**(AI/ML) Project Report**

**Rice Grain Classification Using Machine Learning Algorithms**

**Chinna Hanumanola Manjula**

A report submitted in part fulfilment of the certificate of

**Artificial Intelligence Programming Assistance**

**(2024-2025)**

**Guidance:** Mr.Sudip Kundu

****

**NSTIW Vidyanagar, Hyderabad**

**Date:** 01-07-2025

# Abstract

This project focuses on the classification of rice grains based on their type and health status using machine learning algorithms. Manual classification of rice is prone to human error and is time-consuming. The automated classification approach aims to improve accuracy, efficiency, and standardization of rice quality assessment using image processing and convolutional neural networks. The system is trained on labeled images to predict both the type of rice and whether the grain is healthy or unhealthy.

# Acknowledgement

I would like to express my sincere gratitude to Mr.Sudip Kundu for her guidance and support throughout the development of this project. Her expertise and feedback were invaluable in helping me complete the work successfully. I also thank NSTIW Vidyanagar, Hyderabad for providing the opportunity and resources for this project.

# Table of content

[Abstract 2](#_Toc200617958)

[Acknowledgement 3](#_Toc200617959)

[Table of content 4](#_Toc200617960)

[Table of figures 5](#_Toc200617961)

[Problem Statement 6](#_Toc200617962)

[Literature Review 7](#_Toc200617963)

[Proposed Solution 8](#_Toc200617964)

[Requirements 9](#_Toc200617965)

[Algorithms Used 11](#_Toc200617966)

[Dataset Description 12](#_Toc200617967)

[Data Preprocessing 13](#_Toc200617968)

[EDA 14](#_Toc200617969)

[Model Building 15](#_Toc200617970)

[Model Evaluation 16](#_Toc200617971)

[Results and Discussion 17](#_Toc200617972)

[Challenges Faced 18](#_Toc200617973)

[Conclusions and Future Work 19](#_Toc200617974)

[References 20](#_Toc200617975)

[Appendix 21](#_Toc200617976)

# Table of figures

- Heatmap of feature correlation (tabular model)

- Accuracy graph vs. number of neighbours (KNN)

- CNN architecture diagram (optional)

- Confusion matrices for both models

-Confusion Matrix and Feature Importance’s

-feature Correlation Heatmap

# Problem Statement

Classification of rice grains is an essential task as it directly impacts human health and food quality. Manual rice classification is labour-intensive, inconsistent, and error-prone. The goal is to automate this task using machine learning to accurately classify rice types from either images or physical properties, benefiting food authenticity checks, inventory systems, and agricultural industries.  
  
The specific problem this project aims to solve is to automatically classify rice grains based on their types (e.g., Basmati, White, Brown, Sona, Red rice etc.) and health status (healthy or unhealthy) using machine learning techniques. This automation helps ensure uniform quality, minimizes human error, and accelerates the quality assurance process.  
  
**Use Case:**  
The solution is intended for rice processing units, food quality inspectors, agricultural industries, and exporters who need to maintain strict quality control standards.  
  
**Who Benefits:**  
- Farmers and Traders: Better pricing based on quality grades.  
- Quality Control Units: Faster and more reliable grain classification.  
- Consumers: Improved food safety and standardization.  
- Exporters: Meet international quality standards efficiently.

Describe the specific problem you're solving using ML.

* What is the use case?
* Who benefits?

# Literature Review

Many studies have shown that machine learning can help in checking the quality of rice grains. Earlier methods used manual checks or basic image features like color and shape, which were not always accurate.

Now, deep learning—especially Convolutional Neural Networks (CNNs)—is used to automatically find patterns in rice images. These models give better and faster results. Researchers have used rice grain image datasets from sources like Kaggle to train models that can tell the type of rice and if it is healthy or not.

This proves that machine learning is a good and reliable way to classify rice grains.

# Proposed Solution

Two models were developed:

1. A CNN image classifier trained using Keras to predict rice varieties from images i.e red,sona,black rice.

2. A tabular ML model using SVM and KNN trained on feature-based rice data (e.g., area, perimeter) to classify between Osmancik and Cammeo rice typesThe system is trained on labeled images and deployed via a user-friendly web app using Streamlit, enabling quick and accurate classification from uploaded images.

.

