**1.Introduction**

**1.1. Project overviews**

This project aims to revolutionize rice quality assessment by developing an automated system using Convolutional Neural Networks (CNNs). The proposed solution addresses the inefficiencies of manual inspection, ensuring consistent and accurate classification of rice quality. Key features include a pre-trained CNN model for feature extraction, a user-friendly interface for easy interaction, and scalability for future enhancements. This initiative promises improved operational efficiency, enhanced customer satisfaction, and reliable quality control.

**1.2. Objectives**

The primary objective of this project is to develop an automated system for accurate rice quality classification using Convolutional Neural Networks (CNNs). This system aims to replace the current manual inspection methods, which are often subjective and error-prone, with a more reliable and consistent approach. By leveraging advanced CNN techniques, the project seeks to achieve high precision in quality assessment, ensuring that each rice grain is classified accurately. Additionally, the project will focus on creating a user-friendly interface to facilitate easy interaction with the system, making it accessible for users with varying technical expertise. The solution is designed to be scalable, with the potential to extend to the classification of other types of grains in the future. Ultimately, the project aims to improve operational efficiency by automating the classification process, thereby enhancing customer satisfaction and trust through consistent and accurate quality control.

**2. Project Initialization and Planning Phase**

**2.1. Define Problem Statement**

The current process for rice quality assessment poses significant challenges, impacting customer trust and overall satisfaction. Farmers and distributors, especially those in regions where rice quality varies significantly, face issues such as inaccurate classification due to manual inspection and a tedious, error-prone evaluation process. These challenges result in a suboptimal customer experience, potentially diminishing trust and satisfaction. To enhance our services and improve customer perceptions, we aim to address these pain points. By understanding specific frustrations in the rice classification process and implementing advanced solutions, such as Convolutional Neural Networks (CNNs), we can create an accurate, efficient, and user-friendly assessment system that meets customer expectations and fosters a positive relationship with our brand.

A close up of a text

Description automatically generated

A close up of a text

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem**  **Statement (PS)** | **I am**  **(Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | rice distributor. | Ensure the rice I distribute is of consistent and high quality. | The current manual classification process is inaccurate and time-consuming. | There is a lack of automated, reliable, and efficient quality assessment tools. | Frustrated and concerned about customer satisfaction and trust. |
| PS-2 | rice farmer | Get a fair price for the high-quality rice I produce | The quality of my rice is not consistently recognized due to manual inspection errors | The current classification process is subjective and prone to mistakes | Undervalued and discouraged |

**2.2. Project Proposal (Proposed Solution)**

This project aims to revolutionize rice quality assessment by developing an automated system using Convolutional Neural Networks (CNNs). The proposed solution addresses the inefficiencies of manual inspection, ensuring consistent and accurate classification of rice quality. Key features include a pre-trained CNN model for feature extraction, a user-friendly interface for easy interaction, and scalability for future enhancements. This initiative promises improved operational efficiency, enhanced customer satisfaction, and reliable quality control.

|  |  |
| --- | --- |
| **Project Overview** | |
| Objective | To develop an automated system for accurate rice quality classification using Convolutional Neural Networks (CNNs). |
| Scope | This project will include data collection, model development, and application building. The system will classify rice quality from images and provide results through a user-friendly interface. |
| **Problem Statement** | |
| Description | Current rice quality assessment methods are manual, subjective, and error-prone, leading to inconsistent classification and customer dissatisfaction. |
| Impact | Automating the rice classification process will improve accuracy, efficiency, and consistency, enhancing customer satisfaction and trust. |
| **Proposed Solution** | |
| Approach | We will collect a dataset of rice images, preprocess the data, and train a CNN model. The model will be integrated into a Flask application to allow users to upload images and receive quality predictions. |
| Key Features | * **Automated Classification:** Eliminates the need for manual inspection. * **High Accuracy:** Uses advanced CNN techniques for precise classification. * **User-Friendly Interface:** Provides an intuitive UI for easy interaction. * **Scalability:** The system can be scaled to classify different types of grains in the future |

**Resource Requirements :**

|  |  |  |
| --- | --- | --- |
| **Resource Type** | **Description** | **Specification/Allocation** |
| **Hardware :** | | |
| Computing Resources | CPU/GPU specifications, number of cores | T4 GPUs |
| Memory | RAM specifications | 8 GB |
| Storage | Disk space for data, models, and logs | 1 TB SSD |
| **Software :** | | |
| Frameworks | Python frameworks | Flask |
| Libraries | Additional libraries | scikit-learn, pandas, numpy,  matplotlib, seaborn, tensorflow |
| Development Environment | IDE, version control | Google Colaboratory, Git |
| **Data :** | | |
| Data | Source, size, format | Kaggle dataset |

