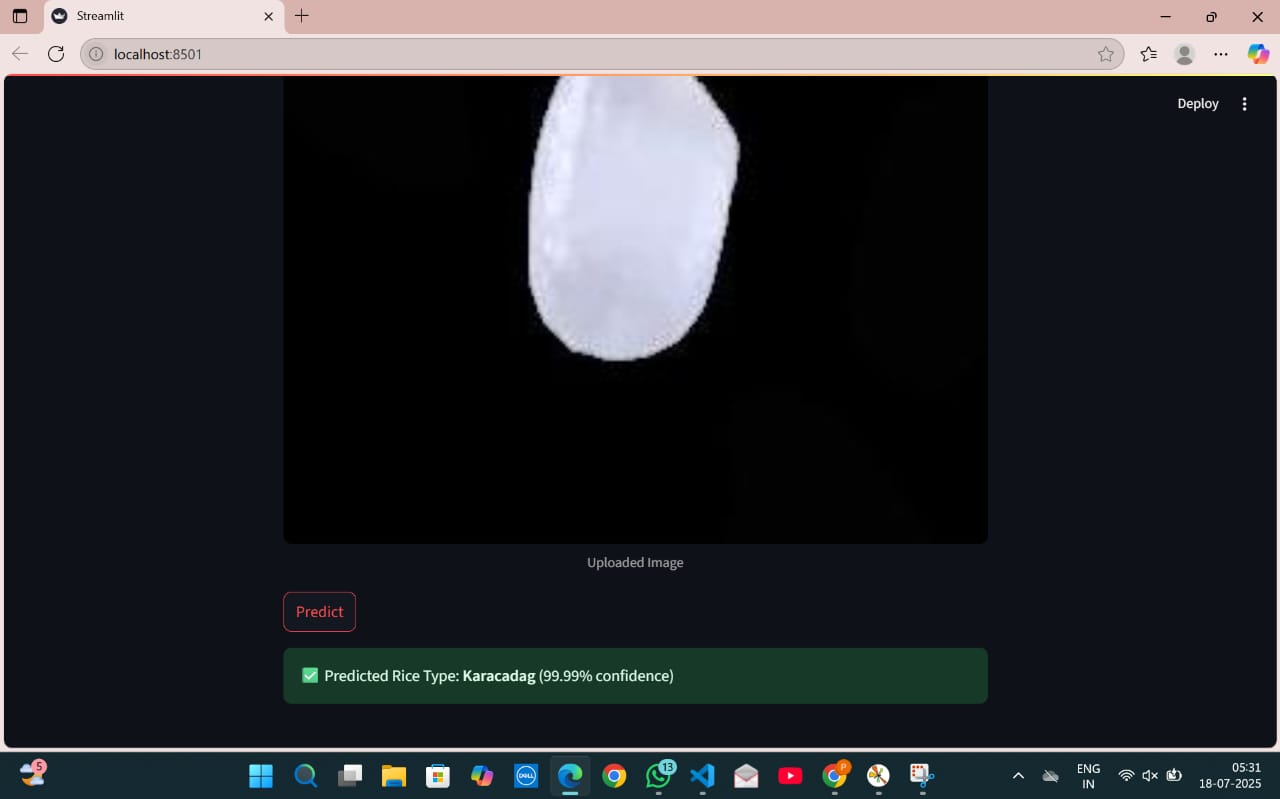
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| Arborio | 0.97 | 0.97 | 0.97 | 3000 |
| Basmati | 0.98 | 0.99 | 0.98 | 3000 |
| Ipsala | 0.99 | 1.00 | 0.99 | 3000 |
| Jasmine | 0.98 | 0.97 | 0.98 | 3000 |
| Karacadag | 0.98 | 0.97 | 0.98 | 3000 |
| Overall | 0.98 | 0.98 | 0.98 | 15000 |