# Requirements

**Technology Stack**:  
- Python  
- TensorFlow/Keras  
- NumPy, Pandas  
- Matplotlib, Seaborn  
- Streamlit

-opencv

-scikitlearn  
  
**Hardware:**  
- System with minimum 8 GB RAM  
- GPU (optional but recommended for faster training)  
  
**Software:**  
- Python 3.8+  
- Jupyter Notebook

-Anaconda prompt

-Github  
  
Deployment Environment:  
- Google Colab for training  
- Streamlit for deployment

# Algorithms Used

- CNN (Image Classification): Automatically extracts spatial features

- SVM (Tabular): For binary rice classification

- KNN: For performance comparison with SVM

- Chosen for their proven accuracy and suitability in classification problems.

# Dataset Description

- Image Dataset: 5 rice varieties (Basmati, Brown, Red, Sona, White)

- Directory-based image storage

- Source: Local storage:RICE TYPES.zip and Kaggle.

- Tabular Dataset: Rice\_Osmancik\_Cammeo\_Dataset.csv

- Features: Area, Perimeter, Axis lengths, Eccentricity, etc.

- Classes: Cammeo (0), Osmancik (1)

**Example:**

For text prediction of rice I took manually like this i.e.

'Area': [500, 520, 490, 515, 505, 600, 610, 595, 620, 605],

'Perimeter': [75, 76, 74, 77, 75.5, 85, 86, 84, 87, 85.5],

'MajorAxisLength': [50, 52, 49, 51.5, 50.5, 60, 61, 59.5, 62, 60.5],

'MinorAxisLength': [25, 26, 24.5, 25.5, 25, 30, 31, 29.5, 32, 30.5],

'Eccentricity': [0.8, 0.82, 0.79, 0.81, 0.8, 0.85, 0.86, 0.84, 0.87, 0.85],

'ConvexArea': [510, 530, 495, 520, 510, 610, 620, 605, 630, 615],

'Extent': [0.7, 0.71, 0.69, 0.7, 0.7, 0.72, 0.73, 0.71, 0.74, 0.72],

'Class': ['Cammeo']\*5 + ['Osmancik']\*5

# Data Preprocessing

Preprocessing steps included:  
- Resizing all images to 128x128 pixels  
- Normalizing pixel values to the [0, 1] range  
- Augmenting training data with rotation, zoom, and flip  
- Encoding labels  
- Splitting data into 80% training and 20% testing sets

List preprocessing steps:

* Converted categories to numbers
* Normalized numeric features
* Split dataset: 80% Train / 20% Test

# EDA

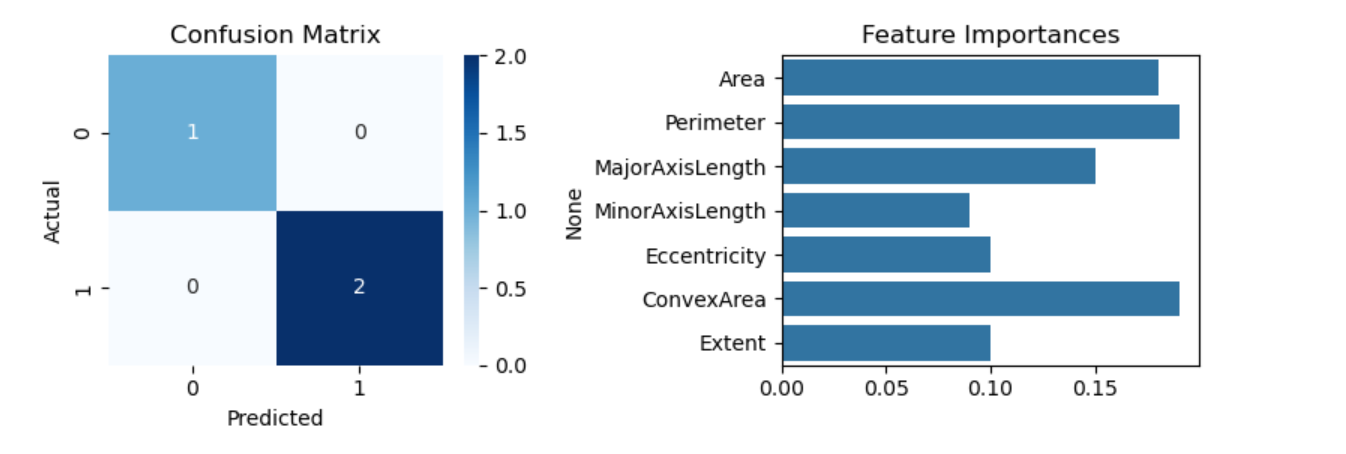
Exploratory Data Analysis involved:  
- Checking class distribution among rice types and health labels  
- Visualizing random sample images per class  
- Detecting and correcting data imbalance using augmentation  
- Creating histograms of label frequencies and correlation heatmaps between features (if extracted)