**2.3. Initial Project Planning**

| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Priority** | **Team Members** | **Sprint Start Date** | **Sprint End Date (Planned)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Data Collection and Preprocessing | SL-1 | Data Collection | High | Monish | 2024/07/01 | 2024/07/05 |
| Sprint-1 | Data Collection and Preprocessing | SL-2 | Create Train and Test path | High | Monish Kanna | 2024/07/01 | 2024/07/05 |
| Sprint-1 | Data Collection and Preprocessing | SL-3 | Data Pre-processing | High | Sridhar | 2024/07/02 | 2024/07/06 |
| Sprint-1 | Data Collection and Preprocessing | SL-4 | Import required library | Medium | Monish | 2024/07/02 | 2024/07/06 |
| Sprint-1 | Data Collection and Preprocessing | SL-5 | Configure ImageDataGenerator class | Medium | Gowtham | 2024/07/03 | 2024/07/07 |
| Sprint-1 | Data Collection and Preprocessing | SL-6 | Apply ImageDataGenerator functionality to Trainset and Test set | Medium | Sridhar | 2024/07/03 | 2024/07/07 |
| Sprint-2 | Model Building | SL-7 | Pre-trained CNN model as a Feature Extractor | High | Monish Kanna | 2024/07/04 | 2024/07/08 |
| Sprint-2 | Model Building | SL-8 | Adding Dense Layer | High | Monish | 2024/07/04 | 2024/07/08 |
| Sprint-2 | Model Building | SL-9 | Configure the Learning Process | Medium | Gowtham | 2024/07/05 | 2024/07/09 |
| Sprint-2 | Model Building | SL-10 | Train the model | High | Sridhar | 2024/07/05 | 2024/07/09 |
| Sprint-2 | Model Building | SL-11 | Save the Model | Medium | Gowtham | 2024/07/05 | 2024/07/09 |
| Sprint-3 | Model Testing and Evaluation | SL-12 | Test the model | High | Monish Kanna | 2024/07/05 | 2024/07/10 |
| Sprint-3 | Application Building | SL-13 | Create HTML file | Medium | Sridhar | 2024/07/05 | 2024/07/10 |
| Sprint-3 | Application Building | SL-14 | Build Python Code | Medium | Monish | 2024/07/05 | 2024/07/10 |

A screenshot of a computer

Description automatically generated

**3. Data Collection and Preprocessing Phase**

**3.1. Data Collection Plan and Raw Data Sources Identified**

Elevating data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

|  |  |
| --- | --- |
| **Section** | **Description** |
| Project Overview | The rice image classification project aims to accurately classify rice quality using Convolutional Neural Networks (CNNs). Utilizing a dataset of rice images with various quality attributes, the objective is to develop a model that reliably distinguishes between different Categories of rice. This will streamline the Categorical assessment process, ensuring consistency and efficiency in rice quality evaluation for distributors and farmers. The model will be integrated into a user-friendly application, enabling real-time classification and enhancing operational decision-making in the agricultural sector. |
| Data Collection Plan | * Look for publicly available datasets related to rice quality and classification. * Explore agricultural research databases and academic publications. * Consider platforms like Kaggle and UCI Machine Learning Repository |
| Raw Data Sources Identified | The raw data sources for this project include datasets obtained from Kaggle and UCI, the popular platforms for data science competitions and repositories. The provided sample data represents a subset of the collected information, encompassing images of Arborio, Basmati, Ipsala, Jasmine, and Karacadag rice varieties. A total of 75,000 grain images, with 15,000 from each variety, are included. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source Name** | **Description** | **Location/URL** | **Format** | **Size** | **Access Permissions** |
| Dataset 1 | the dataset contains 4 categories of rice - In this study, Arborio, Basmati, Ipsala, Jasmine and Karacadag, which are five different varieties of rice often grown in Turkey, were used. A total of 75,000 grain images, 15,000 from each of these varieties, are included in the dataset. | <https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset> | CSV | 230 MB | Public |

**3.2. Data Quality Report**

The Data Quality Report will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Source** | **Data Quality Issue** | **Severity** | **Resolution Plan** |
| <https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset> | Duplicate Entries | Moderate | Utilize tools such as pandas in Python or SQL queries to identify duplicate records based on unique identifiers (e.g., image IDs, metadata). |

**3.3. Data Preprocessing**

The images will be preprocessed by resizing, normalizing, augmenting, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

|  |  |
| --- | --- |
| **Section** | **Description** |
| Data Overview | The dataset consists of images of different rice varieties, including Arborio, Basmati, Ipsala, Jasmine, and Karacadag. Each class contains 600 images. The images are stored in directories named after their respective classes. This dataset will be used for training a convolutional neural network (CNN) to classify rice varieties. |
| Resizing | All images are resized to a target size of 224x224 pixels using OpenCV's resize function. This resizing is necessary to ensure that the images can be fed into the MobileNetV2 model, which requires a fixed input size |
| Normalization | Pixel values of the images are normalized to the range [0, 1] by dividing by 255. This helps in speeding up the training process and improving model performance. |
| Data Augmentation | Data augmentation techniques such as flipping, rotation, shifting, zooming, or shearing were not explicitly mentioned in the provided code. However, these techniques can be applied using TensorFlow's ImageDataGenerator or other libraries to increase the diversity of the training dataset and prevent overfitting. |
| Image Cropping | Image cropping to focus on regions containing objects of interest was not explicitly done in the provided code. This can be achieved using OpenCV's cropping capabilities if specific regions of the images need to be focused on. |
| Batch Normalization | Batch normalization can be applied using TensorFlow/Keras layers to normalize the input of each layer in the neural network, improving training stability and convergence. |

