Lauren March
Student ID: 001421111
SIM3 Task 2
Mushroom Identification ML Project

Table of Contents

Part A

Letter of Transmittal

June 10, 2024

Portia Cremini
Mushi Bistro
1234 Portobello Ln,
Spore, Mushroom Land

Dear Ms.Cremini,

Subject: Proposal for Mushroom Identifying Application

I am writing to present a proposal for a data product in order to aid Mushi Bistro. Mushi Bistro is a scratch kitchen that forages their own mushrooms for their cuisine. Our project's goal is to help Mushi Bistro with identifying mushrooms using a machine learning application designed to accurately identify mushrooms from user-provided images. The application will not only support the culinary creativity of Mushi Bistro but also enhance safety and efficiency in their foraging process.

Mushi Bistro currently faces challenges in cost-effectively and accurately identifying mushrooms in a timely manner. These challenges can result in potential health risks and risk the quality of their dishes. Their current method of manual identification is time-consuming and is potentially prone to errors. Therefore, an automated solution is desirable.

Our proposed solution is an intuitive Windows OS application that uses a machine learning algorithm to identify mushrooms accurately and in real-time. With the use of advanced image processing and machine learning techniques, this application will provide identification that is both instant and precise. This will ensure Mushi Bistro can confidently use the correct mushrooms in their cuisine.

Not only will the application significantly enhance the accuracy and efficiency of mushroom identification, it will ensure ingredient safety along with improving the quality of the dishes. This will allow chefs to increase their focus on culinary creativity rather than their foraging process.

The project will be funded by Mushi Bistro under contract by ML Solutions Inc.. The costs inclusive to this will be data acquisition, development, and any required computational resources. The timeline will be an estimated 140 hours including planning, development, and documentation. The data is licensed under the Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license. This ensures that no proprietary or sensitive information is compromised and the data is used for non-commercial purposes. Mushi Bistro intends to use the application internally and will not sell or use it for commercial purposes in order to comply with the Attribution-NonCommercial 4.0 International license.

With extensive experience in machine learning, AI, and computer science concepts I am well equipped to lead this project and bring it to fruition. My expertise in developing and deploying

machine learning models will ensure the successful execution and implementation of this solution.

I am highly confident that this project will benefit Mushi Bistro and support their unique culinary endeavors. I look forward to discussing this proposal further. I will be available to address any questions or concerns you may have.

Sincerely,

Lauren March
Senior AI/ML Software Engineer

Project Proposal

Summary of the Problem

Currently Mushi Bistro uses mushroom experts and text resources (i.e. books) for identifying their mushrooms. This is not only costly, but inefficient and potentially a health risk. With the use of a machine learning application that processes images that can accurately identify mushrooms in real-time, Mushi Bistro can cut down on the cost of hiring mushroom experts and safely identify mushrooms confidently to use in their cuisine.

Description of how Data Product Benefits the Customer and Supports the Decision-Making Process

Our proposed solution will provide Mushi Bistro with an accurate and quick method for mushroom identification. The application will help chefs and foragers make efficient and reliable decisions by automating the identification process. This will not only maintain safety from any mushroom related health hazard, but will also ensure high quality of their curated dishes. With this application, chefs and foragers can save time and bring their focus closer to their other business needs and decisions.

Outline of the Data Product

The data product will be a stand-alone Windows OS application developed using Python and TensorFlow that uses Convoluted Neural Networks (CNNs) to accurately predict mushroom types from uploaded images. We will also leverage transfer learning techniques that will allow us to increase accuracy and reliability. “Transfer learning is usually expressed through the use of pre-trained models” (Marcelino, 2018).

Specifically we will be using ResNet-50 as our pre-trained model, which “is CNN architecture that belongs to the ResNet (Residual Networks) family, a series of models designed to address the challenges associated with training deep neural networks. Developed by researchers at Microsoft Research Asia, ResNet-50 is renowned for its depth and efficiency in image classification tasks.”(Potrimba, 2024).

The data product will also feature an intuitive user-friendly interface where users can upload their own images of mushrooms and the application will give them the identified mushroom type. A metrics dashboard section of the application will also provide visualization tools such as a confusion matrix and how well the model performs during training (accuracy/loss) for the user to evaluate the efficacy of the model.

Description of the Data That Will Be Used to Construct the Data Product

The data used for this project will be sourced from a dataset uploaded to Kaggle.com. This data will consist of labeled images of 60 different species of mushrooms with a total of 20,251 images. This dataset was modified in order to balance the classes. The original dataset totalled ~50,000 across ~100 classes. The color images from this dataset will inherently have features such as pixel intensity and color channels. Exploratory Data Analysis (EDA) will be used to identify patterns and anomalies within the dataset. The data is under the Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license, and therefore will be able

to be used so long as the data is attributed and not commercially used. Since, Mushi Bistro has no intention of selling this application and will solely be using it internally, they will be in compliance with this license.

Objectives and Hypotheses of the Project

- **Objective**: To develop an intuitive Windows OS application that reliably and accurately identifies mushrooms using CNNs.
- **Hypotheses:**
    - The CNN-based model will achieve a training accuracy of above 90%.
    - The CNN-based model will achieve 80% validation accuracy or better in identifying mushroom types from images.
    - The model will achieve a precision score of above 70%.
    - The model will achieve a recall score of above 70%.
    - The model will achieve an F1 score of above 70%.

Outline of the Project Methodology

1. **Data Acquisition and Preprocessing**: Gather and clean data from Kaggle.com.
2. **Model Development**: Develop and train a CNN model using Python and TensorFlow.
3. **Evaluation**: Evaluate the model using metrics such as accuracy, precision, recall, and F-1 score. Implement a confusion matrix to understand misclassifications.
4. **Implementation**: Develop the stand-alone application compatible with Windows OS.
5. **Validation**: Validate the model against success criteria and make necessary adjustments.
6. **Documentation**: Document the process and the results, including visualization and summary statistics.

Funding Requirements

The project will be funded by Mushi Bistro under contract by ML Solutions Inc. inclusive of data acquisition, development, and any required computational resources cost. Therefore, this project will not need additional outside funding or contributions.

Impact of the Solution on Stakeholders

The primary stakeholders, which include the chefs, foragers, waitstaff, and management at Mushi Bistro, will benefit from improved accuracy and efficiency in mushroom identification. Additionally, this will also benefit the overall dining experience for Mushi Bistro's customers ensuring quality and safe dishes for all its patrons.

Ethical and Legal Considerations and Precautions That Will Be Used When Working With and Communicating About Sensitive Data

All data used will be sourced ethically from publicly licensed datasets, ensuring no proprietary or sensitive information is compromised.

Your Expertise Relevant to the Solution You Propose

With extensive experience in machine learning, AI, and computer science concepts I am well equipped to lead this project and bring it to fruition. My expertise in developing and deploying

machine learning models will ensure the successful execution and implementation of this solution.

Part B

Executive Summary

Decision Support Problem/Opportunity

Mushi Bistro currently uses mushroom experts and text resources (i.e. mushroom identifying books) to identify mushrooms. This practice is not only costly, but is inefficient and potentially poses health risks. Our project aims to develop a machine learning application that processes user uploaded photos to accurately identify mushrooms in real-time, thereby reducing costs and enhancing safety and efficiency.

Customer Description and Needs

Our primary customers are Mushi Bistro's chefs, foragers, waitstaff, and management. The general need is for a reliable and quick method of mushroom identification. This will allow them to free up time and focus more on their culinary creativity and other business needs. The application will directly help them fulfill those needs by providing an automated means for mushroom identification. This will ensure their safety and quality needs are met.

Existing Data Product Gaps

The current method of manually identifying mushrooms by experts and text resources is costly in both time and money, and is prone to errors. There is not an efficient automated solution in place, which underscores the need for our proposed application. There is also no way to validate the accuracy of the hired experts and foragers and offer business insights for continuous improvement practices.

Data Available/Collected

The data will be sourced from a dataset from Kaggle.com. All data will be ethically sourced and will have an Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license. This will ensure that no proprietary or sensitive information is compromised. The data will consist of various images for 60 species of mushrooms with a total of 20,251 images. This dataset was modified in order to balance the classes. The original dataset totalled ~50,000 across ~100 classes. Mushi Bistro intends to use the application solely for internal, non-commercial purposes.

Methodology

We will follow the CRISP-DM methodology which includes phases of business understanding, data understanding, data preparation, modeling, evaluation, and deployment.

Business Understanding

During the business understanding phase, we will meet with the applicable stakeholders to focus on fully defining the project's goals and full scope, and identify key performance indicators that will help track and adequately assess our goals. This phase will ensure that our project

aligns with Mushi Bistro's business goals and target the specific needs of Mushi Bistro's customer base.

Data Understanding

The Data Understanding phase will encompass collecting mushroom dataset from Kaggle.com. We will then conduct exploratory data analysis (EDA) in order to understand the dataset.

Data Preparation

During Data Preparation, we will clean and preprocess the data for modeling. This will involve labeling the data, normalization, and ensure that the data is suitable for CNNs.

Modeling

During the modeling phase, we will utilize ResNet50, a pre

Evaluation

For the Evaluation phase, we will evaluate the model using metrics such as accuracy, precision, recall, and F1-score. We will also implement a confusion matrix to understand misclassification and validate that the model meets the project's objectives.

Deployment

The Deployment phase will include developing the mushroom identification application that is compatible with Windows OS, integrate the pre-trained model, and test and prepare the release of the application for Mushi Bistro.

Documentation

In addition to a technical release, we will also prepare documentation of findings throughout the whole CRISP-DM cycle, provide any training materials for internal and external use, and set up a support framework.

Deliverables

The deliverables associated with the design and development of the data product will include:

- **Preprocessed and Labeled Dataset** - Prior to training the model, we will have a preprocessed and cleaned dataset that will be labeled for supervised learning.
- **Pre-Trained CNN Model** - Once that dataset has been cleaned, we will deliver the use of a pre-trained CNN model that will be trained on the cleaned images of mushrooms in order to accurately identify mushroom types.
- **Stand-alone application for Windows OS** - The third deliverable will be the actual application that the user can upload images to for mushroom identification. This will include an intuitive and user-friendly interface that will be compatible with Windows OS.
- **Documentation** - The final deliverable will be the documentation that will include the process and results, and visualizations and how well the model performs during and

after training along with summary statistics of the data. There will also be additional business documentation for maintenance of the product.

Implementation Plan

The implementation plan will include acquiring data, training the model, development of the application, and an evaluation. The anticipated outcome will be a reliable and user-friendly Windows OS application that accurately identifies mushroom types that Mushi Bistro can use to enhance their cuisine and food safety practices.

Validation and Verification

The KPIs that we will be looking at in order to validate and verify the developed data product will be accuracy, precision, recall, and F1-score metrics. We will also use a confusion matrix to understand any possible misclassifications. Prior to release, we will have QA teams conduct iterative testing and release testing. After the release we will collect user-feedback to ensure that Mushi Bistro's requirements are met and they are satisfied with the product.

Programming Environment and Resources

These costs are based on the projected timeline of this project.

- Hardware:
    - High Spec'd PC running Windows 10 or 11
- Software:
    - Machine learning frameworks (e.g., TensorFlow).
- Programming Languages:
    - Python
- Human resources at salary positions (9am-5pm work hours):
    - Data scientists x 1
    - Project manager x 1
    - Software engineers x 1
    - QA engineers x 1
    - Devops engineer x 1
    - Technical Writer x 1
- Cost:
    - Machine learning frameworks: $0 (opensource)
    - Human resources: $100,000
    - Estimated overall budget of $110,000

Projected Timeline

Sprint	Start	End	Tasks

1	May 15th, 2024	June 7tht, 2024	Planning and Design (80 hours)
2	June 10th, 2024	June 28th, 2024	Development (40 hours)
3	July 1, 2024	July 12, 2024	Documentation (20 Hours)

Milestones

Data Acquisition and Preprocessing
May 15, 2024 - May 28, 2024 (2 weeks)
Data is cleaned and preprocessed.
Resources:

- Data Scientist x 1
- Project Manager x 1

Model Development
May 29, 2024 - June 4, 2024 (1 week)
Model is trained on the cleaned data.
Resources:

- Data Scientist x 1
- Project Manager x 1

Evaluation and Validation
June 5, 2024 - June 11, 2024 (1 week)
Model accuracy is evaluated and checked for validity.
Resources:

- Data Scientist x 1
- Project Manager x 1

Application Development
June 12, 2024 - June 26, 2024 (2 weeks)
The application is being developed by the team and tested by QA.
Resources:

- Developer x 1
- QA Engineer x 1
- Project Manager x 1

Final Testing and Deployment
June 27, 2024 - July 3, 2024 (1 week)
QA completes final release testing.
Resources:

- QA Engineer x 1
- DevOps Engineer x 1
- Project Manager x 1

Documentation
July 4, 2024 - July 12, 2024 (1 week)
Documentation is prepared, written, and deployed.
Resources:

- Technical Writer x 1
- Developer x 1
- Project Manager x 1

Part C

Application

Relevant Directory Structure:

WGU_CS_Capstone/

- data/
  - modelDataVisualizations/
  - processedData/ (Too large for 200MB limit for submission see screenshots in Part D, images and .h5 file omitted, class_names.pkl still present)
  - rawData/ - (Too large for 200MB limit for submission see screenshots in Part D, images and .h5 file omitted)
  - testData/
- models/
  - model_kpis.pkl
  - Mushroom_identifier.keras
- src/
  - data_process.py
  - mushi_id_app.py
  - main.py
  - predict.py
  - train_model.py
  - classdistro.py
  - kpi_read.py

data/:

This folder contains:

- Raw data from https://www.kaggle.com/datasets/thehir0/mushroom-species.
- Processed data after running data_process.py script.
- Test data of images pulled from a Google image search in order to test the accuracy of the mushroom identification application.
- Images for the 3 visualizations included in the Metrics Dashboard of the application.

models/:

This folder contains:

- Mushroom_identifier.keras created from running the train_model.py script that saves the trained model keras file.
- Model_kpis.pkl created from running the train_model.py script that saves the kpi data of validation accuracy, precision, recall, and F1-score to a pickle file.

src/:

This folder contains:

- data_process.py script resizes, splits the data between training and validation sets, and preprocesses the data for training the model.
- train_model.py takes the preprocessed data and trains it using a transfer learning from a pre-trained model (ResNet50) and a custom CNN.