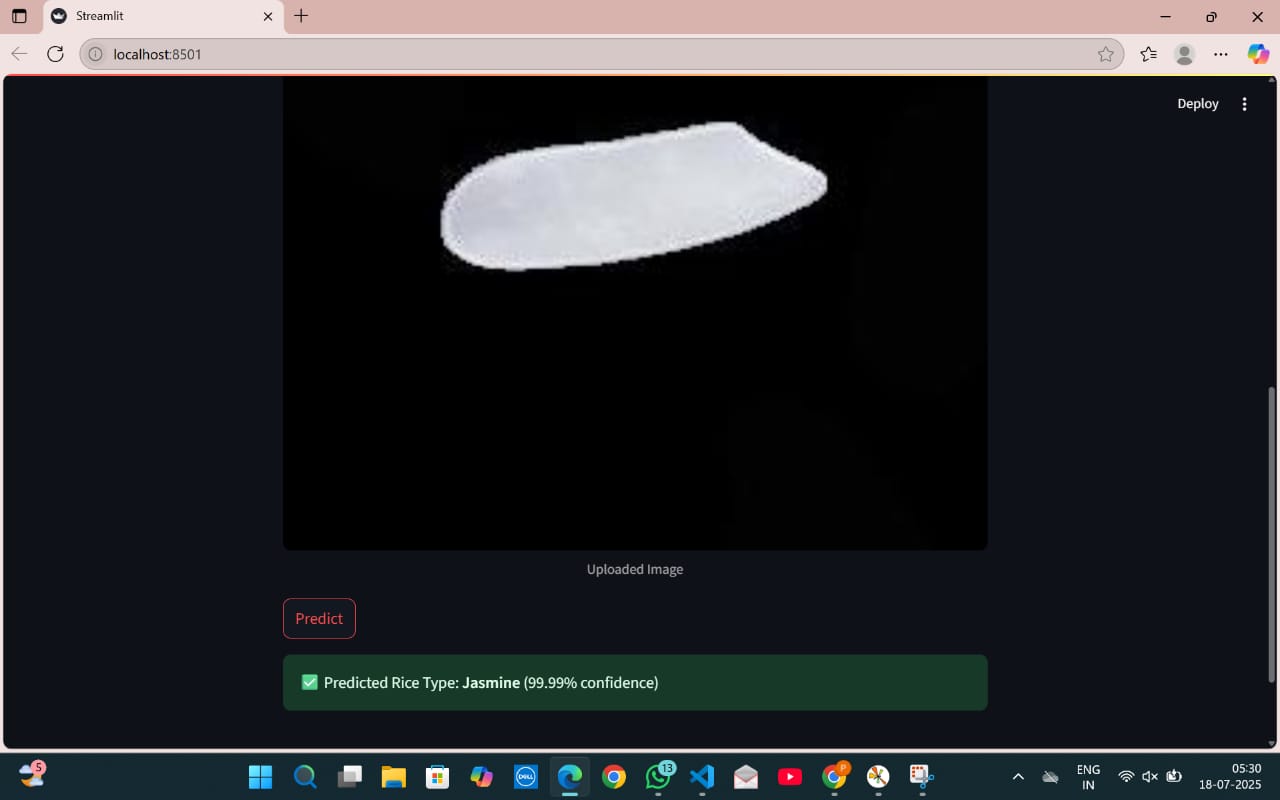
#### Interpretation:

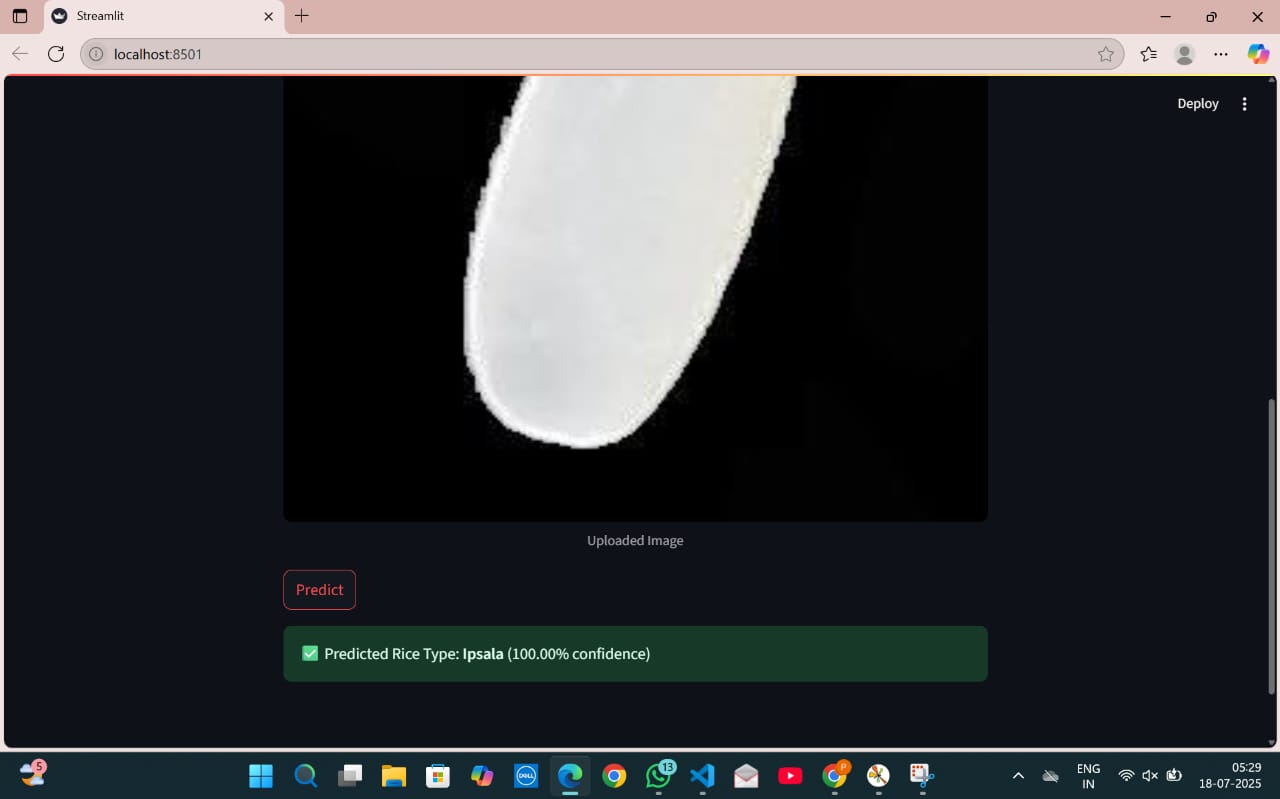
* All classes show **precision, recall, and F1-scores of 0.97 or above**, indicating robust performance with minimal false positives and false negatives.
* Ipsala rice achieved perfect recall, meaning it was never misclassified in the test set.
* The macro and weighted averages reflect consistent performance across all classes, confirming the model's reliability.