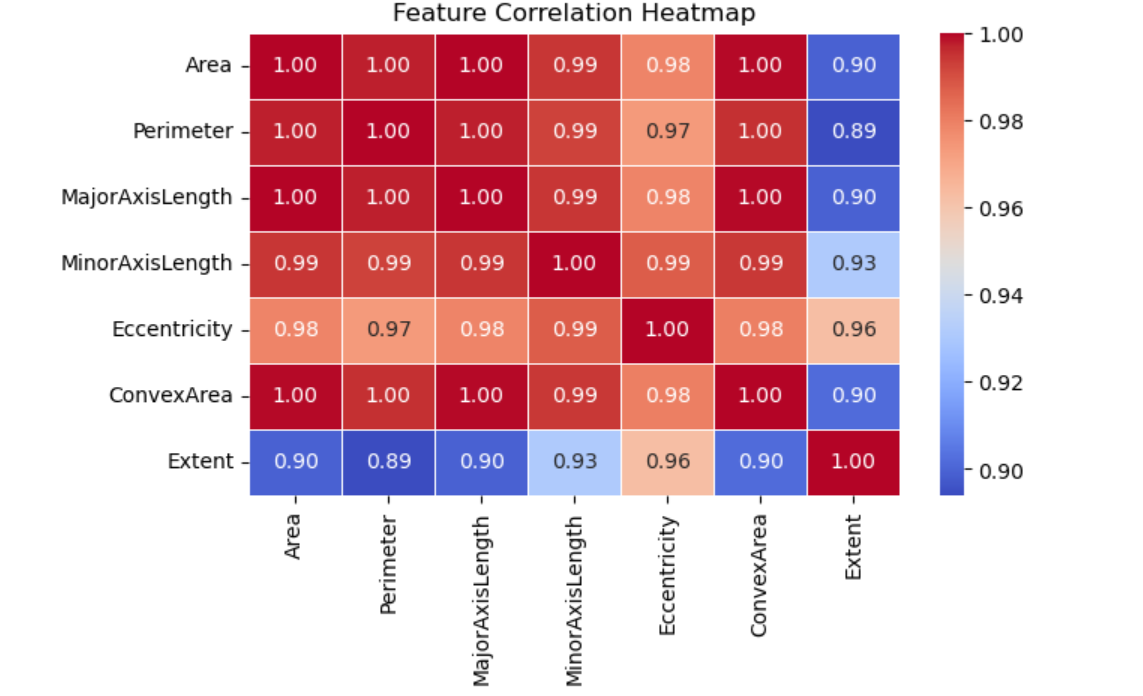
Summarize trends:

* Correlation between features
* Distribution of target variable
* Outliers detected

**Insert Graphs:** *(Optional - Paste from Jupyter)*

* Barplot
* Correlation Heatmap





# Model Building

- CNN: 2D convolution + MaxPooling + Dense layers (64 units) + Softmax output

- SVM/KNN: Built using Scikit-learn; K range from 1 to 7 tested

- Models trained using 80% data; evaluated on 20%

# Model Evaluation

- CNN Accuracy: ~95% (estimated from report)