- predict.py script that was used for testing the model predictions prior to the creation of the full application.
- mushi_id_app.py script that runs a version of the prediction script along with the GUI.
- main.py script as a centralized point of app entry that runs the mushi_id_app.py, but has the flexibility to run additional scripts that will integrate with the mush_id_app.py in the future if necessary.
- classdistro.py script was used for Exploratory Data Analysis (EDA) in order to see the class distribution and plot histograms of the class data.
- kpi_read.py script was used to read the kpi Pickle file located at WGU_CS_Capstone\models\model_kpis.pkl, this file stores the KPI data created after training the model.

Part D

Business Vision

Introduction

A scratch-made, in-house foraged, restaurant, Mushi Bistro, is on a mission to enhance its operational efficiency by implementing a machine learning-based mushroom identification application. The goal of the application is to provide accurate and real-time identification of mushrooms from user-uploaded images. This will reduce the reliance of manual identification methods which will thereby reduce costs, both monetarily and time.

Vision Statement

The vision for the mushroom identification application is to leverage advanced machine learning techniques that will revolutionize the way Mushi Bistro identifies and utilizes foraged mushrooms. This will bolster their efforts in safety and quality of their dishes along with providing a unique and creative experience for Mushi Bistro's customers.

Business Objectives

1. Enhance Operational Efficiency: Automation of mushroom identification process during foraging to save time and reduce cost.
2. Ensure Safety and Compliance: Provide accurate identification to mitigate health risks that may arise from misidentification of mushrooms.
3. Support Culinary Innovation: Allow chefs to focus more on culinary creativity by freeing up their time from manual identification.
4. Improve Decision Making: Offer data-driven insights that will be saved for future use along with a section for visualization tools and summary statistics.

Product Overview

The application will be compatible with Windows OS built using Python and TensorFlow. It will utilize Convolutional Neural Networks (CNNs) to accurately identify mushroom species from user-uploaded images. The application will be intuitive and include a user-friendly interface for image upload and display the identified mushroom type. In addition, the application will include a reporting dashboard with visualization tools for data analysis.

Key Features

1. **Real-Time Identification**: Quick and accurate identification of mushrooms from user-uploaded images using a pre-trained CNN model.
2. **User-Friendly Interface**: Intuitive interface for easy upload and result display.
3. **Reporting and Analytics**: Tools for visualizing data including a confusion matrix and training and validation accuracy/loss over the time of training.
4. **Data Security and Privacy**: Ensure all data used and processed meets ethical and legal standards. Ensure customer data is not collected.

Target Marker and Users

The primary users of the mushroom identification application will be:

- **Chefs and Foragers**: For quick identification during the foraging process.
- **Management**: To monitor and ensure the safety and quality of ingredients used in the kitchen.

Benefits to Mushi Bistro

1. **Cost Savings**: Reduce the cost of manually identifying mushrooms from hiring mushroom experts.
2. **Time Efficiency**: Faster identification, allows chefs and management to focus on culinary and business tasks.
3. **Safety**: Reduce risk hazard associated with mushroom misidentification.
4. **Data Insights**: Detailed reports and visualizations for informed decision-making.

Implementation Plan

1. **Data Acquisition and Preprocessing**: Collect and preprocess data from publicly available datasets.
2. **Model Development**: Develop and use a pre-trained CNN model using Python and TensorFlow.
3. **Application Development**: Build a stand-alone Windows OS application and integrate the pre-trained model.
4. **Testing and Validation**: Conduct thorough testing to ensure accuracy and reliability.
5. **Deployment and Maintenance**: Deploy the application and provide ongoing support and updates.

Project Timeline and Milestones

- **Data Acquisition and Preprocessing**: May 15, 2024 - May 28, 2024
- **Model Development**: May 29, 2024 - June 4, 2024
- **Evaluation and Validation**: June 5, 2024 - June 11, 2024
- **Application Development**: June 12, 2024 - June 26, 2024
- **Final Testing and Deployment**: June 27, 2024 - July 3, 2024
- **Documentation**: July 4, 2024 - July 12, 2024

Resources and Budget

- **Hardware**: High-spec Windows PC.
- **Software**: TensorFlow, Python.
- **Human Resources**: Data scientists, software engineers, project managers, QA engineers, DevOps engineers, and technical writers.
- **Budget**: Estimated overall budget of $110,000, including human resources and other operational costs.

Conclusion

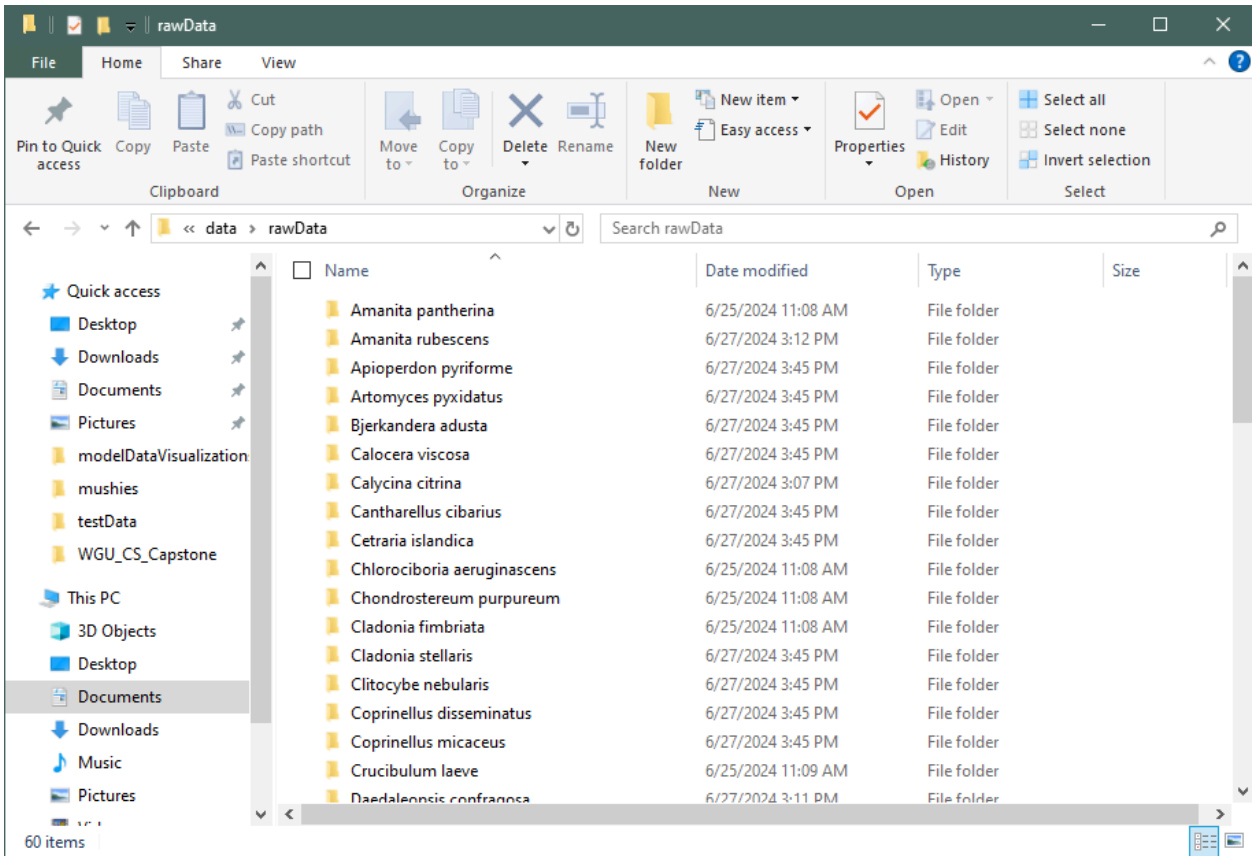
The mushroom identification application will not only enhance the efficiency and safety of the foraging process, but also enhance the culinary innovation at Mushi Bistro. The application will

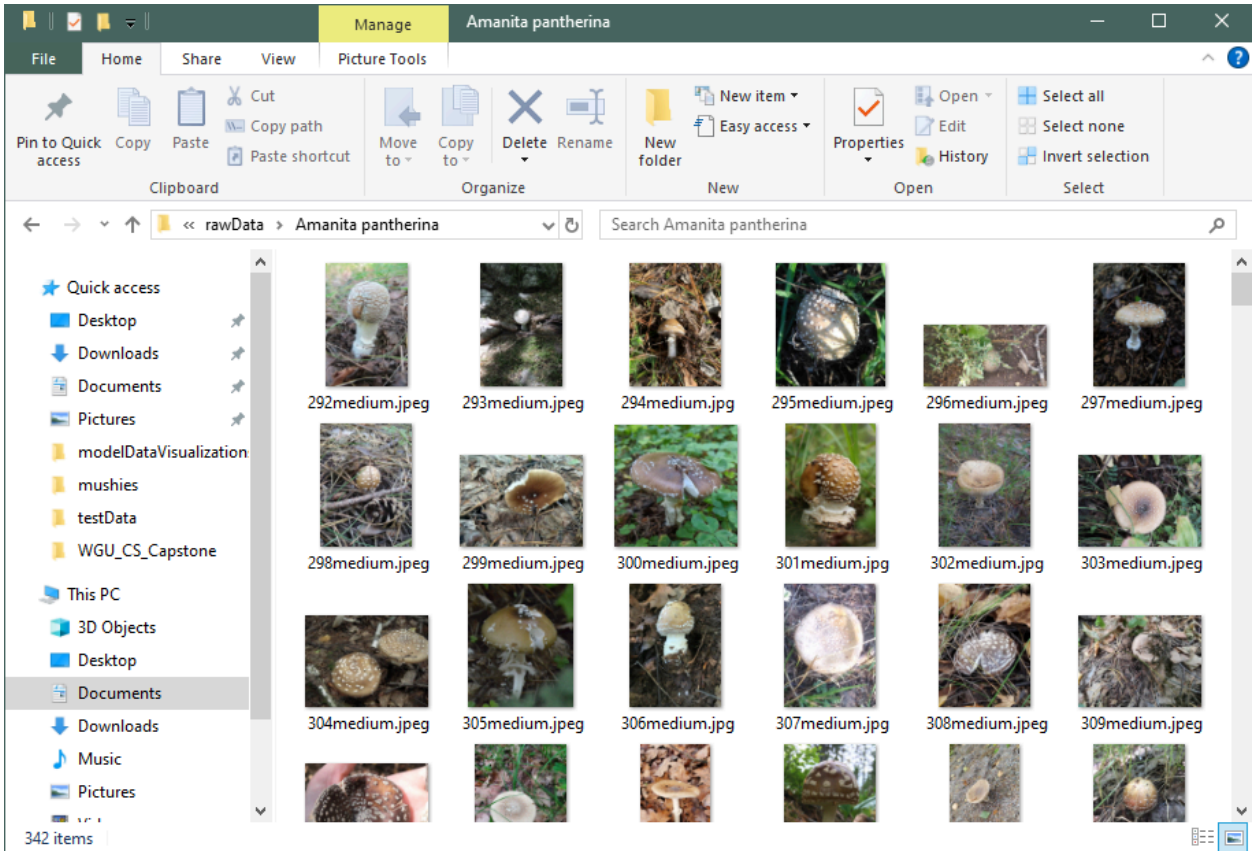
reduce costs and time by ways of automation. This will support Mushi's Bistro mission to provide unique and exceptional dining experiences.