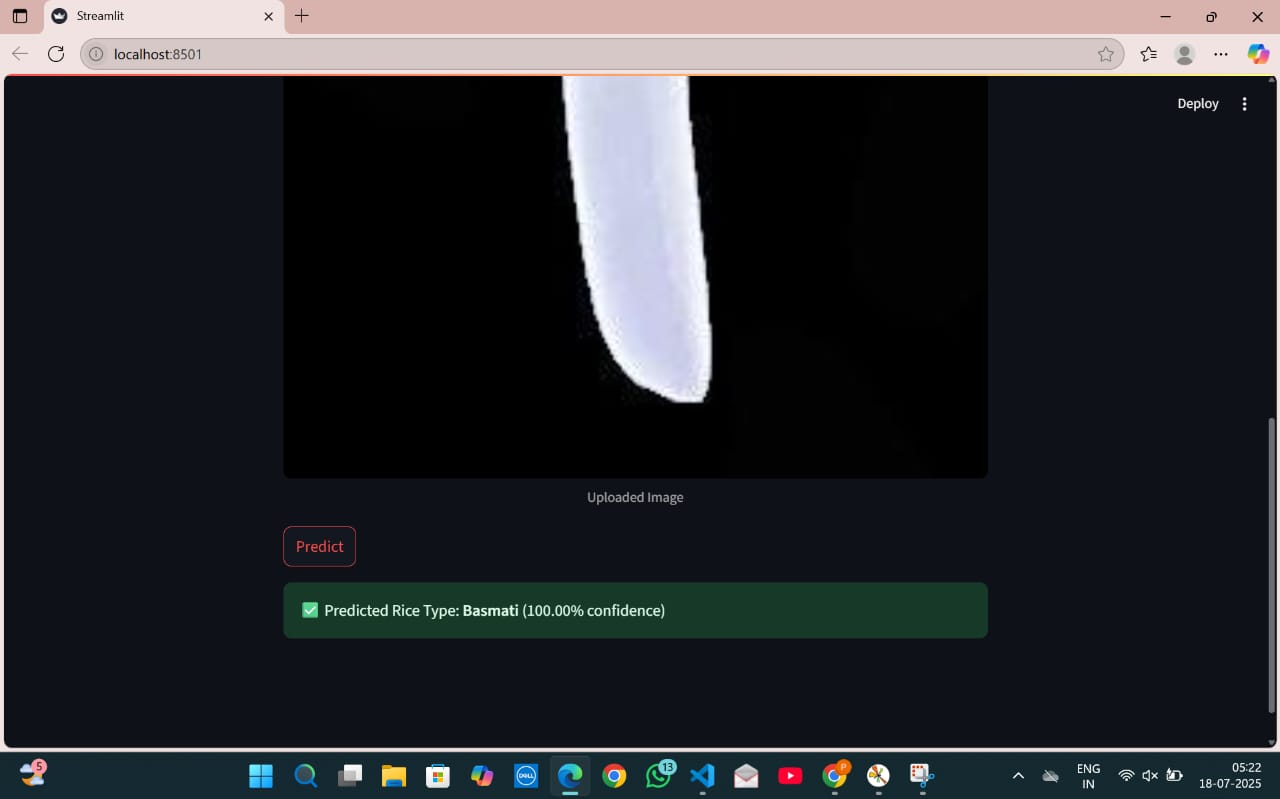
# RESULTS

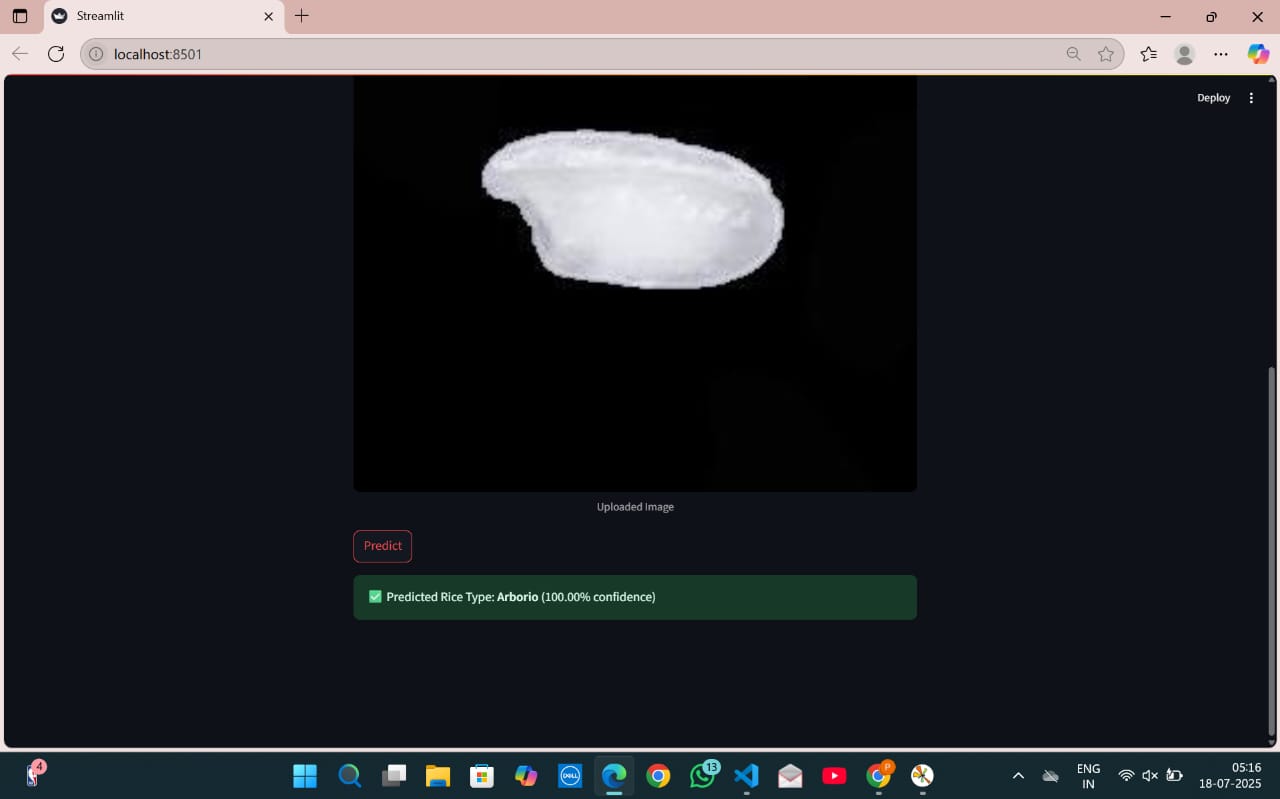
* 1. Output Screenshots

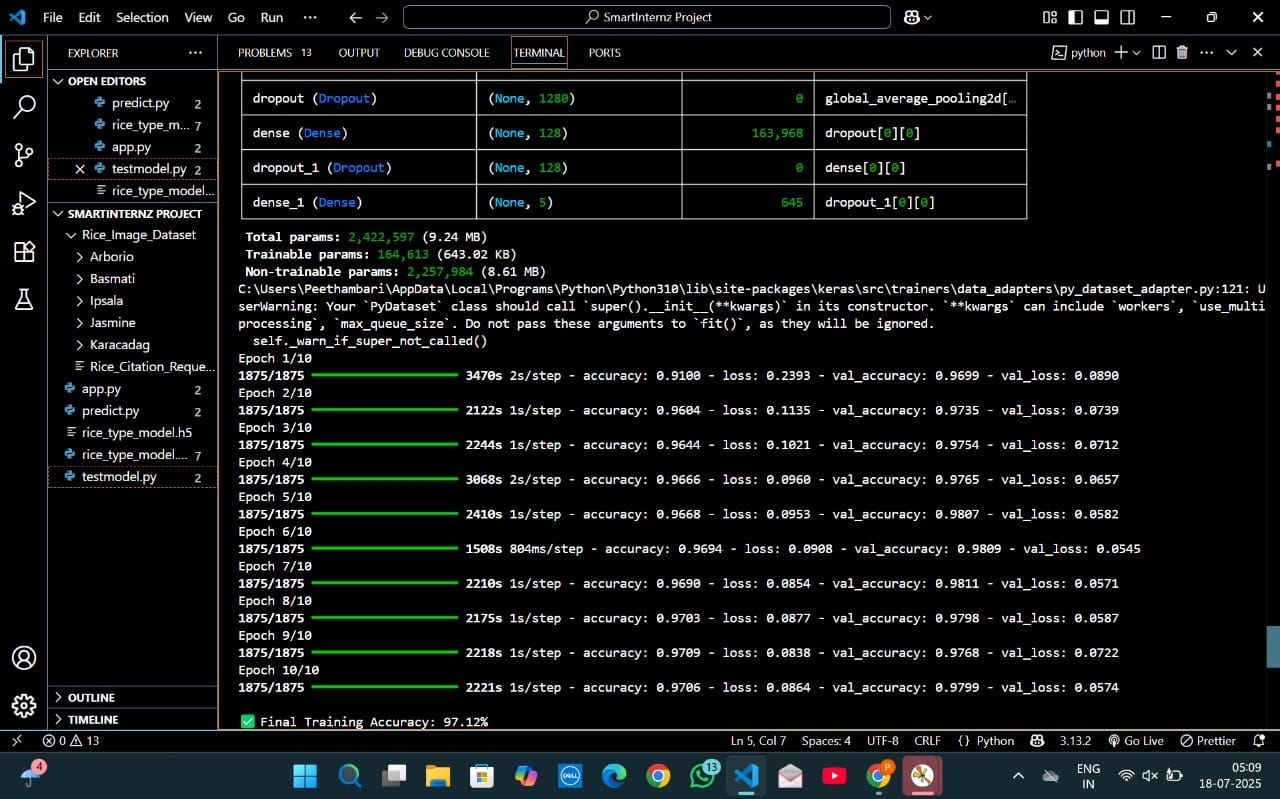












1. **ADVANTAGES & DISADVANTAGES**

### ****Advantages****

* Fast and accurate rice classification.
* User-friendly interface.
* Lightweight model (MobileNetV2).
* Easy deployment using Streamlit.