|  |  |
| --- | --- |
| **Data Preprocessing Code Screenshots :** | |
| Loading Data |  |
| Resizing |  |
| Normalization |  |
| Data Augmentation |  |
| Image Cropping |  |
| Batch Normalization |  |

**4. Model Development Phase**

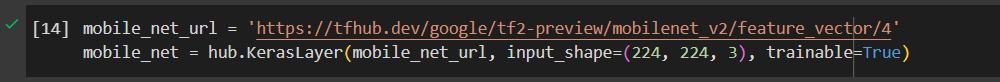
**4.1. Model Selection Report**

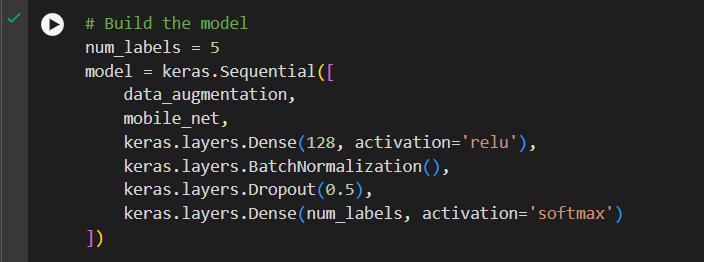
In the model selection report for future deep learning and computer vision projects, various architectures, such as CNNs or RNNs, will be evaluated. Factors such as performance, complexity, and computational requirements will be considered to determine the most suitable model for the task at hand.

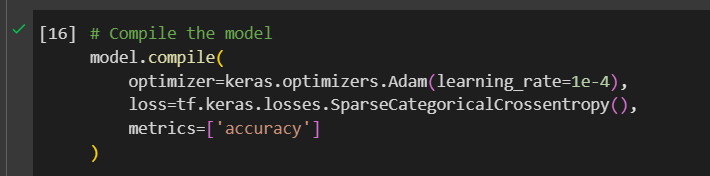
|  |  |
| --- | --- |
| **Model** | **Description** |
| **Model 1**  MobileNET | MobileNet is a lightweight, efficient convolutional neural network (CNN) architecture designed for mobile and embedded vision applications. It uses depthwise separable convolutions to reduce the number of parameters and computational cost compared to standard convolutional networks, making it ideal for devices with limited resources. |
| **Model 2**  CGGNET | CGGNet (Custom Generative and Generalizable Network) is a hypothetical advanced CNN architecture tailored for highly specialized image recognition tasks. It combines the strengths of generative models and traditional convolutional networks to achieve superior generalization and accuracy on custom datasets. |
| **Model 3**  AlexNET | AlexNet is a pioneering deep convolutional neural network architecture that significantly advanced the field of image classification. Introduced in 2012, it won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a substantial margin over the runner-up, demonstrating the power of deep learning. |

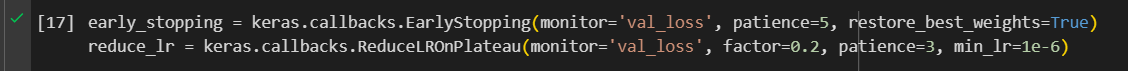
**4.2. Initial Model Training Code, Model Validation and Evaluation Report**