Raw and Cleaned Datasets with the Code and Executable Files Used to Scrape and Clean Data

Raw Data:

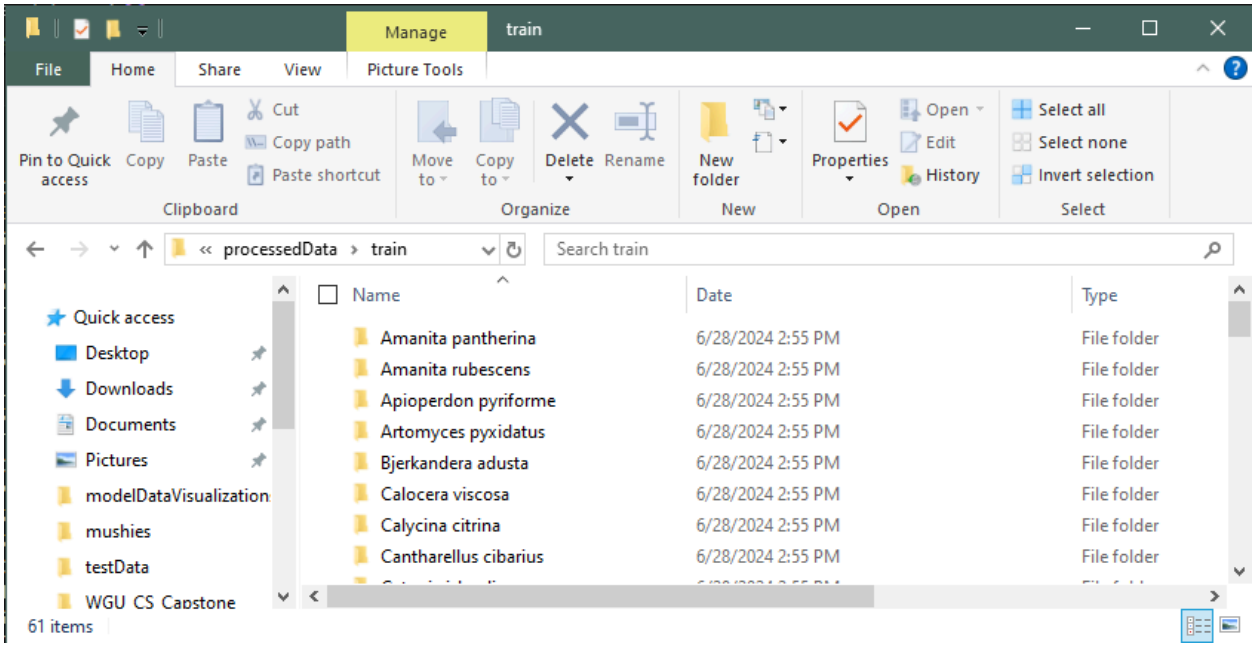
- The raw dataset comprises 20,251 images across 60 classes of mushrooms. Each of the 60 mushroom classes have ~300-350 images that correspond to each mushroom type. The data has been modified by deleting some images and classes for better class balancing. The original Kaggle.com dataset totalled ~50,000 across ~100 classes. Please see the screenshots below of the rawData directory.
- Source: https://www.kaggle.com/datasets/thehir0/mushroom-species

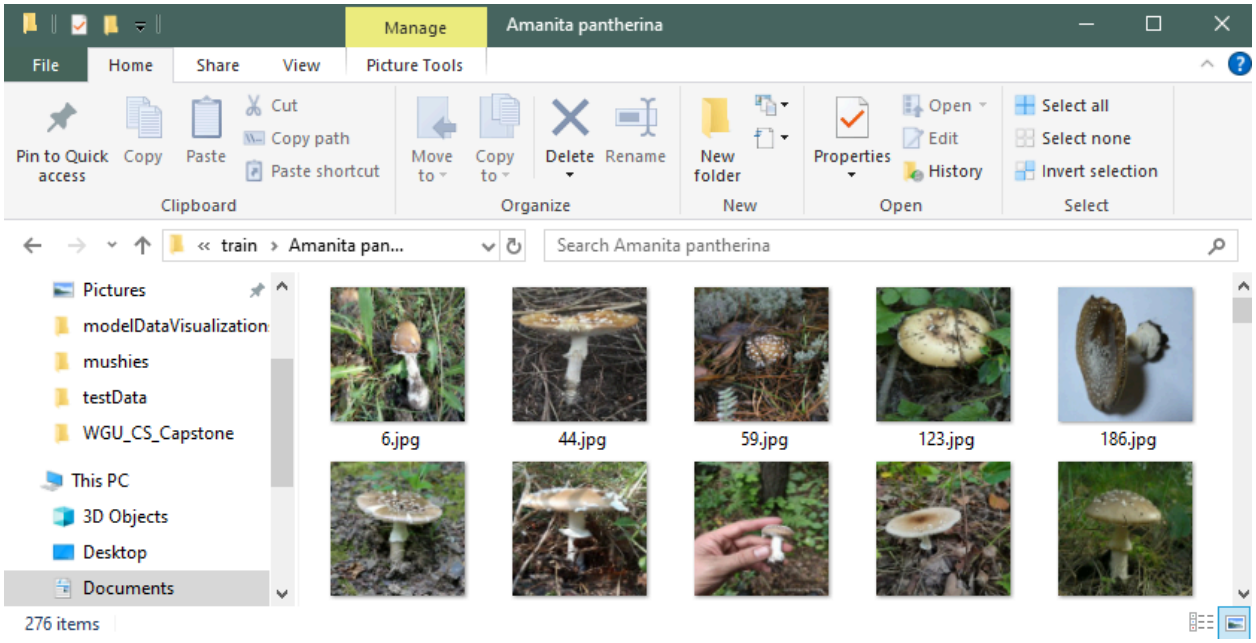


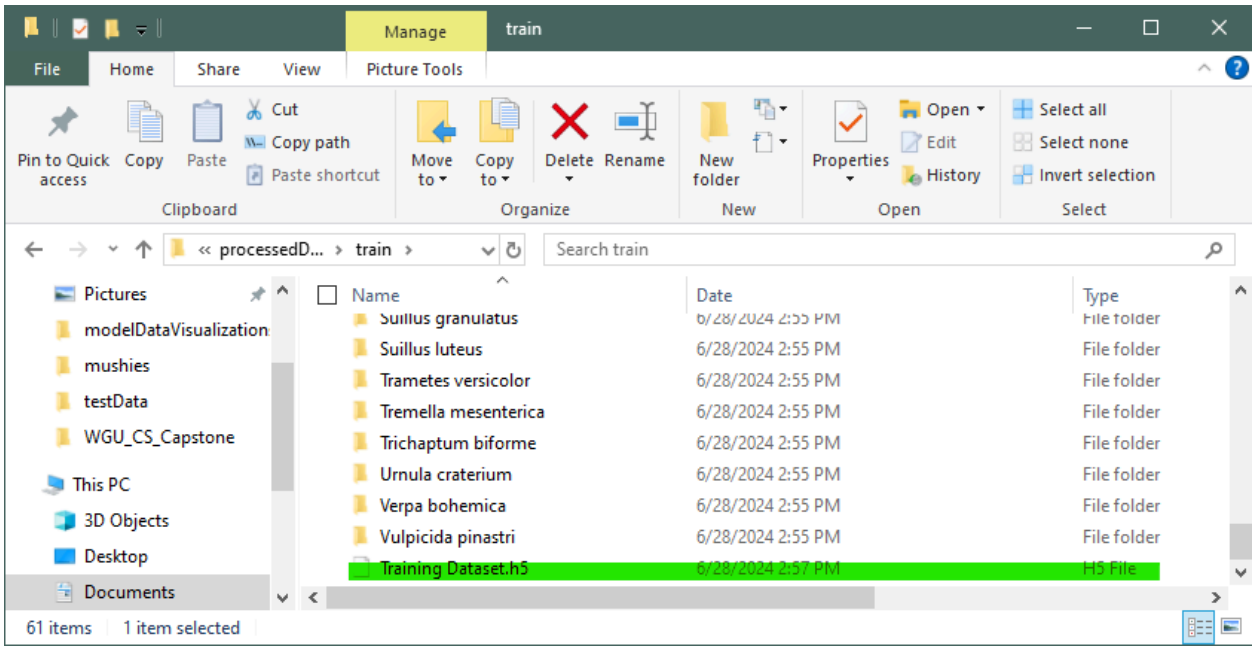


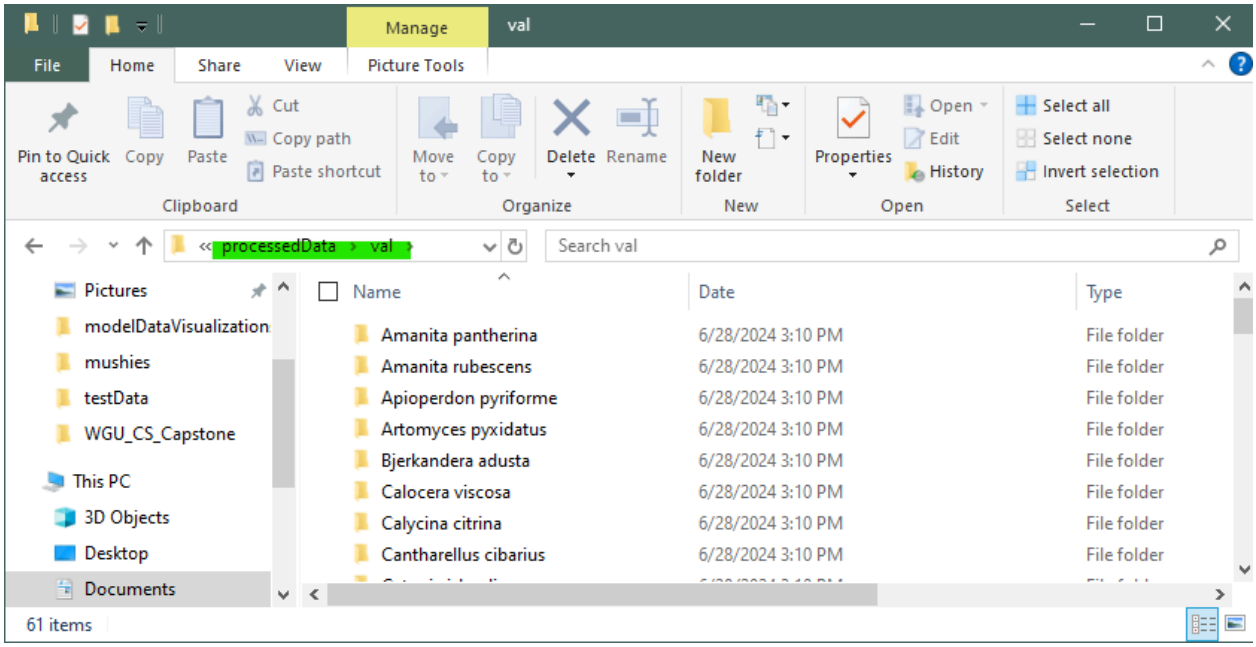
Cleaned Data:

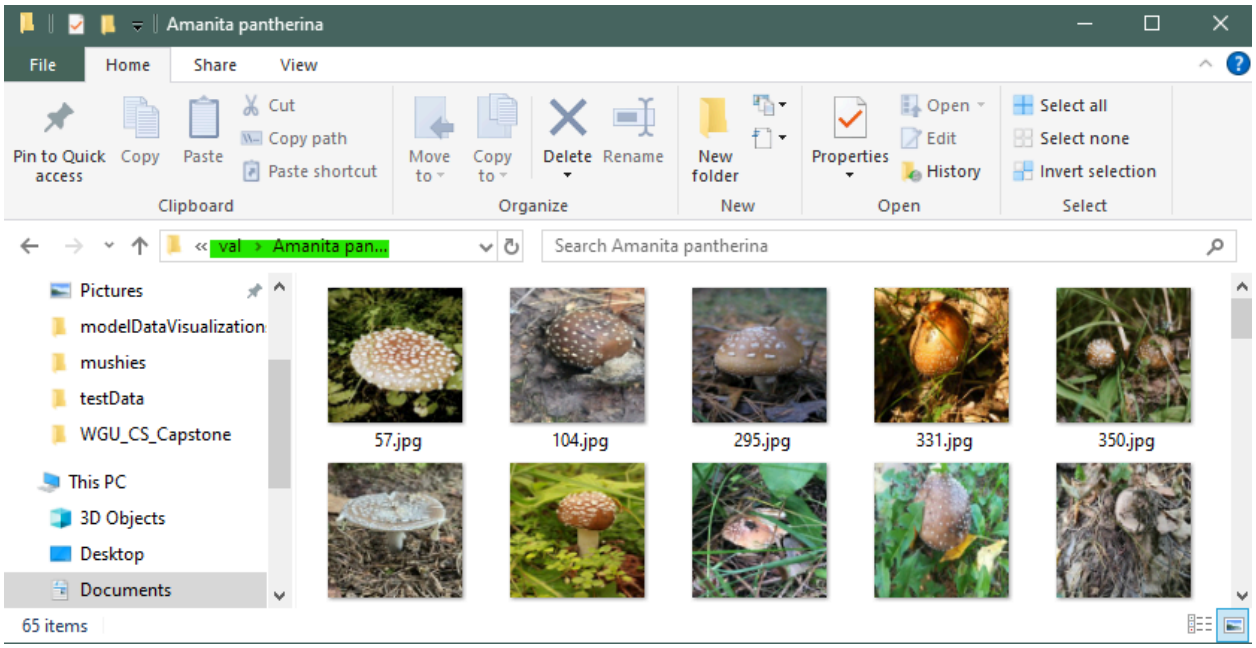
- The processed dataset consists of an ~80/20 split for training and validation sets. The training set is 16,192 images while the validation set has 4063 images. These are saved as images and as a Hierarchical Data Format 5 file format for the model to quickly access this large dataset. Please see screenshots below.
- Created by running data_process.py script.