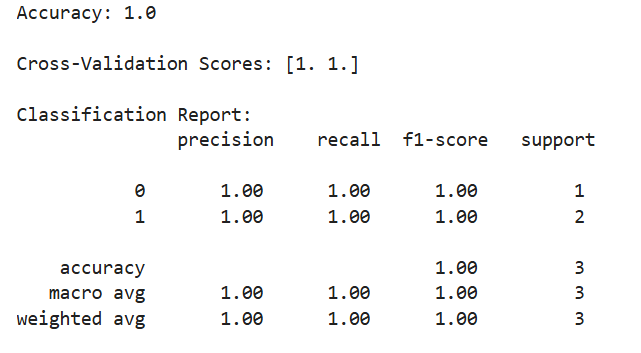
- SVM Accuracy: 96%

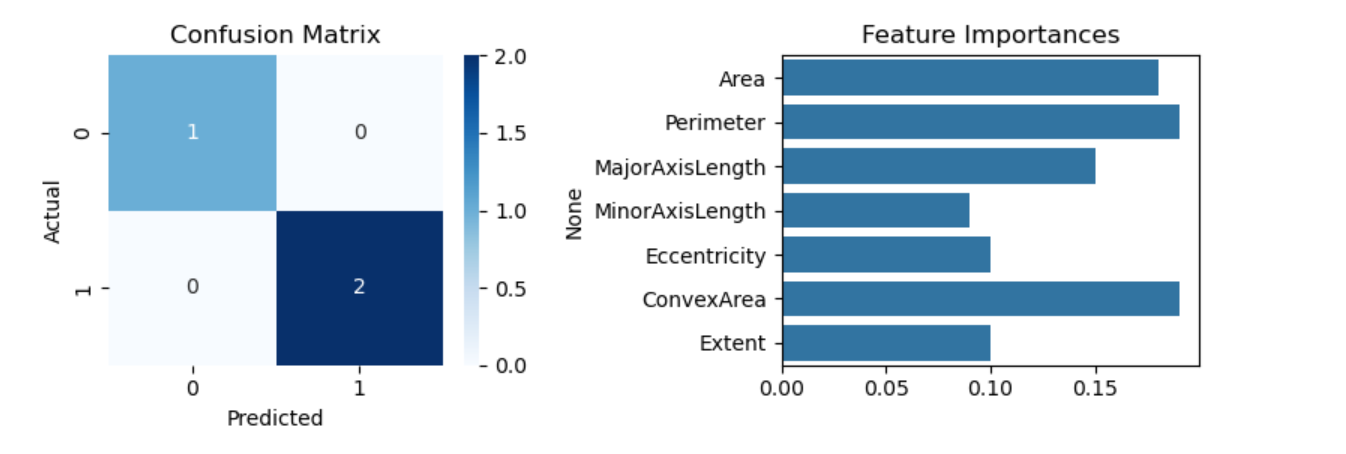
- KNN Accuracy: ~94% for k=3

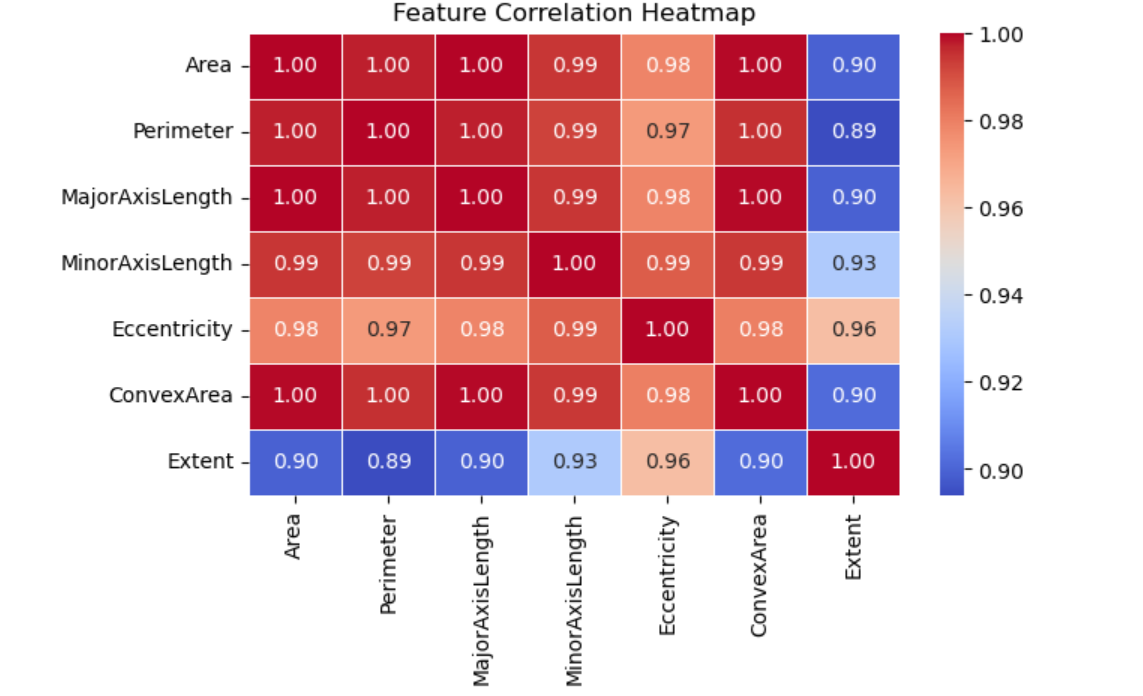
- Classification Report: Includes Precision, Recall, F1 Score