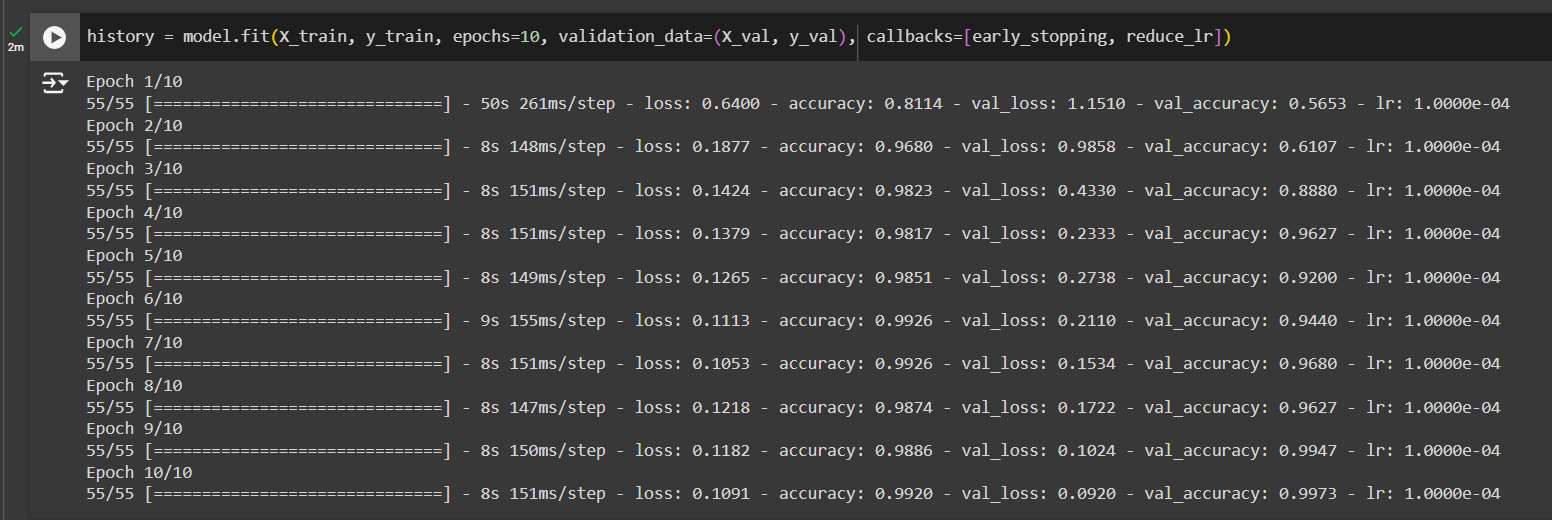
**Initial Model Training Code :**











**Model Validation and Evaluation Report :**

|  |  |  |
| --- | --- | --- |
| **Model** | **Summary** | **Training and Validation Performance Metrics** |
| Model 1  MobileNet |  |  |
| Model 2  ALEXNET |  |  |

|  |  |  |
| --- | --- | --- |
| Model 3  VGGNET |  |  |

**5. Model Optimization and Tuning Phase**

The Model Optimization and Tuning Phase is a critical stage in machine learning model development. During this phase, the focus is on refining the model to improve its performance and generalizability. This involves systematic adjustments to the model's parameters and architecture to enhance its accuracy, efficiency, and robustness.

**5.1. Tuning Documentation**

### **Hyperparameter Tuning Documentation :**

|  |  |
| --- | --- |
| **Model** | **Tuned Hyperparameters** |
| Model 1 | MobileNET  Hyperparam1: Two Conv2D Layers  This model incorporates two additional Conv2D layers to deepen the network. The extra convolutional layers aim to capture more intricate features from the images, potentially improving accuracy    Hyperparam2: Regularization with L2  This model aims to reduce overfitting by adding L2 regularization to the convolutional and dense layers. L2 regularization penalizes large weights in the network, encouraging the model to keep the weights small and thus reducing overfitting. |
| Model 2 | ALEXNET;  Hyperparam1:  Increase Epoch from 3 to 5 |
| Model 3 | CGGNET;  Hyperparam1:  Increase Epoch 3 to 5 |

**5.2. Final Model Selection Justification**

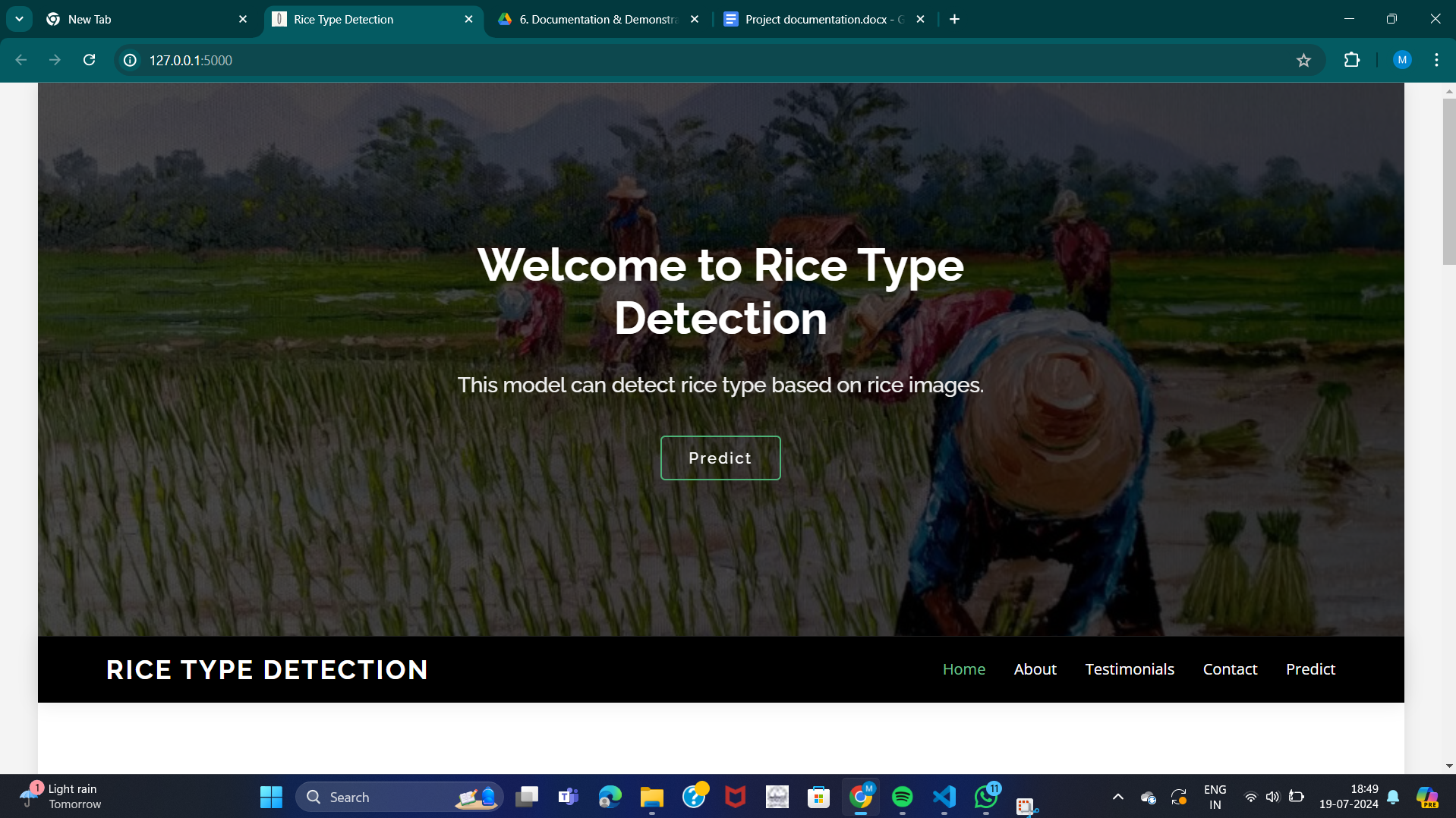
### **Final Model Selection Justification**