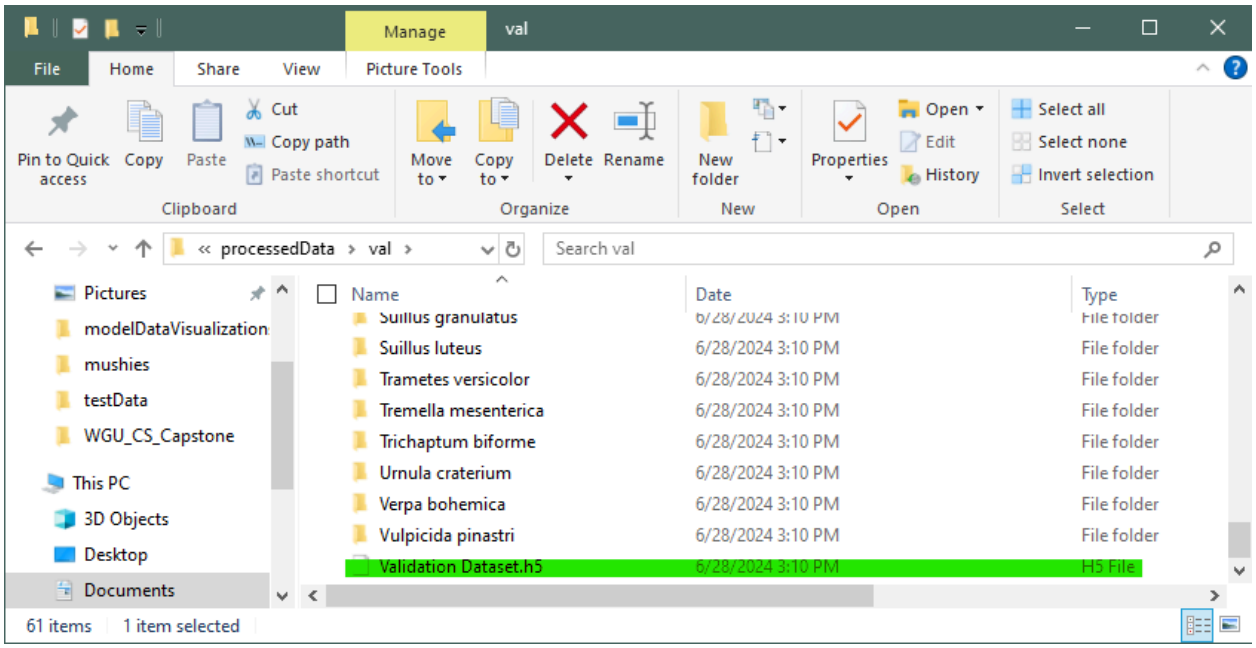



Manage
Amanita pantherina
File
Home
Share
View
Picture Tools
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
« train › Amanita pan...
Search Amanita pantherina
Pictures
modelDataVisualization
mushies
testData
WGU_CS_Capstone
This PC
3D Objects
Desktop
Documents
6.jpg
44.jpg
59.jpg
123.jpg
186.jpg
276 items


Manage
train
File
Home
Share
View
Picture Tools
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
« processedD... › train ›
Search train
Pictures
modelDataVisualization
mushies
testData
WGU_CS_Capstone
This PC
3D Objects
Desktop
Documents
Name
Date
Type
Suillus granulatus
Suillus luteus
Trametes versicolor
Tremella mesenterica
Trichaptum biforme
Urnula craterium
Verpa bohemica
Vulpicida pinastri
Training Dataset.h5
6/28/2024 2:55 PM
File folder
6/28/2024 2:57 PM
H5 File
61 items
1 item selected


Manage
val
File
Home
Share
View
Picture Tools
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
« processedData › val ›
Search val
Pictures
modelDataVisualization
mushies
testData
WGU_CS_Capstone
This PC
3D Objects
Desktop
Documents
Name
Date
Type
Amanita pantherina 6/28/2024 3:10 PM File folder
Amanita rubescens 6/28/2024 3:10 PM File folder
Apioperdon pyriforme 6/28/2024 3:10 PM File folder
Artomyces pyxidatus 6/28/2024 3:10 PM File folder
Bjerkandera adusta 6/28/2024 3:10 PM File folder
Calocera viscosa 6/28/2024 3:10 PM File folder
Calycina citrina 6/28/2024 3:10 PM File folder
Cantharellus cibarius 6/28/2024 3:10 PM File folder
61 items


Amanita pantherina
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
« val › Amanita pan...
Search Amanita pantherina
Pictures
modelDataVisualization
mushies
testData
WGU_CS_Capstone
This PC
3D Objects
Desktop
Documents
57.jpg
104.jpg
295.jpg
331.jpg
350.jpg
65 items



Code for processing data:

- Located at: /WGU_CS_Capstone/src/data_process.py
- This script resizes images, splits data into training and validation sets, and preprocesses the data for training the model. It also saves images outside of the .h5 for human readable images after normalization.

Code Used to Perform the Analysis of the Data

- **data_process.py -** Processes raw data, performs image preprocessing and prepares datasets for training.
- **train_model.py -** Trains the model using the processed data.
- **predict.py -** Headless script that predicts mushroom types from the trained model.
- **mushroom_id_app.py -** GUI that incorporates a version of the prediction script that allows the user to upload an image for prediction.

Assessment of the Hypotheses for Acceptance or Rejection

Hypotheses:

- The CNN-based model will achieve a training accuracy of above 90%.
- The CNN-based model will achieve 80% validation accuracy or better in identifying mushroom types from images.
- The model will achieve a precision score of above 70%.
- The model will achieve a recall score of above 70%.
- The model will achieve an F1 score of above 70%.

Assessment:

The hypotheses were tested by evaluating the model using metrics such as training accuracy, overall validation accuracy, precision, recall, and F1-score.

Results:

- Epochs: 14/50 (training ended after the 14th Epoch since implementing early stopping)
- Training Accuracy: 99.96%
- Validation Accuracy: 84.35%
- Validation Loss: 0.6386
- Precision: 84.87%
- Recall: 84.35%
- F1 Score: 84.40%

```
Training the model...
Epoch 1/50
2024-07-08 12:34:43.994382: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8100
506/506 [==============================] - 165s 281ms/step - loss: 1.5269 - accuracy: 0.6030 - val_loss: 4.2281 - val_accuracy: 0.0216 - lr: 1.0000e-04
Epoch 2/50
506/506 [==============================] - 135s 267ms/step - loss: 0.5491 - accuracy: 0.8331 - val_loss: 1.4563 - val_accuracy: 0.5888 - lr: 1.0000e-04
Epoch 3/50
506/506 [==============================] - 113s 223ms/step - loss: 0.2998 - accuracy: 0.9090 - val_loss: 0.7685 - val_accuracy: 0.7748 - lr: 1.0000e-04
Epoch 4/50
506/506 [==============================] - 167s 330ms/step - loss: 0.1859 - accuracy: 0.9407 - val_loss: 0.9307 - val_accuracy: 0.7505 - lr: 1.0000e-04
Epoch 5/50
506/506 [==============================] - 138s 272ms/step - loss: 0.1347 - accuracy: 0.9582 - val_loss: 0.7648 - val_accuracy: 0.7976 - lr: 1.0000e-04
Epoch 6/50
506/506 [==============================] - 141s 279ms/step - loss: 0.1018 - accuracy: 0.9689 - val_loss: 0.9204 - val_accuracy: 0.7872 - lr: 1.0000e-04
Epoch 7/50
506/506 [==============================] - 207s 407ms/step - loss: 0.0953 - accuracy: 0.9711 - val_loss: 0.9105 - val_accuracy: 0.7914 - lr: 1.0000e-04
Epoch 8/50
506/506 [==============================] - 120s 238ms/step - loss: 0.0625 - accuracy: 0.9828 - val_loss: 0.9463 - val_accuracy: 0.7790 - lr: 1.0000e-04
Epoch 9/50
506/506 [==============================] - 126s 248ms/step - loss: 0.0244 - accuracy: 0.9946 - val_loss: 0.6211 - val_accuracy: 0.8435 - lr: 2.0000e-05
Epoch 10/50
506/506 [==============================] - 100s 196ms/step - loss: 0.0074 - accuracy: 0.9990 - val_loss: 0.6237 - val_accuracy: 0.8482 - lr: 2.0000e-05
Epoch 11/50
506/506 [==============================] - 195s 385ms/step - loss: 0.0052 - accuracy: 0.9993 - val_loss: 0.6510 - val_accuracy: 0.8447 - lr: 2.0000e-05
Epoch 12/50
506/506 [==============================] - 109s 214ms/step - loss: 0.0037 - accuracy: 0.9996 - val_loss: 0.6416 - val_accuracy: 0.8490 - lr: 2.0000e-05
Epoch 13/50
506/506 [==============================] - 104s 205ms/step - loss: 0.0034 - accuracy: 0.9996 - val_loss: 0.6379 - val_accuracy: 0.8500 - lr: 4.0000e-06
Epoch 14/50
506/506 [==============================] - 104s 206ms/step - loss: 0.0032 - accuracy: 0.9996 - val_loss: 0.6386 - val_accuracy: 0.8522 - lr: 4.0000e-06
Evaluating the model...
126/126 [==============================] - 7s 45ms/step
Shape of val_generator.labels: (4063,)
Shape of y_val_pred_classes: (4032,)
Model training complete and saved to C:\Users\psycl\Documents\GitHub\WGU_CS_Capstone\models\mushroom_identifier.keras
Model KPIs saved to C:\Users\psycl\Documents\GitHub\WGU_CS_Capstone\models\model_kpis.pkl
Validation Accuracy: 0.8435
Precision: 0.8487
Recall: 0.8435
F1 Score: 0.8440
```

Conclusion:

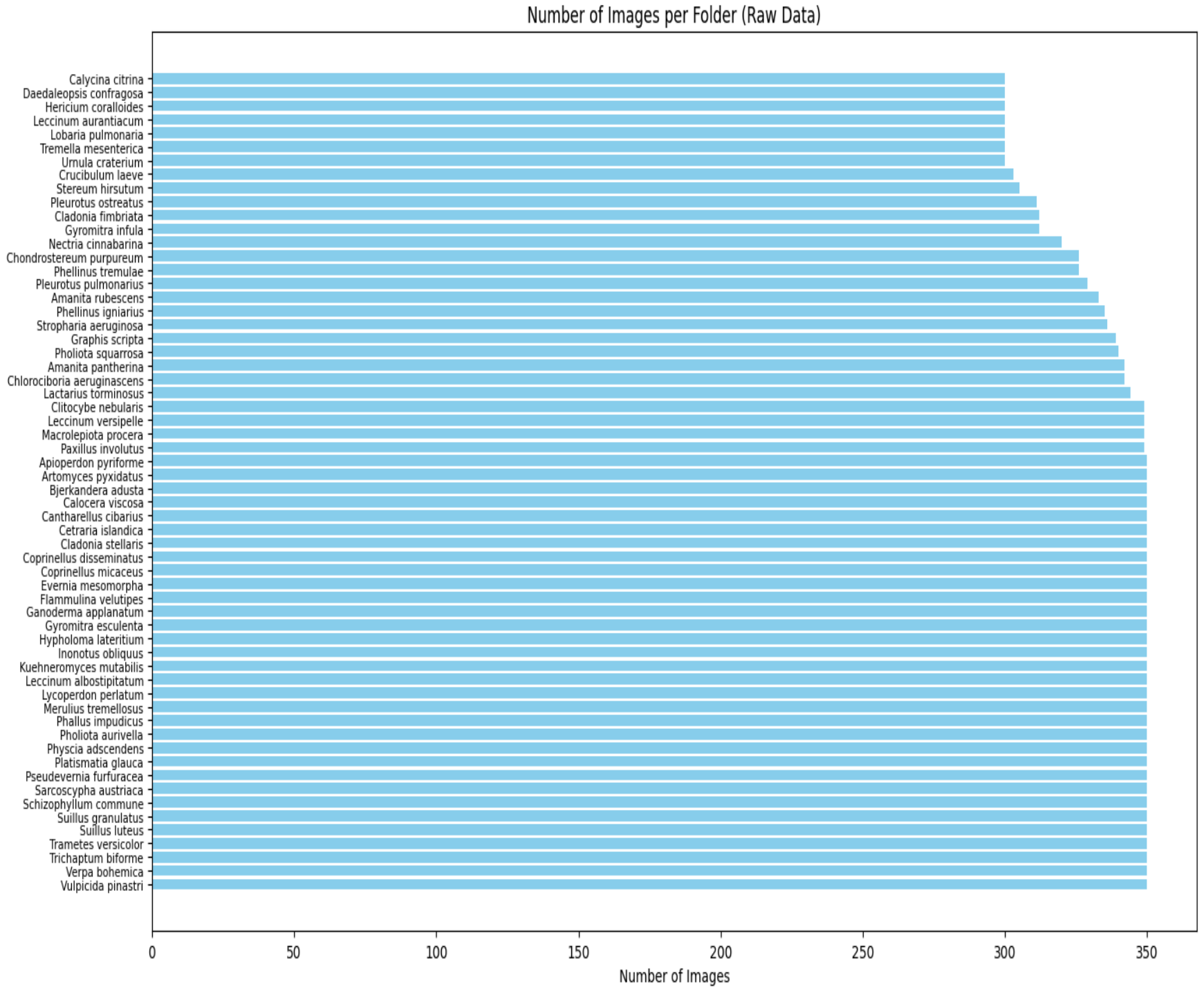
The results have exceeded the hypothesized predictions. All values are above the hypothesized thresholds of every key performance indicator (KPI), therefore the hypotheses have met the acceptance criteria.