### ****Disadvantages****

* Model accuracy may drop on poor-quality or highly similar images.
* Requires internet for app usage if deployed on a remote server.
* Limited to the dataset types (5 classes only).

1. **CONCLUSION**

This project successfully demonstrates an AI-powered system for rice type classification. By leveraging deep learning, specifically MobileNetV2, and combining it with a simple frontend using Streamlit, users can easily classify rice grain images. The tool is efficient, reliable, and has real-world applications in the agricultural and food sectors.

1. **FUTURE SCOPE**

* Expand to classify more rice varieties or grains like wheat and barley.
* Include a mobile version of the app.
* Improve performance with more complex models (e.g., EfficientNet).
* Add multilingual support to enhance accessibility.

1. **APPENDIX**

Source Code

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

import matplotlib.pyplot as plt

import os

import numpy as np

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

# Paths

data\_dir = "Rice\_Image\_Dataset"

image\_size = 224

batch\_size = 32

# Data Generator

datagen = ImageDataGenerator(

rescale=1./255,

validation\_split=0.2,

rotation\_range=20,

zoom\_range=0.2,

horizontal\_flip=True

)

train\_data = datagen.flow\_from\_directory(

data\_dir,

target\_size=(image\_size, image\_size),

batch\_size=batch\_size,

class\_mode='categorical',

subset='training',

shuffle=True

)

val\_data = datagen.flow\_from\_directory(

data\_dir,

target\_size=(image\_size, image\_size),

batch\_size=batch\_size,

class\_mode='categorical',

subset='validation',

shuffle=False # Important: keep this False for correct label alignment

)

# Base Model

base\_model = MobileNetV2(input\_shape=(image\_size, image\_size, 3),

include\_top=False,

weights='imagenet')

base\_model.trainable = False

# Custom Head

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dropout(0.3)(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.3)(x)

predictions = Dense(5, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

# Compile

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Summary of Model Structure

model.summary()

# Train

history = model.fit(

train\_data,

epochs=10,

validation\_data=val\_data

)

train\_acc = history.history['accuracy'][-1]

val\_acc = history.history['val\_accuracy'][-1]

print(f"\n✅ Final Training Accuracy: {train\_acc \* 100:.2f}%")

print(f"✅ Final Validation Accuracy: {val\_acc \* 100:.2f}%\n")

model.save("rice\_type\_model.h5")

# Reset validation generator

val\_data.reset()

# True Labels

y\_true = val\_data.classes

class\_labels = list(val\_data.class\_indices.keys())

# Predictions

y\_pred\_probs = model.predict(val\_data)

y\_pred = np.argmax(y\_pred\_probs, axis=1)

# Confusion Matrix

cm = confusion\_matrix(y\_true, y\_pred)

print("\n✅ Confusion Matrix:")

print(cm)

# Accuracy

acc = accuracy\_score(y\_true, y\_pred)

print(f"\n✅ Accuracy Score: {acc \* 100:.2f}%")

# Classification Report

print("\n✅ Classification Report:")

print(classification\_report(y\_true, y\_pred, target\_names=class\_labels))

# Plot Accuracy

plt.plot(history.history['accuracy'], label='Train Acc')

plt.plot(history.history['val\_accuracy'], label='Val Acc')

plt.legend()

plt.title("Accuracy")

plt.figure()

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Val Loss')

plt.legend()

plt.title("Loss")

plt.show()

Dataset Link : <https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset>

GitHub & Project Demo Link :

<https://github.com/ValliGayathri/GrainPalette---A-Deep-Learning-Odyssey-In-Rice-Type-Classification-Through-Transfer-Learning>