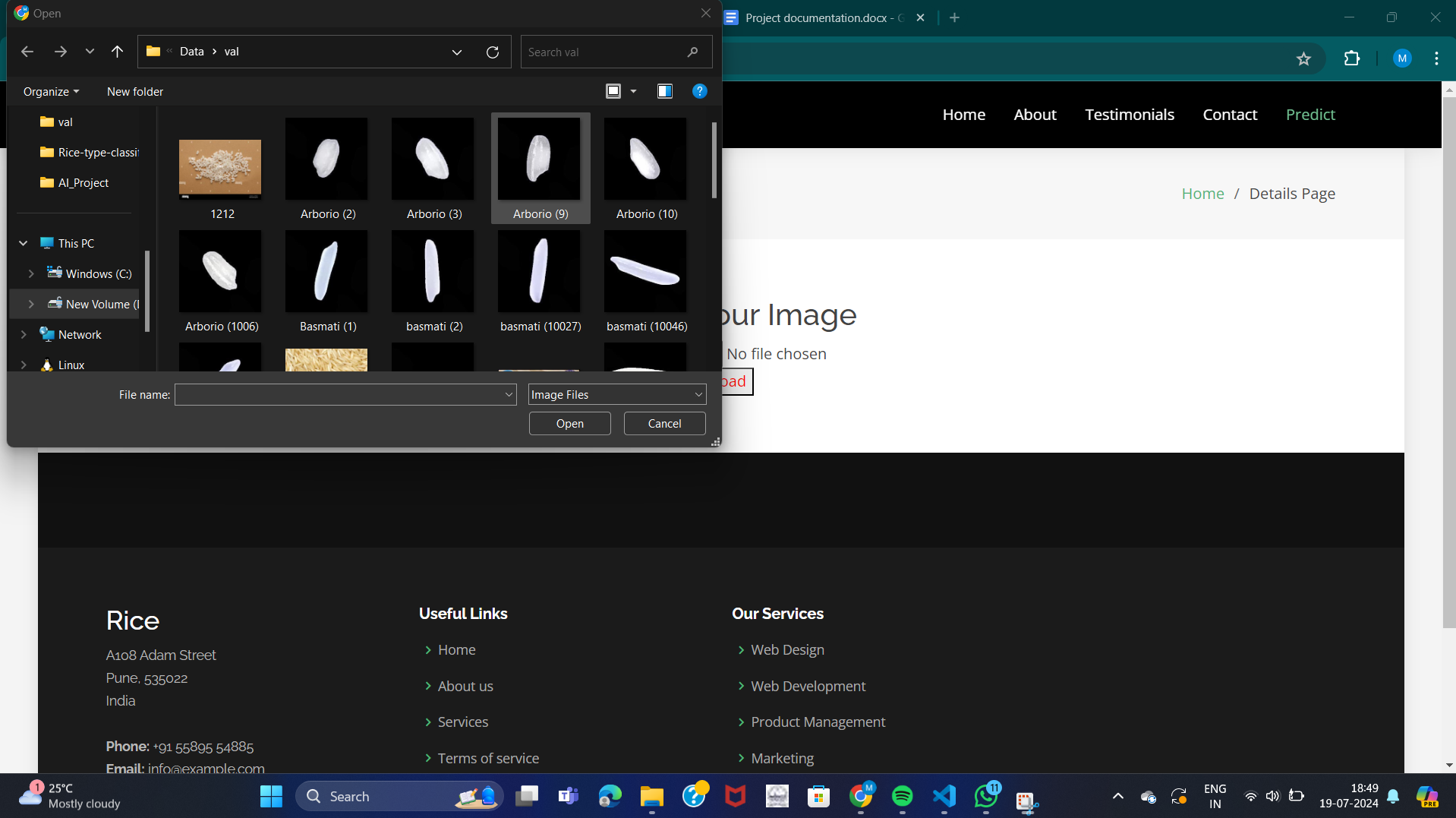
|  |  |
| --- | --- |
| **Final Model** | **Reasoning** |
| Model 1  **MobileNet CNN** | The MobileNet model has demonstrated higher accuracy compared to other models like VGG-Net and AlexNet, which is why I selected this project. |