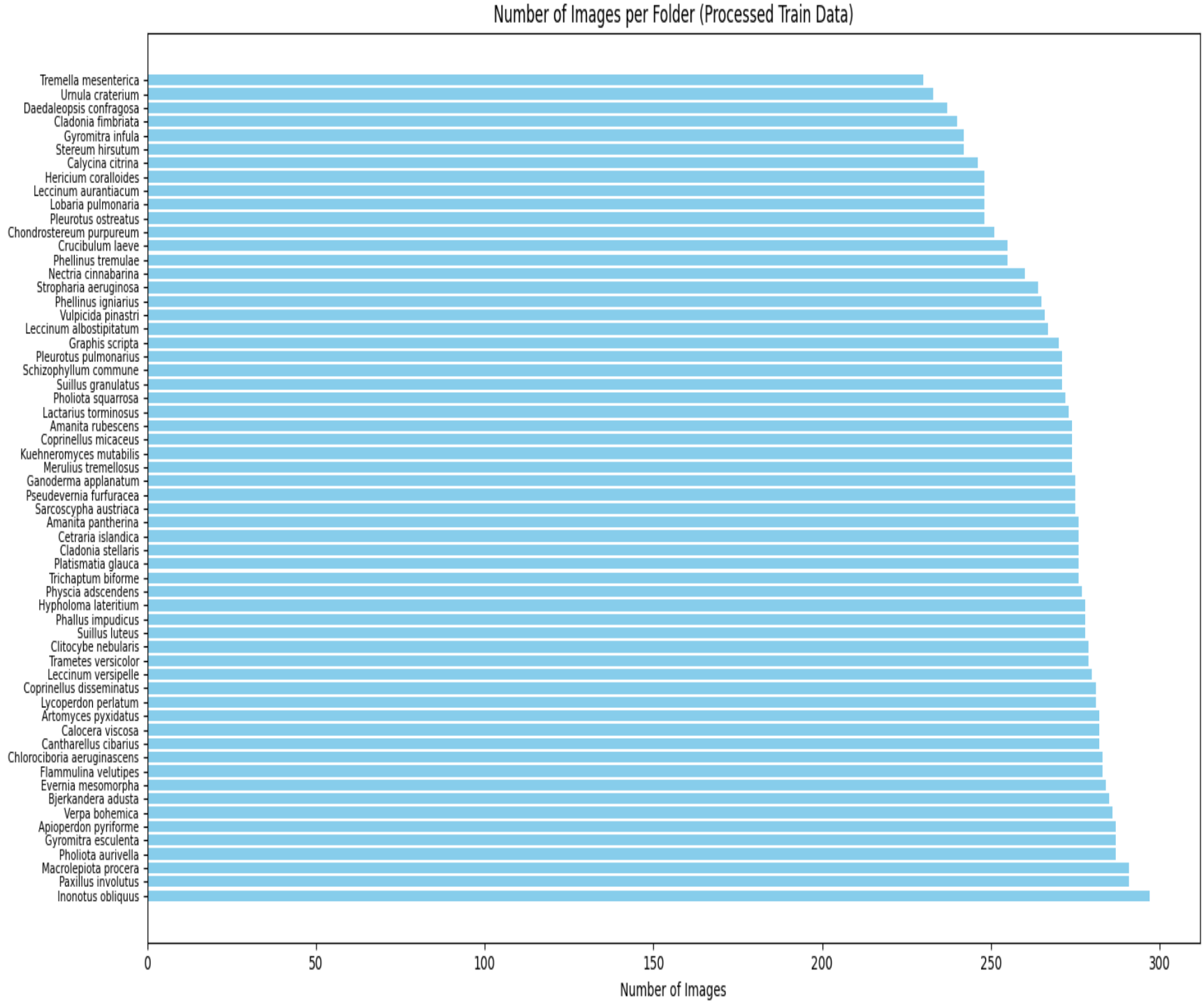
Data Exploration and Preparation

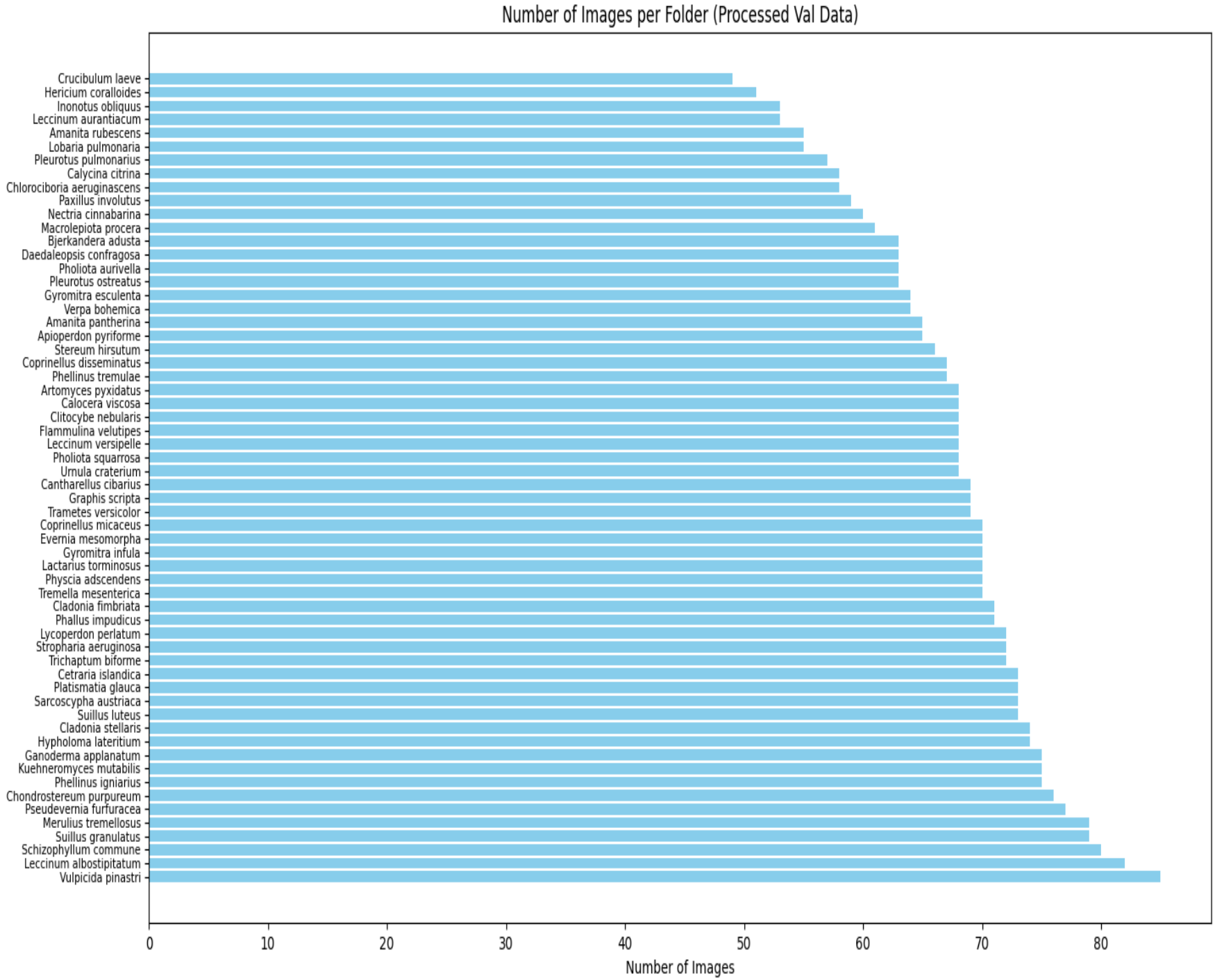
Sample Images from Mushroom Dataset:

Data Distribution

The raw dataset comprises 60 different species of mushrooms and a total of 20,251 images. Each mushroom has at least 300 images and no more than 350 images for an even class distribution. This dataset was modified in order to balance the classes. The original dataset totalled ~50,000 across ~100 classes. Below are histograms of the balanced classes from the raw dataset to the 80/20 split preprocessed datasets (80% train and 20% val). Class balancing ensures the model will not outweigh data to the dominant species (has the majority of images) of mushrooms.


Number of Images per Folder (Raw Data)
Calycina citrina
Daedaleopsis confragosa
Hericium coralloides
Leccinum aurantiacum
Lobaria pulmonaria
Tremella mesenterica
Urnula craterium
Crucibulum laeve
Stereum hirsutum
Pleurotus ostreatus
Cladonia fimbriata
Gyromitra infula
Nectria cinnabarina
Chondrostereum purpureum
Phellinus tremulae
Pleurotus pulmonarius
Amanita rubescens
Phellinus igniarius
Stropharia aeruginosa
Graphis scripta
Pholiota squarrosa
Amanita pantherina
Chlorociboria aeruginascens
Lactarius torminosus
Clitocybe nebularis
Leccinum versipelle
Macrolepiota procera
Paxillus involutus
Apioperdon pyriforme
Artomyces pyxidatus
Bjerkandera adusta
Calocera viscosa
Cantharellus cibarius
Cetraria islandica
Cladonia stellaris
Coprinellus disseminatus
Coprinellus micaceus
Evernia mesomorpha
Flammulina velutipes
Ganoderma applanatum
Gyromitra esculenta
Hypholoma lateritium
Inonotus obliquus
Kuehneromyces mutabilis
Leccinum albostipitatum
Lycoperdon perlatum
Merulius tremellosus
Phallus impudicus
Pholiota aurivella
Physcia adscendens
Platismatia glauca
Pseudevernia furfuracea
Sarcoscypha austriaca
Schizophyllum commune
Suillus granulatus
Suillus luteus
Trametes versicolor
Trichaptum biforme
Verpa bohemica
Vulpicida pinastri
0
50
100
150
200
250
300
350
Number of Images

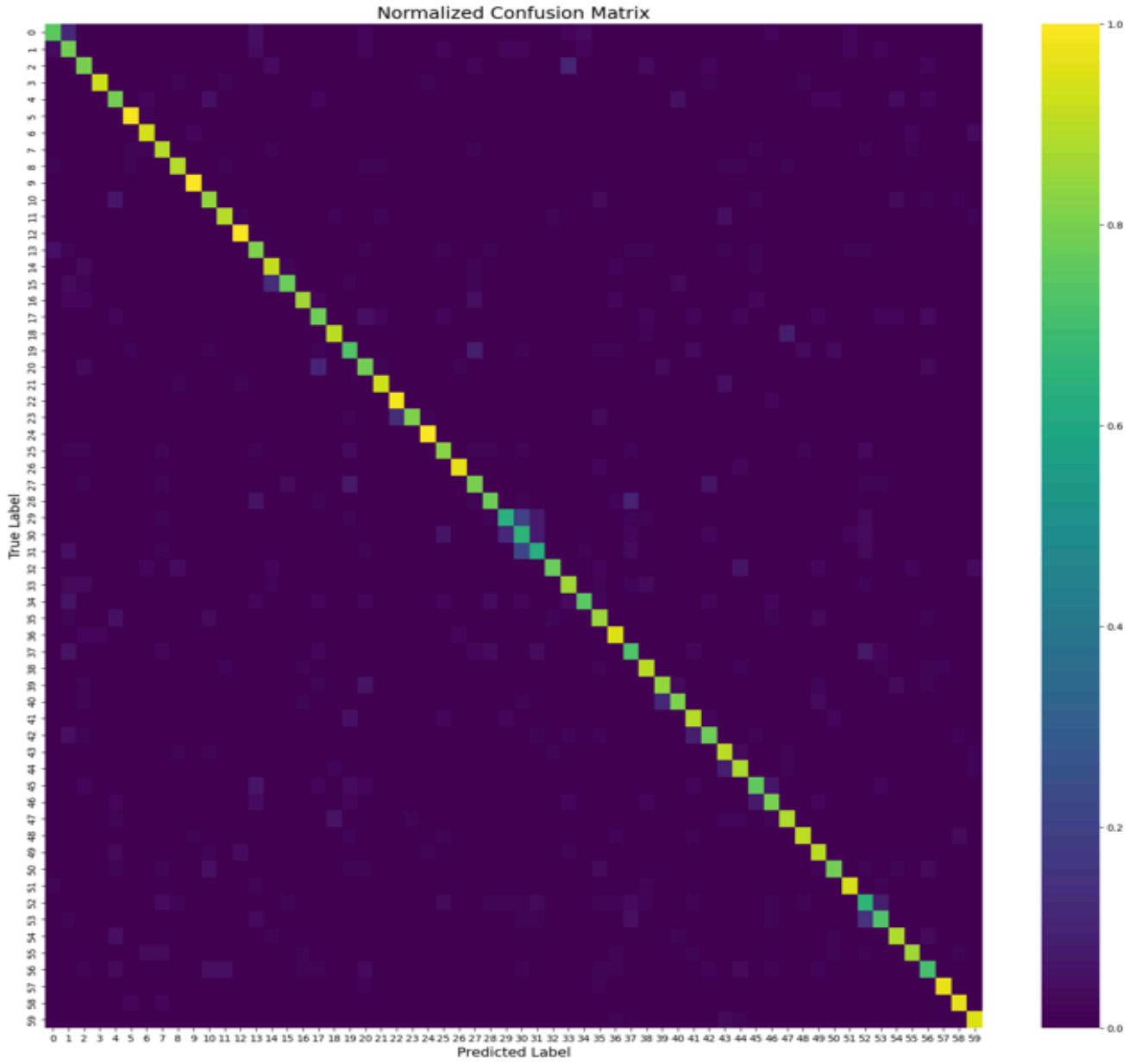

Number of Images per Folder (Processed Train Data)
Tremella mesenterica
Urnula craterium
Daedaleopsis confragosa
Cladonia fimbriata
Gyromitra infula
Stereum hirsutum
Calycina citrina
Hericium coralloides
Leccinum aurantiacum
Lobaria pulmonaria
Pleurotus ostreatus
Chondrostereum purpureum
Crucibulum laeve
Phellinus tremulae
Nectria cinnabarina
Stropharia aeruginosa
Phellinus igniarius
Vulpicida pinastri
Leccinum albostipitatum
Graphis scripta
Pleurotus pulmonarius
Schizophyllum commune
Suillus granulatus
Pholiota squarrosa
Lactarius torminosus
Amanita rubescens
Coprinellus micaceus
Kuehneromyces mutabilis
Merulius tremellosus
Ganoderma applanatum
Pseudevernia furfuracea
Sarcoscypha austriaca
Amanita pantherina
Cetraria islandica
Cladonia stellaris
Platismatia glauca
Trichaptum biforme
Physcia adscendens
Hypholoma lateritium
Phallus impudicus
Suillus luteus
Clitocybe nebularis
Trametes versicolor
Leccinum versipelle
Coprinellus disseminatus
Lycoperdon perlatum
Artomyces pyxidatus
Calocera viscosa
Cantharellus cibarius
Chlorociboria aeruginascens
Flammulina velutipes
Evernia mesomorpha
Bjerkandera adusta
Verpa bohemica
Apioperdon pyriforme
Gyromitra esculenta
Pholiota aurivella
Macrolepiota procera
Paxillus involutus
Inonotus obliquus
0
50
100
150
200
250
300
Number of Images