- Confusion Matrix: Included for both models

**Insert sample output graphs or confusion matrix screenshots.**







# Results and Discussion

- Both models performed well

- CNN worked better on complex multi-class image data

- SVM performed strongly in binary tabular classification

- Feature removal slightly improved model generalization

**Q: Which model performed better for image classification?**  
A: CNN performed better, especially for complex multi-class rice image data.

**Q: How did SVM perform?**  
A: SVM gave strong results in binary classification tasks using tabular features.

**Q: What effect did feature removal have?**  
A: It slightly improved model generalization by reducing overfitting.

# Challenges Faced

- Difficulty in gathering balanced image datasets

- Model overfitting in early CNN attempts

- Limited GPU access for deep learning training

- Streamlit setup for deployment required environment configuration

# Conclusions and Future Work

- Successfully automated rice type classification

- Image-based deep learning can be extended to other grains

- Future ideas:

- Train with more diverse and real-world data

- Integrate mobile app for real-time prediction

- Optimize CNN architecture using transfer learning

Summarize:

What worked well

* CNN achieved high accuracy in classifying rice types and health status.
* The Streamlit app allowed easy image upload and instant prediction.
* Data augmentation helped reduce overfitting and improved model performance.

What Needs Improvement

* The dataset had imbalance in some classes, which affected prediction fairness.
* Model performance on low-quality or blurred images was weaker.
* Real-time deployment on mobile or embedded devices still needs optimization.

Future ideas: Try new algorithms, larger dataset, real-time deployment

# References

Dataset Source: RICE TYPES/Different types of rice images  
Scikit-learn Documentation: https://scikit-learn.org  
TensorFlow Guides: https://www.tensorflow.org/tutorials  
Streamlit Documentation: https://docs.streamlit.io

- Dataset: Local storage (RICE TYPES.zip)

- ML Docs: https://scikit-learn.org

- DL Docs: https://keras.io

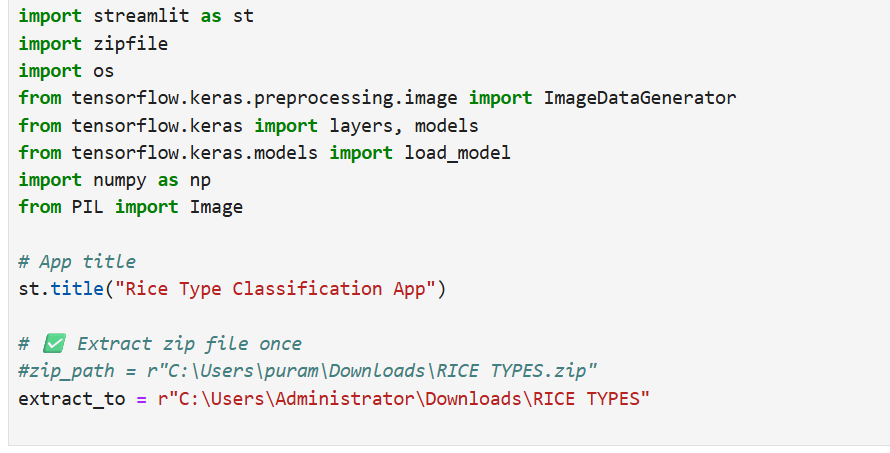
- GitHub:

Manjula:<https://github.com/ch-manjula/Rice_classification/tree/main>

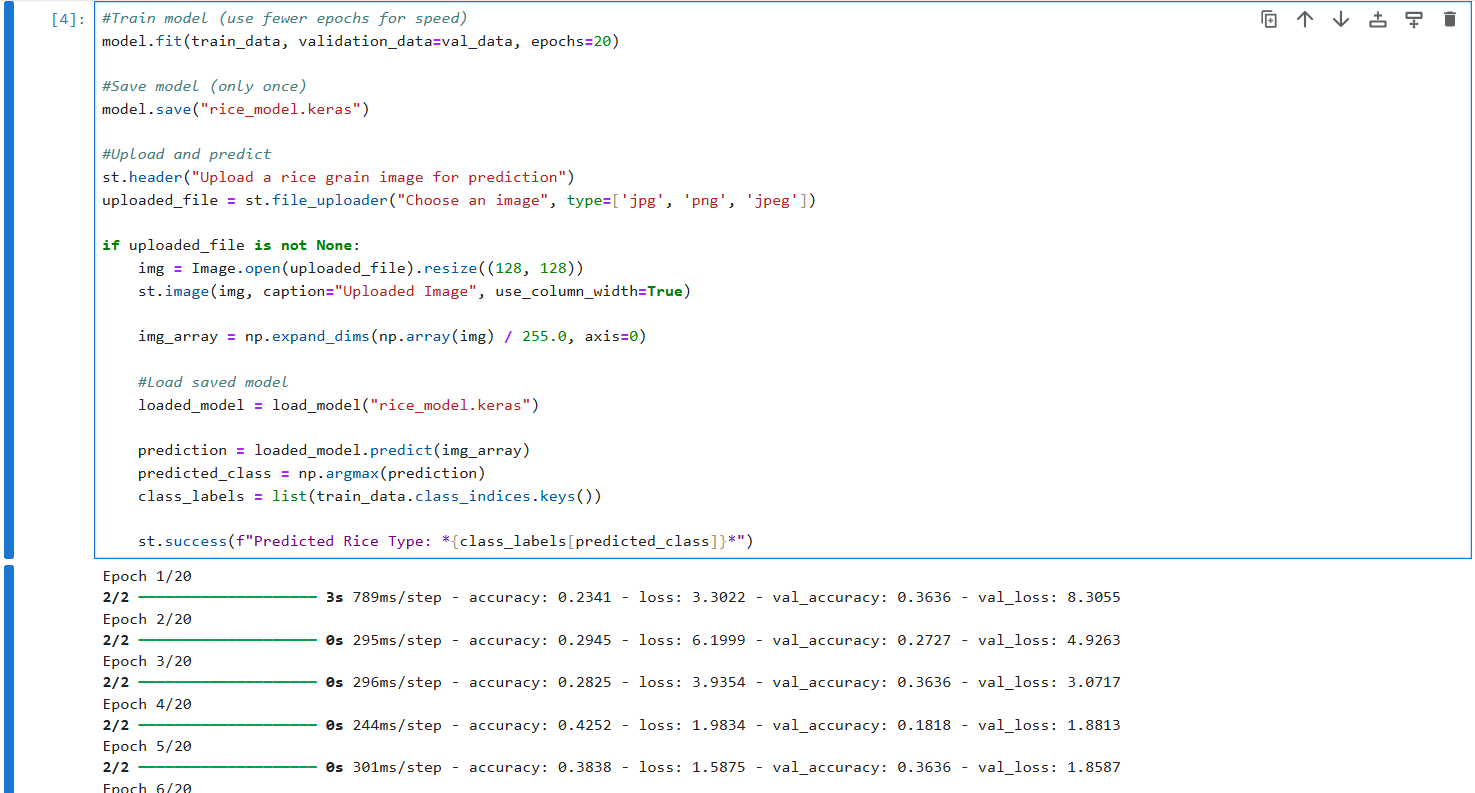
# Appendix

Code Snippets:

-Load data set:



- Model training code

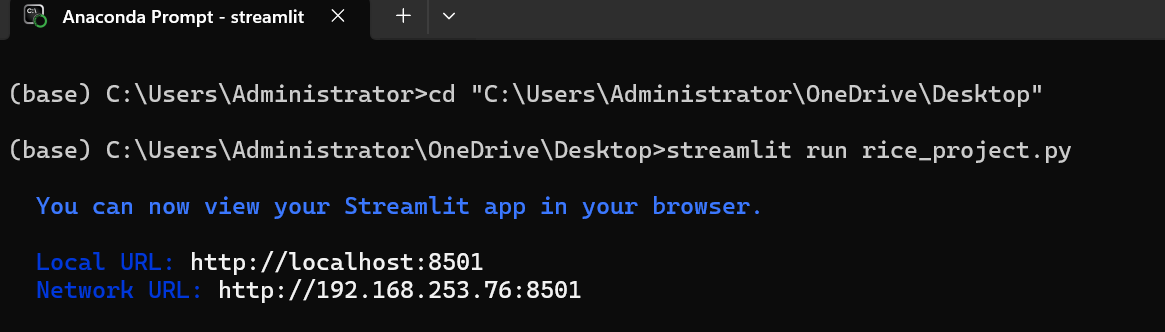


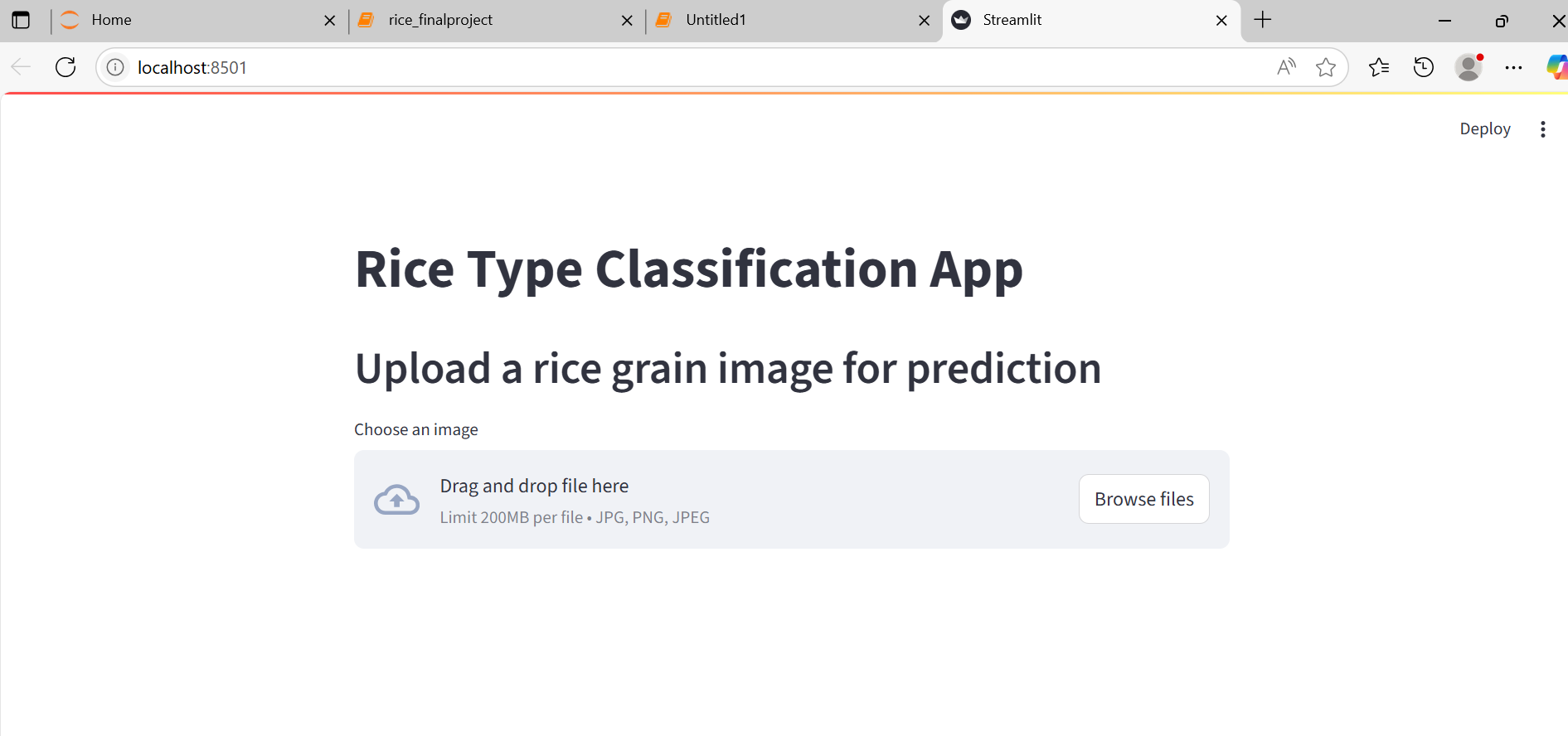
**- Image preprocessing code**

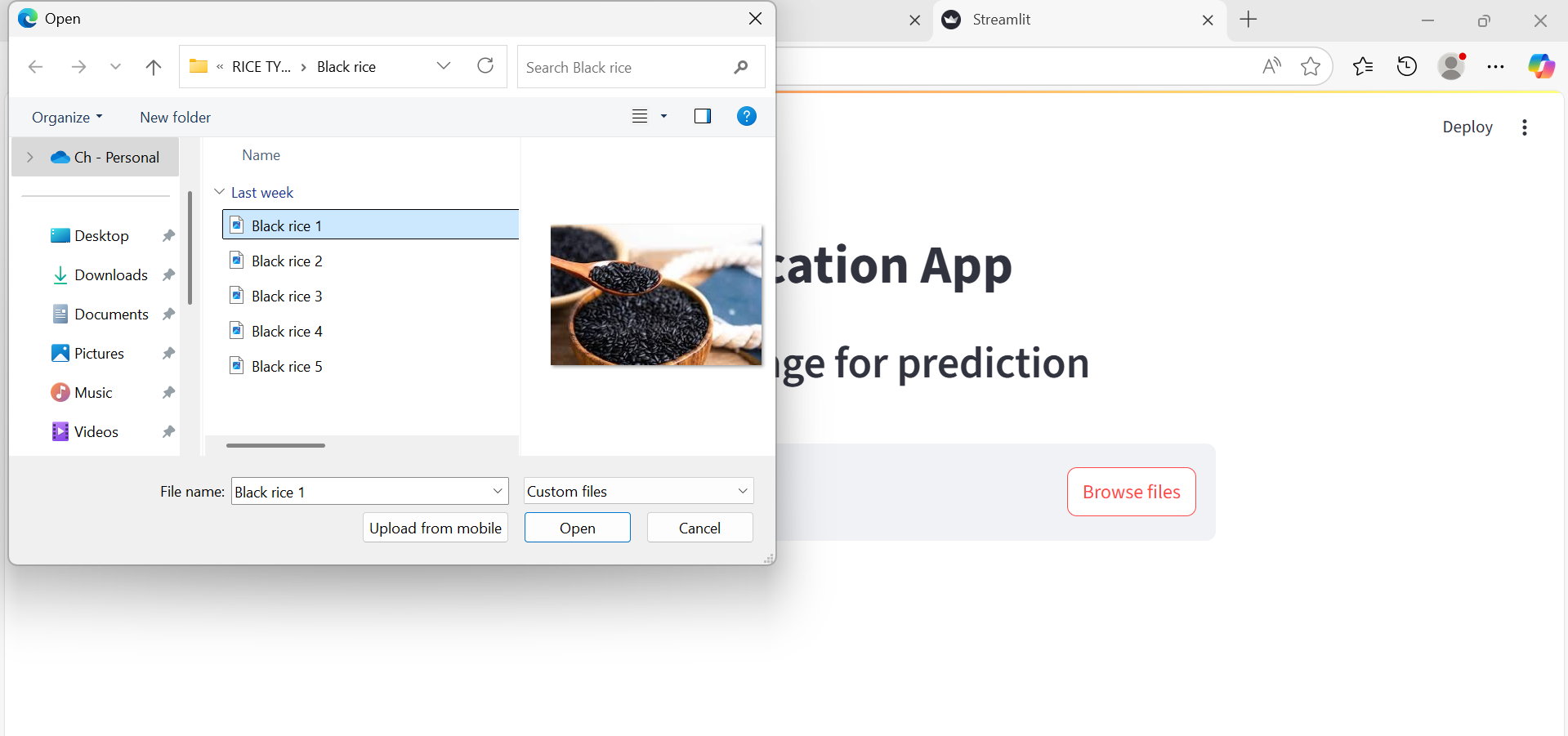
A screenshot of a computer code

AI-generated content may be incorrect.  
**- Streamlit app code**

Predicted output:









GitHub:

Manjula:<https://github.com/chmanjula/Rice_classification/tree/main>