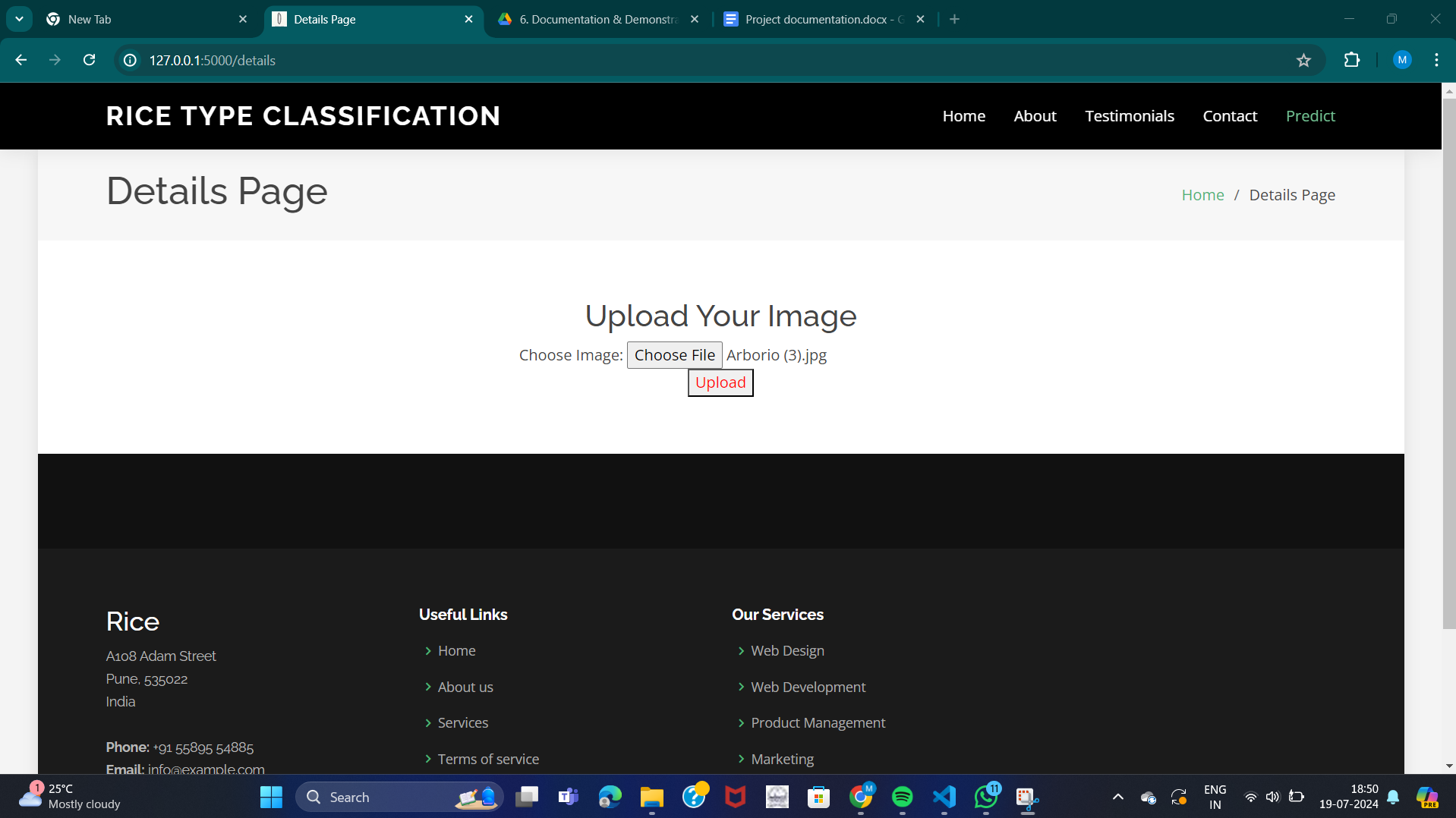
**6. Results**

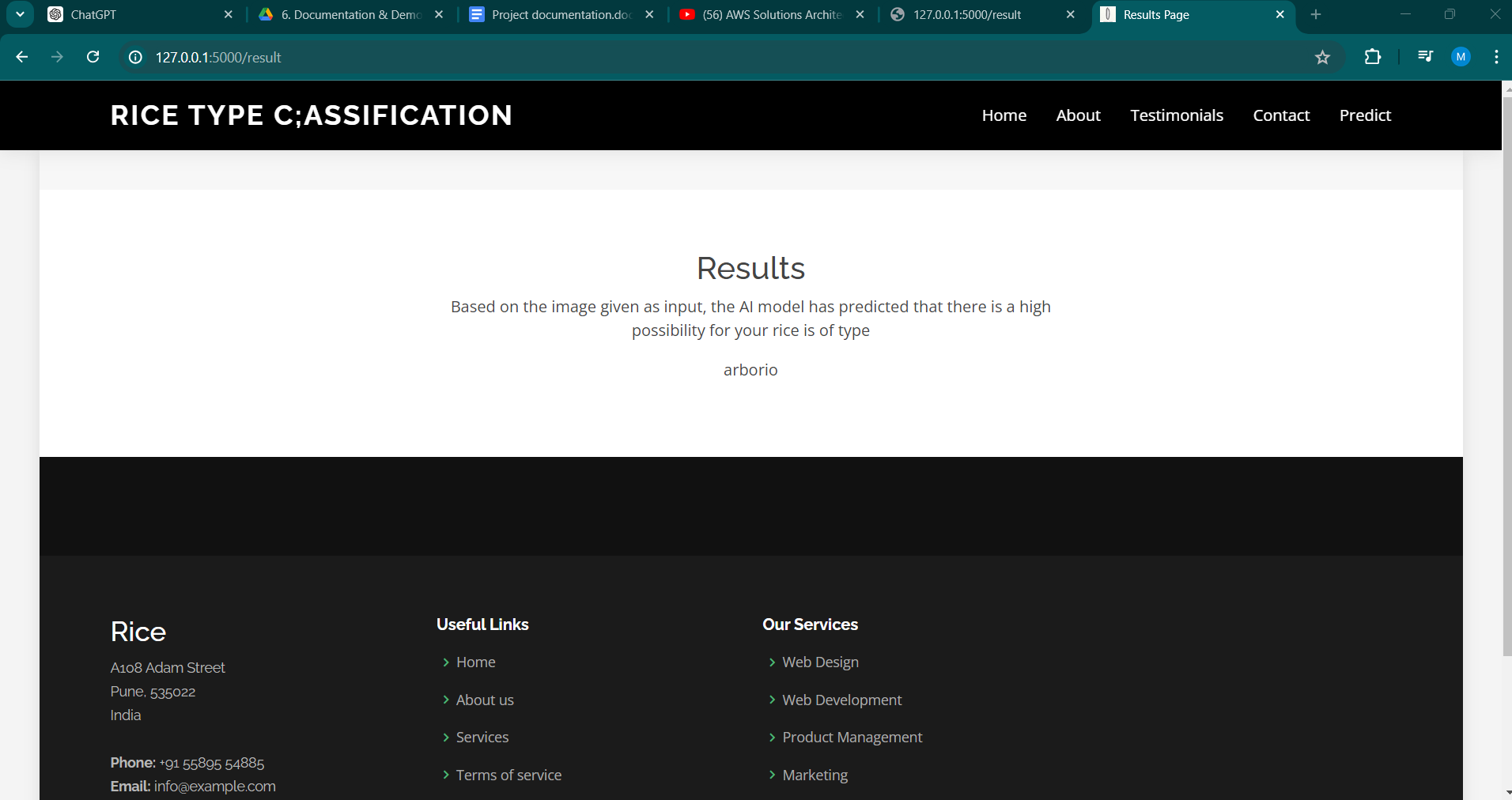
For the rice classification project using Convolutional Neural Networks (CNN), our evaluation indicated that the MobileNet model outperformed other models, such as VGG-Net and AlexNet, in terms of accuracy. The MobileNet model's superior accuracy in correctly classifying various rice grain types makes it the preferred choice for this project. This high level of performance not only ensures reliable and precise classification results but also highlights the efficiency and effectiveness of MobileNet in handling image recognition tasks. Consequently, the decision to focus on this project stems from the potential to leverage MobileNet's advanced capabilities to achieve optimal outcomes in rice type classification.