Number of Images per Folder (Processed Val Data)
Crucibulum laeve
Hericium coralloides
Inonotus obliquus
Leccinum aurantiacum
Amanita rubescens
Lobaria pulmonaria
Pleurotus pulmonarius
Calycina citrina
Chlorociboria aeruginascens
Paxillus involutus
Nectria cinnabarina
Macrolepiota procera
Bjerkandera adusta
Daedaleopsis confragosa
Pholiota aurivella
Pleurotus ostreatus
Gyromitra esculenta
Verpa bohemica
Amanita pantherina
Apioperdon pyriforme
Stereum hirsutum
Coprinellus disseminatus
Phellinus tremulae
Artomyces pyxidatus
Calocera viscosa
Clitocybe nebularis
Flammulina velutipes
Leccinum versipelle
Pholiota squarrosa
Urnula craterium
Cantharellus cibarius
Graphis scripta
Trametes versicolor
Coprinellus micaceus
Evernia mesomorpha
Gyromitra infula
Lactarius torminosus
Physcia adscendens
Tremella mesenterica
Cladonia fimbriata
Phallus impudicus
Lycoperdon perlatum
Stropharia aeruginosa
Trichaptum biforme
Cetraria islandica
Platismatia glauca
Sarcoscypha austriaca
Suillus luteus
Cladonia stellaris
Hypholoma lateritium
Ganoderma applanatum
Kuehneromyces mutabilis
Phellinus igniarius
Chondrostereum purpureum
Pseudevernia furfuracea
Merulius tremellosus
Suillus granulatus
Schizophyllum commune
Leccinum albostipitatum
Vulpicida pinastri
0
10
20
30
40
50
60
70
80
Number of Images

Data Analysis

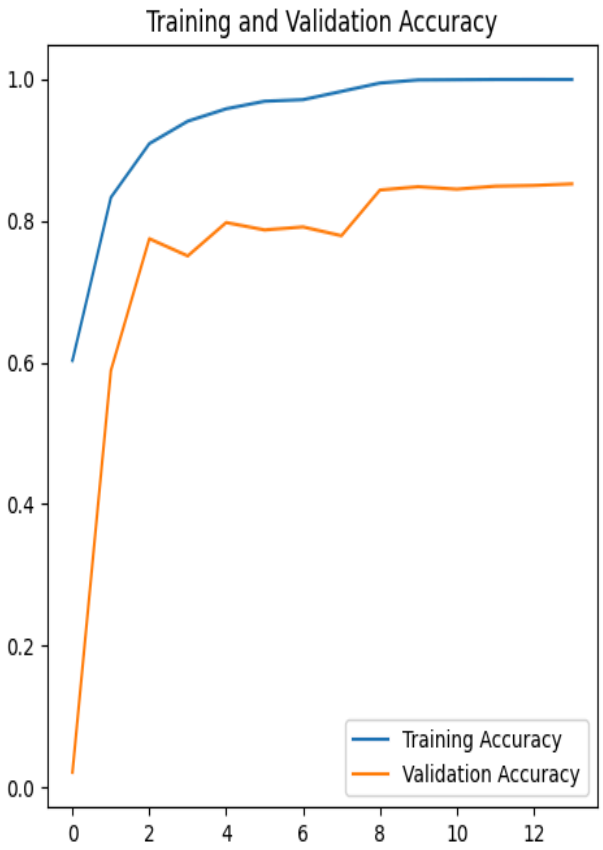
Confusion Matrix

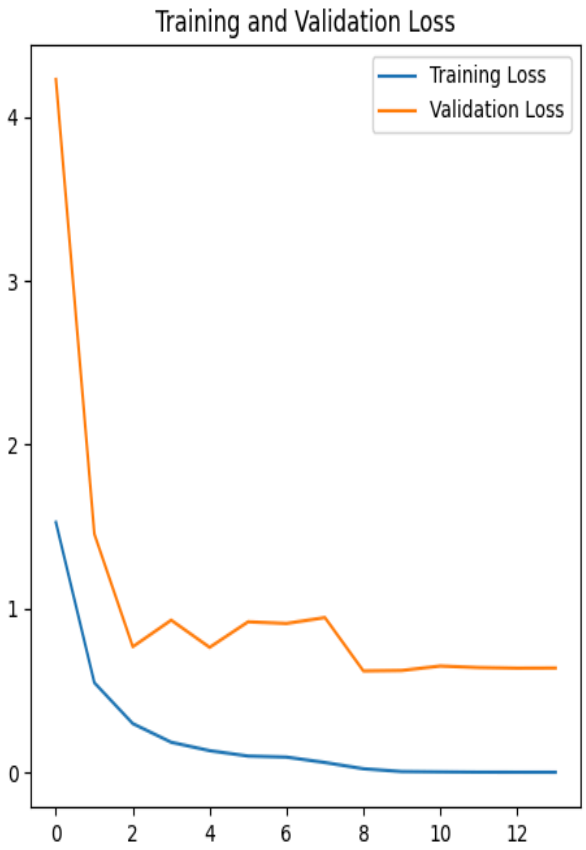
After training the model, we can see the performance of the model through a confusion matrix heatmap. Below is a confusion matrix that has all 60 classes that represent true and predicted classes. The model demonstrates high accuracy as the diagonal elements, which represent correctly predicted classes, show high values. The misclassifications (off the diagonal) show low values, which indicates that the model is performing well with a high percentage of correct predictions and low percentages of outliers.



Training and Validation Accuracy/Loss:

Below are two graphs that depict the Training and Validation Accuracy and Training and Validation Loss. This visualization is to see how well the model is performing over each epoch. This will help us discover any potential overfitting or underfitting in the data. As we can see from both the training and validation accuracy and loss that the model is training effectively and does not suggest any over or under fitting. The training accuracy shows a steady increase while the validation accuracy increases slightly and then maintains.The training loss steadily decreases and the validation loss remains relatively stable. This is an indication that the model is well-generalized.

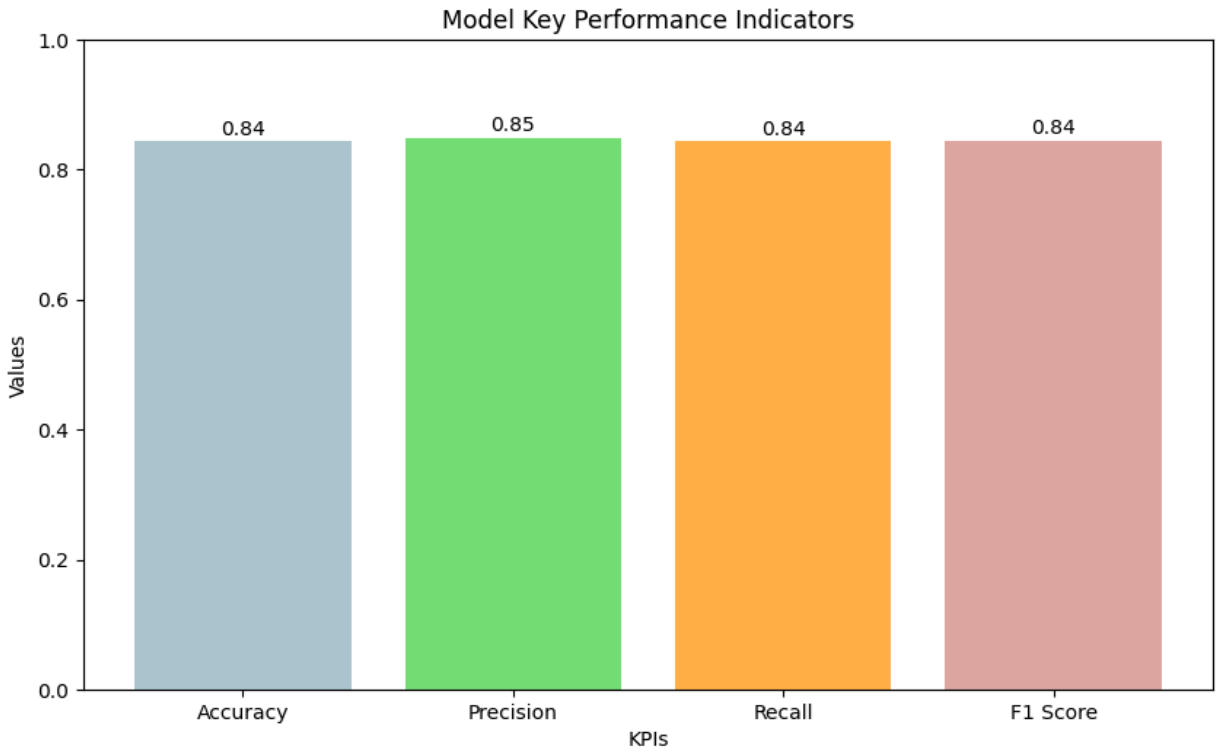




Data Summary

Model Metrics Summary

We have decided to use overall accuracy, precision, recall, and F1 score in order to evaluate the efficacy of the model. Below is a bar chart of each KPI. All KPIs are above 80% which indicates an effective model with high predictability.



Assessment of the Product's Accuracy

Model Performance Metrics

We used training accuracy, validation accuracy, precision, recall, and F1 score in order to evaluate the model. These metrics provide a comprehensive view of the model's ability to identify mushroom types from images accurately.

Results:

- Epochs: 14/50 (training ended after the 14th Epoch since implementing early stopping)
- Training Accuracy: 99.96%
- Validation Accuracy: 84.35%
- Validation Loss: 0.6386
- Precision: 84.87%
- Recall: 84.35%
- F1 Score: 84.40%

The model demonstrates a high training accuracy which indicates it learned the training data well. It also exceeded hypothesized values in validation accuracy, precision, recall, and F1 score, confirming that the model is robust and effective.

User Feedback

In addition to offering a prediction with a confidence percentage, the application features a user feedback mechanism. This allows the user to confirm whether or not the prediction was accurate. This is to be used in early stages by mycologist experts in order to further prove the efficacy of mushroom identification accuracy.

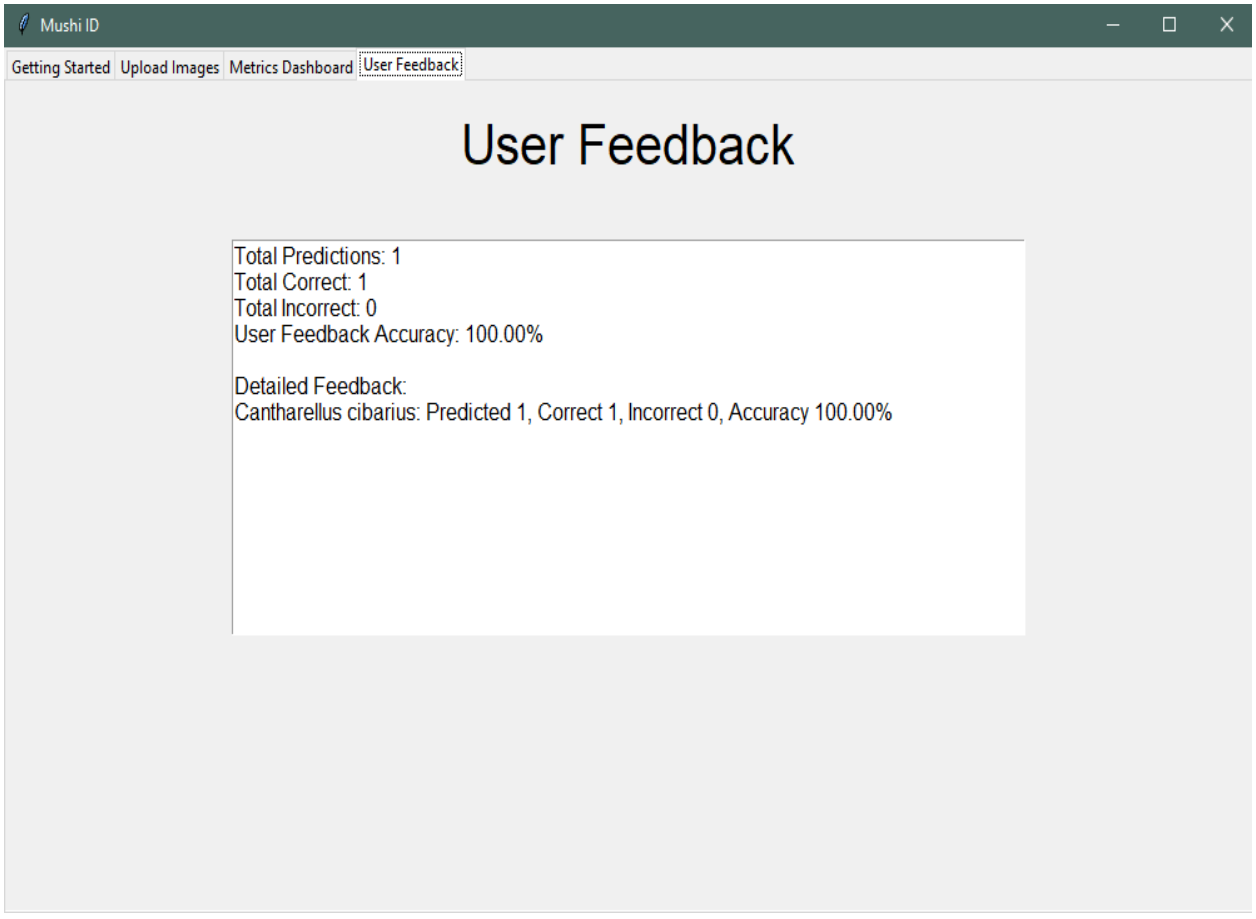
User Feedback Implementation:

- Users can select “Yes” or “No” after an image has been uploaded and a prediction has been made.
- This feedback is then logged into a CSV file with the mushroom type and how often the mushroom has been identified correctly or incorrectly.
- In the User Feedback tab in the application, a user can see overall user feedback accuracy and detailed feedback for each mushroom type.

User Feedback Metrics:

- Total Predictions: The total number of predictions made by the model.
- Total Correct Predictions: The number of predictions marked as correct by the users.
- Total Incorrect Predictions: The number of predictions marked as incorrect by the users.
- User Feedback Accuracy: Calculated as (Total Correct/Total Predictions) * 100, represented as a percentage.

User feedback coupled with the model’s own prediction confidence will provide a way to continuously monitor and analyze user feedback in order to gain insights into the model’s real-world performance. This will ensure the model remains accurate and reliable over time.



Data Product Testing Results

Initial data product testing focused on a custom CNN. These results were indicative of overfitting and required a revision to the model architecture. Using transfer learning techniques that allow us to use a pre-trained model “that was trained on a large benchmark dataset to solve a problem similar to the one that we want to solve”(Marcelino, 2018), we were able to achieve better accuracy results.

This led to the use of ResNet-50 as a supplement pre-trained model. Initially the use of ResNet-50 alone yielded high accuracy, precision, recall, and F1 values, however, had a low prediction confidence and was not accurately predicting mushroom types from known mushroom images.

Early results with ResNet-50 implementation:

- Validation Accuracy: 0.8545
- Precision: 0.8645
- Recall: 0.8545
- F1 Score: 0.8562

calycina_citrina_test_data.jpg most likely belongs to 'Daedaleopsis confragosa' with a confidence of 0.02 percent.

When the results were not satisfactory, further data pre-processing strategies were tested to yield better results. In order to further enhance the model’s ability to better distinguish different mushroom images, we leveraged Local Binary Pattern (LBP) and Canny edge detection.

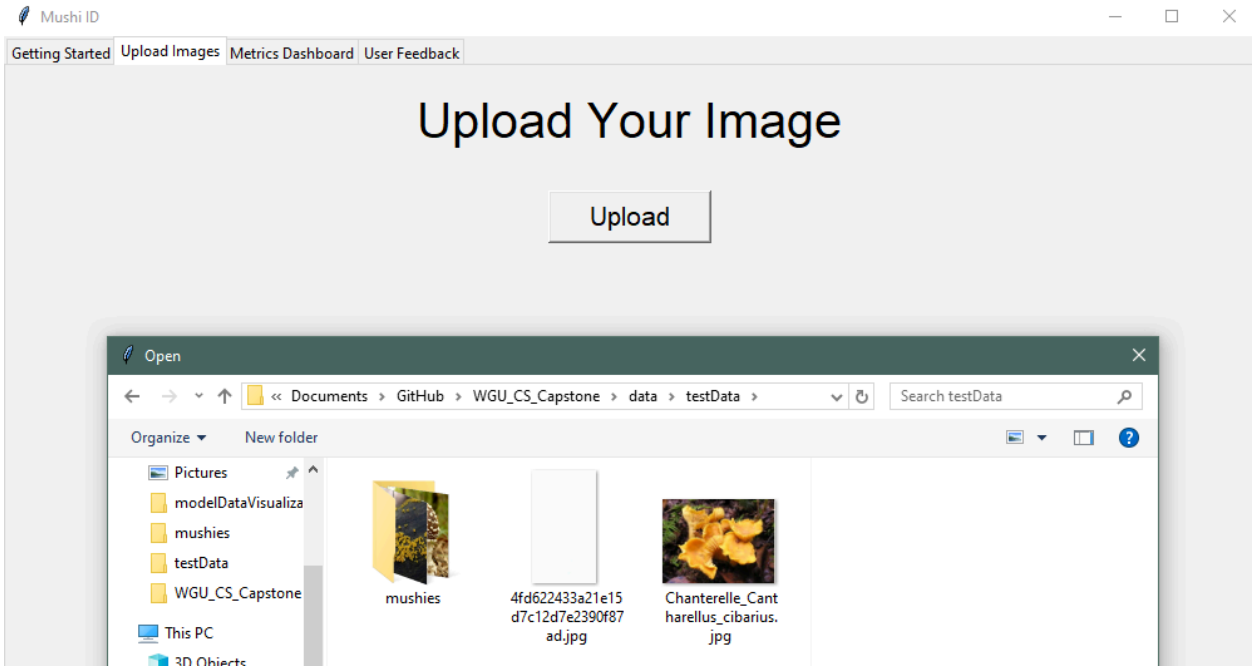
Local Binary Pattern, defined in the scikit-image library documentation, is a method used to classify textures. This technique helps capture the texture details of mushroom images in order to differentiate between various species (Local Binary Pattern for Texture Classification — Skimage V0.19.2 Docs, n.d.).

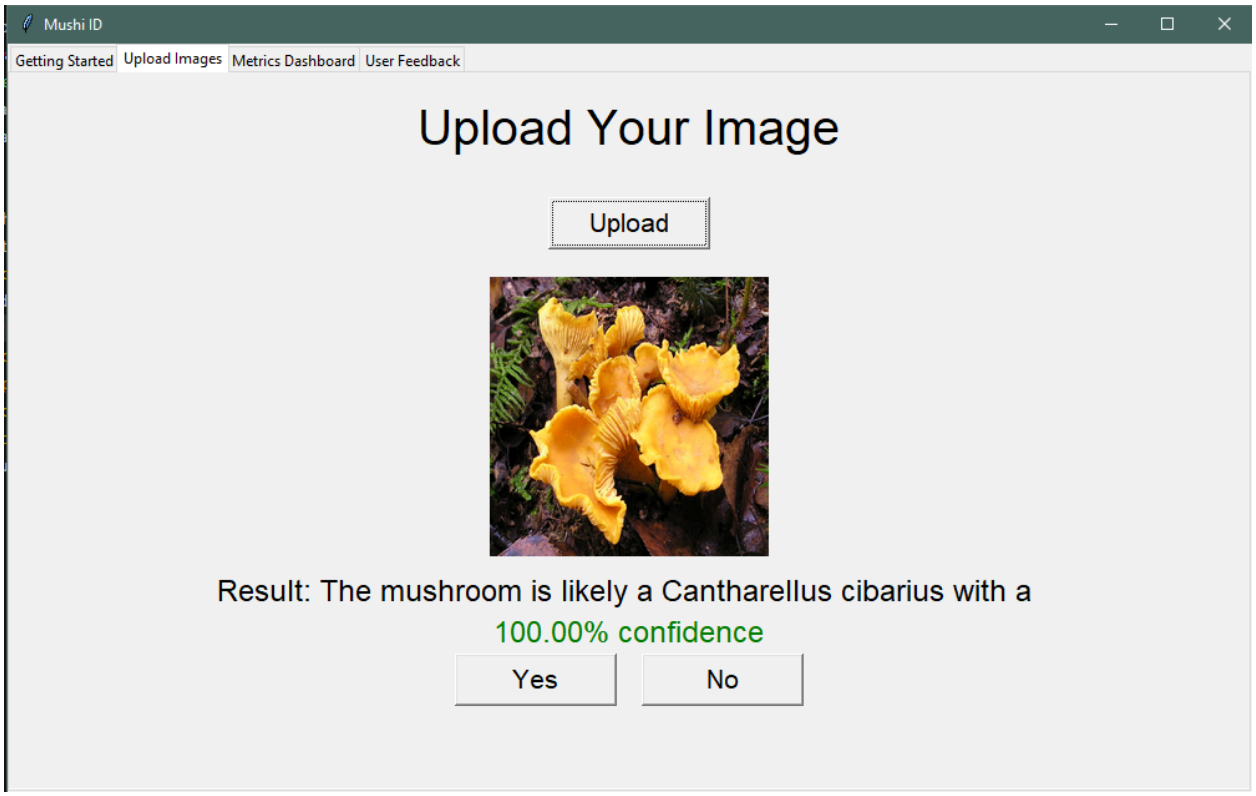
According to GeeksForGeeks (2021), the “Canny() Function in OpenCV is used to detect the edges in an image.” This will significantly help with detecting the outlines of objects within the image that will more accurately identify a mushroom from another.

Here are the results after LBP and Canny edge detection was applied:

- Training Accuracy: 99.96%
- Validation Accuracy: 84.35%
- Validation Loss: 0.6386
- Precision: 84.87%
- Recall: 84.35%
- F1 Score: 84.40%

(See screenshots below for model prediction confidence within the application)





Source code and executable file(s)

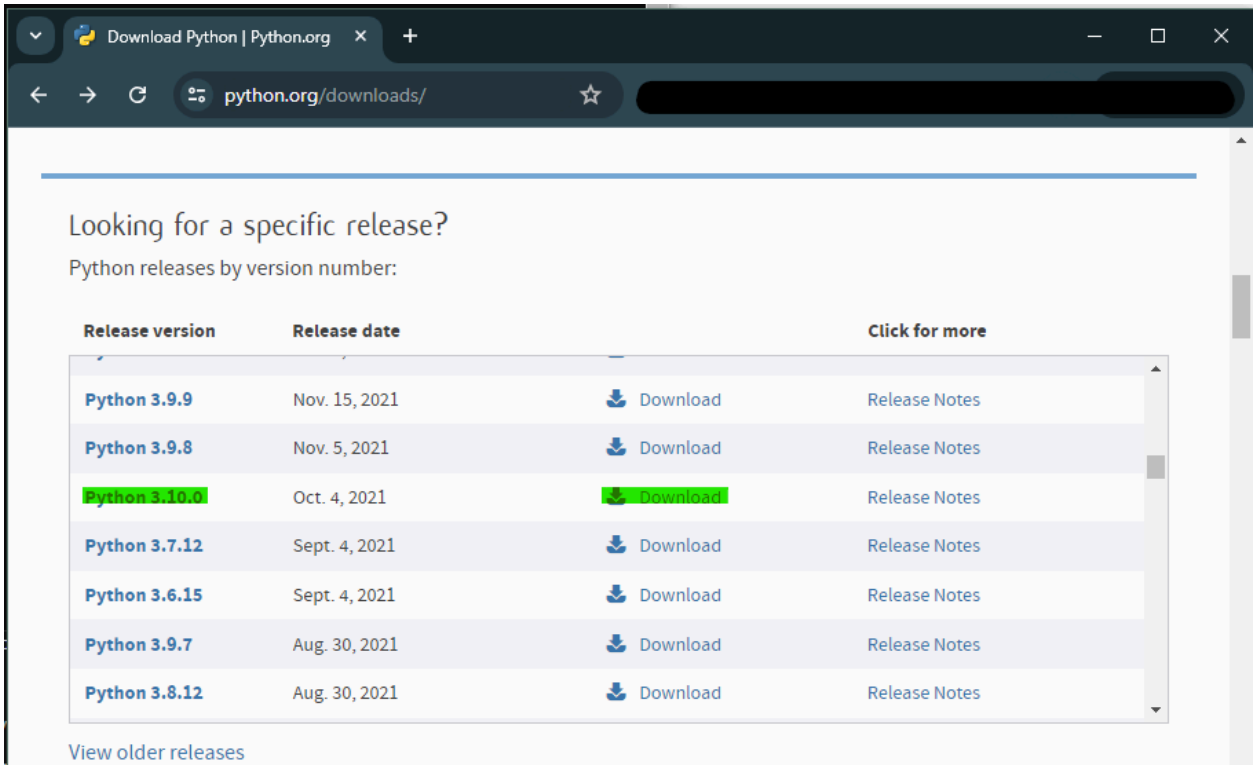
- Source code can be found: WGU_CS_Capstone\src

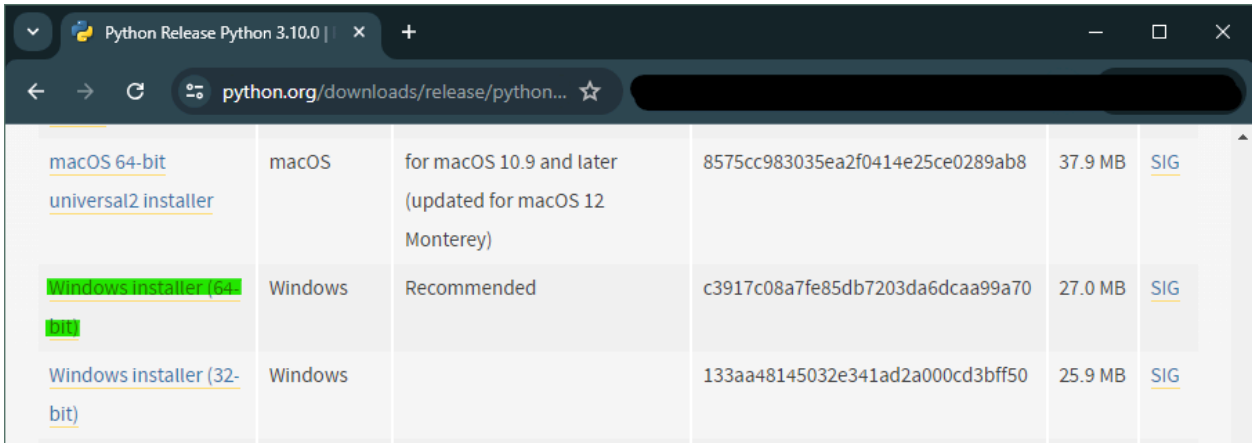
Quick Start Guide

Mushi ID is a python desktop application that will run seamlessly on Windows OS. Please follow the instructions below.