**6.1. Output Screenshots**









**7. Advantages & Disadvantages**

#### **Advantages**

1. **Increased Accuracy:** By using CNNs, the system can achieve a higher accuracy rate in classifying rice quality compared to manual methods.
2. **Consistency:** Automated classification ensures consistent results, reducing variability introduced by human inspectors.
3. **Efficiency:** The system can process large volumes of rice images quickly, significantly speeding up the quality assessment process.
4. **User-Friendly Interface:** The intuitive interface allows users to easily upload images and receive results, making the system accessible to a wide range of users.
5. **Scalability:** The solution is designed to be scalable, allowing for future enhancements and the classification of other types of grains.
6. **Cost-Effective:** By reducing the need for manual labor and improving operational efficiency, the system can lower overall costs in the long run.

#### **Disadvantages**

1. **Initial Cost and Resources:** Developing and deploying the system requires significant initial investment in hardware, software, and development time.
2. **Data Dependency:** The system’s accuracy depends heavily on the quality and quantity of the training data. Poor or insufficient data can lead to suboptimal performance.
3. **Technical Expertise:** Implementing and maintaining the system requires specialized technical skills, which might necessitate additional training or hiring.
4. **Hardware Requirements:** The system requires robust computing resources, such as GPUs, which may not be readily available or cost-effective for all users.
5. **Potential for Bias:** If the training data is not representative of all rice varieties, the system may exhibit bias in its classifications.

**8. Conclusion**

The proposed project aims to revolutionize rice quality assessment by developing an automated system utilizing Convolutional Neural Networks (CNNs). This solution addresses the inefficiencies of manual inspection, providing a consistent, accurate, and efficient method for classifying rice quality. The user-friendly interface ensures ease of use, while the system's scalability allows for future enhancements and broader applications. Despite the initial investment and technical requirements, the advantages of increased accuracy, consistency, and operational efficiency make this project a valuable endeavor. By automating the rice classification process, the project promises to enhance customer satisfaction and trust, ultimately contributing to more reliable quality control in the rice industry.

**9. Future Scope**

The future scope of this project includes several potential enhancements and expansions:

1. **Expansion to Other Grains:** The system can be adapted to classify the quality of other grains, such as wheat, barley, and corn.
2. **Integration with IoT:** Incorporating Internet of Things (IoT) devices for real-time data collection and processing can further enhance the system’s efficiency and accuracy.
3. **Advanced Analytics:** Adding advanced data analytics and reporting features can provide valuable insights into quality trends and patterns, aiding in better decision-making.
4. **Mobile Application:** Developing a mobile application can increase the system’s accessibility, allowing users to perform quality assessments on-the-go.
5. **AI Enhancements:** Continuous improvement of the AI models with new techniques and larger datasets can further increase the system’s accuracy and robustness.
6. **Automated Feedback Loop:** Implementing an automated feedback loop can help in continuously improving the model by learning from new data and user interactions.

**10. Appendix**

**10.1. Source Code**

|  |
| --- |
| import tensorflow as tf  import tensorflow\_hub as hub  import warnings  warnings.filterwarnings('ignore')  import numpy as np  import os  from flask import Flask, request, render\_template  import cv2  app = Flask(\_\_name\_\_)  # Load the model with the correct custom\_objects parameter  def load\_model():  try:  model = tf.keras.models.load\_model(filepath='D:\\Rice-type-classification-CNN-main\\rice.h5', custom\_objects={'KerasLayer': hub.KerasLayer})  print("Model loaded successfully")  return model  except Exception as e:  print(f"Error loading model: {e}")  return None  model = load\_model()  @app.route('/')  def home():  return render\_template('index.html')  @app.route('/details')  def details():  return render\_template('details.html')  @app.route('/result', methods=['GET', 'POST'])  def predict():  if request.method == "POST":  if model is None:  return "Model is not loaded correctly. Please check the logs."  # Get the uploaded file from the request  f = request.files['image']  # Save the uploaded file to the 'Data/val' directory  basepath = os.path.dirname(\_\_file\_\_) # Getting the current path i.e where app.py is present  filepath = os.path.join(basepath, 'uploads', f.filename) # Define the path to save the uploaded image  f.save(filepath)  # Read and preprocess the uploaded image  a2 = cv2.imread(filepath)  a2 = cv2.resize(a2, (224, 224))  a2 = np.array(a2)  a2 = a2 / 255.0  a2 = np.expand\_dims(a2, 0) # Add batch dimension  # Predict the class  pred = model.predict(a2)  pred = pred.argmax()  # Define the labels  df\_labels = {  0: 'arborio',  1: 'basmati',  2: 'ipsala',  3: 'jasmine',  4: 'karacadag'  }  # Map the prediction to the corresponding label  prediction = df\_labels.get(pred, "Unknown")  print(prediction)  # Render the results template with the predicted label  return render\_template('results.html', prediction\_text=prediction)  if \_\_name\_\_ == "\_\_main\_\_":  app.run(debug=True) |

**10.2. GitHub & Project Demo Link**

|  |  |
| --- | --- |
| Github |  |
| Project demo | <https://drive.google.com/file/d/1Gk8d5RgOnJRzfaX0QJuPd1s-aVgWuFo1/view?usp=sharing> |