1. Download Python

First, start by downloading Python version 10.1 for Windows (64-bit) directly from the python website https://www.python.org/downloads/.





2. Necessary Python Libraries to Download

Next you'll need to download the versions of the libraries below:

- TensorFlow 2.9.1 (This must be 2.9.1, newer versions of TensorFlow/Keras are incompatible)
- NumPy 1.26.4
- Pillow 10.4.0
- OpenCV-Python 4.10.0.84
- Scikit-Image 0.24.0
- Scikit-Learn 1.5.0

The libraries above can be installed via ‘pip’ in the command line.

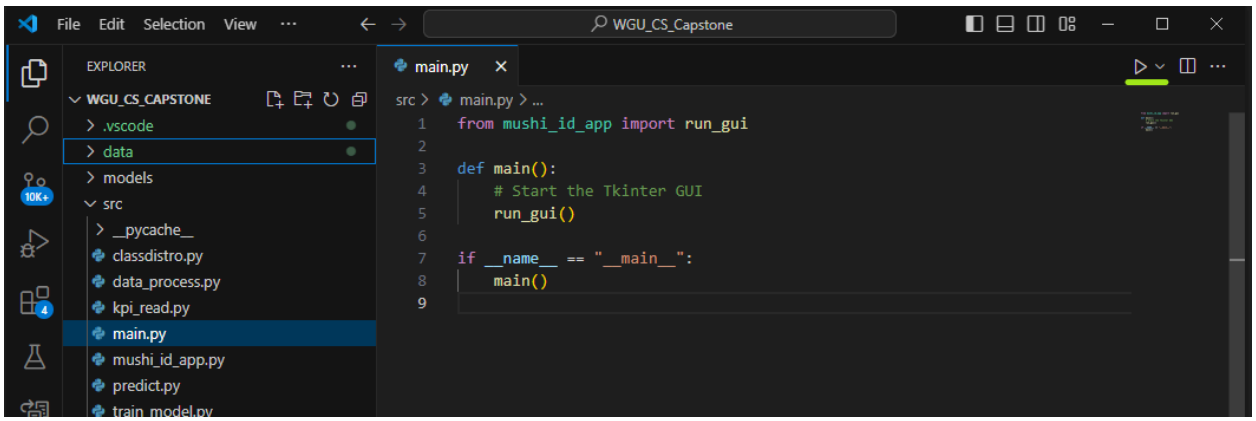
```
pip install tensorflow==2.9.1 numpy==1.26.4 Pillow==10.4.0
opencv-python==4.10.0.84 scikit-image==0.24.0 scikit-learn==1.5.0
```

3. Running the Program

Once all of the necessary libraries are installed, you can navigate to WGU_CS_Capstone/src directory. From here you can either open the main.py file in Visual Studio Code or run directly from the command line.

Visual Studio Code:

1. Open main.py in a new window of Visual Studio Code.
2. From the top right corner select the ‘Play’ button to run from the VS Code terminal.



Command Line:

1. Open a new command line prompt by selecting the Windows start button and searching for “Command Prompt” or “cmd”
2. Navigate to the WGU_CS_Capstone\src folder with the following command:

```
cd path/to/WGU_CS_Capstone/src
```

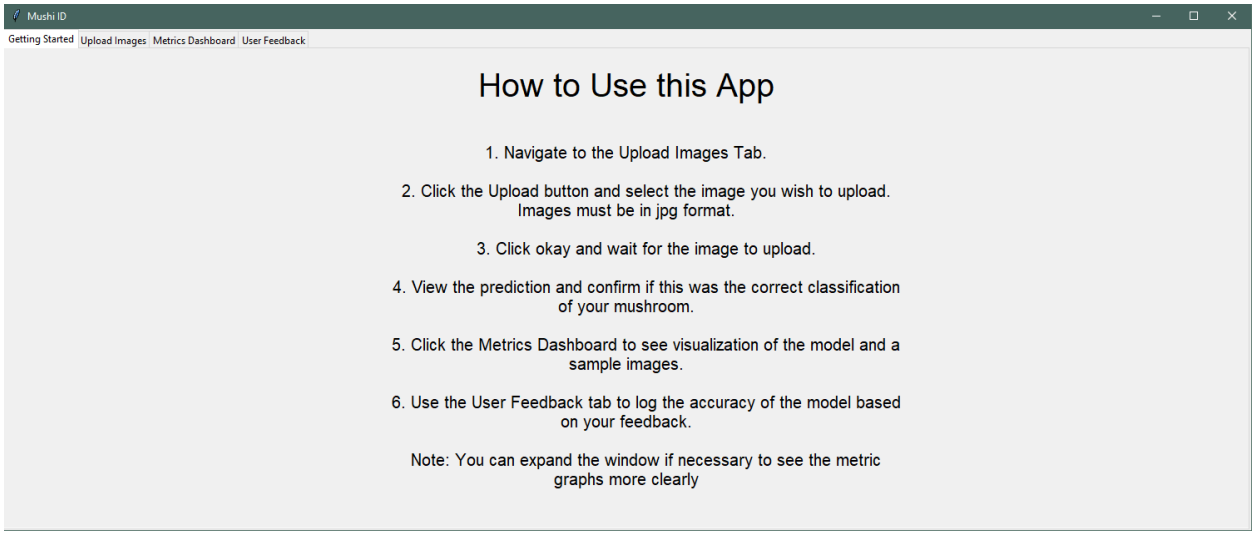
*replace ‘path/to/’ with your directory in which WGU_CS_Capstone/src is saved

3. Then run the main.py script with the following command:

```
python main.py
```

4. How to Use this App Tab

After the program is run, you will begin on the ‘How to Use this App’ tab that has instructions on how to use this application.



5. Upload Images Tab

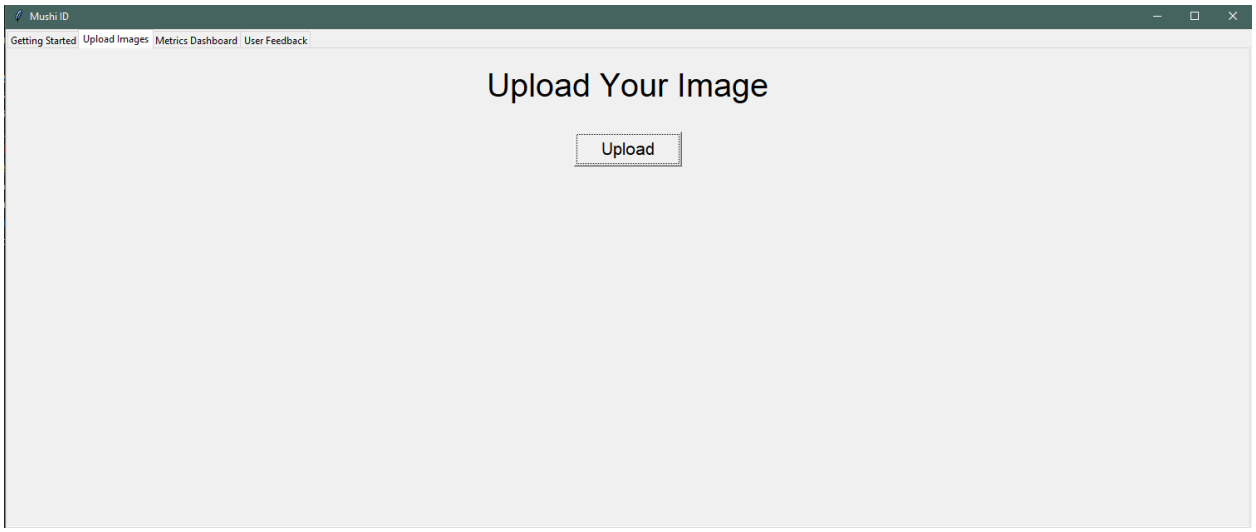
The Upload Images tab is where you will upload images and receive model predictions to identify your chosen mushroom images.

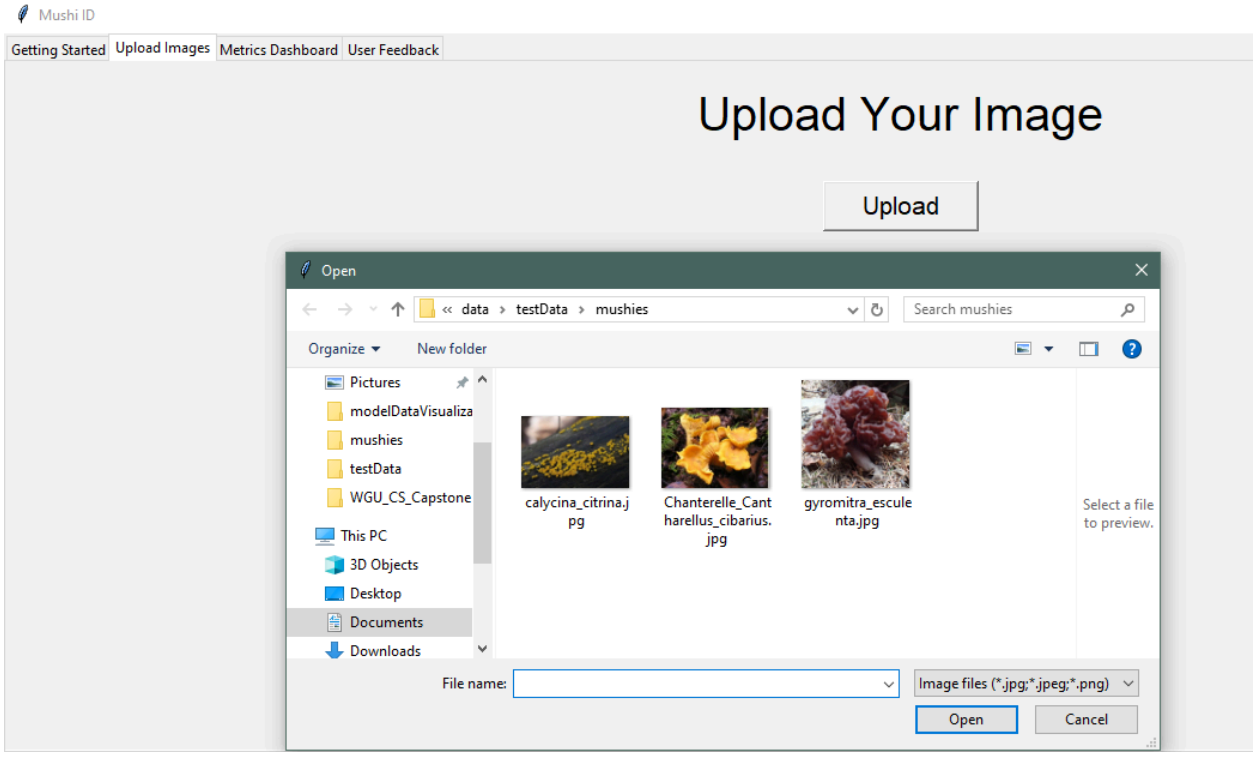
1. Click the ‘Upload’ button and select either your own image (please note the mushroom classes you can choose from below), or one from the testData folder (see below).
2. Wait for the image to process and view the model’s prediction and its prediction confidence.
3. If you know with certainty whether or not the model has predicted the correct or incorrect mushroom type, you may then select either the ‘Yes’ or ‘No’ button to log your feedback.
4. If you would like to identify another mushroom, you can simple click ‘Upload’ again to receive another prediction.

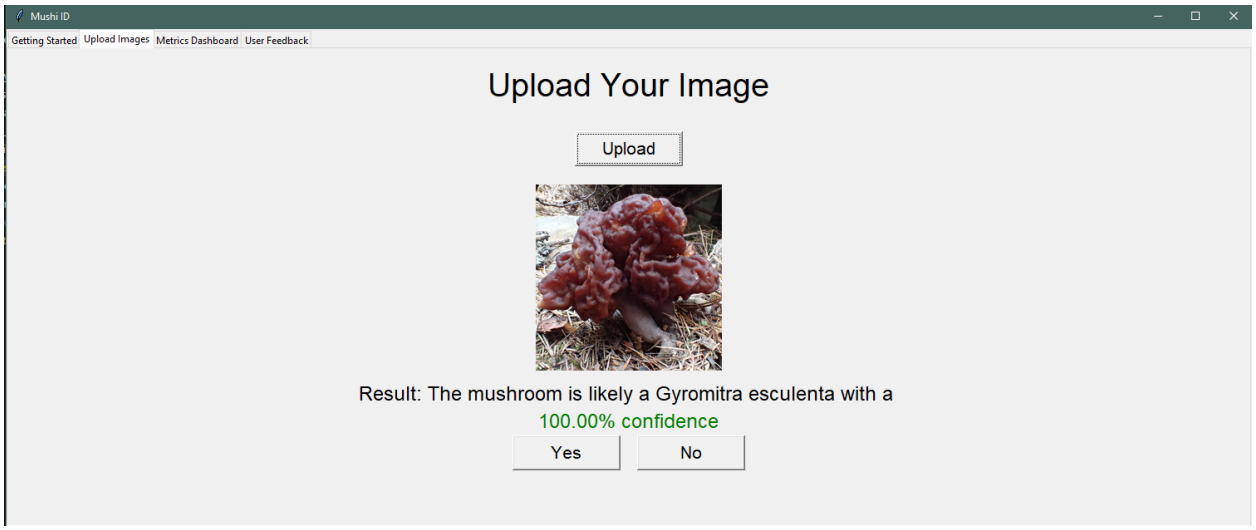
Test Data

If you do not have your own image to upload, there are some images that you can try out the application with located in the WGU_CS_Capstone\data\testData\mushies directory. You can also try the negativeTestImages to see how well the model does with an image that is clearly not a mushroom.

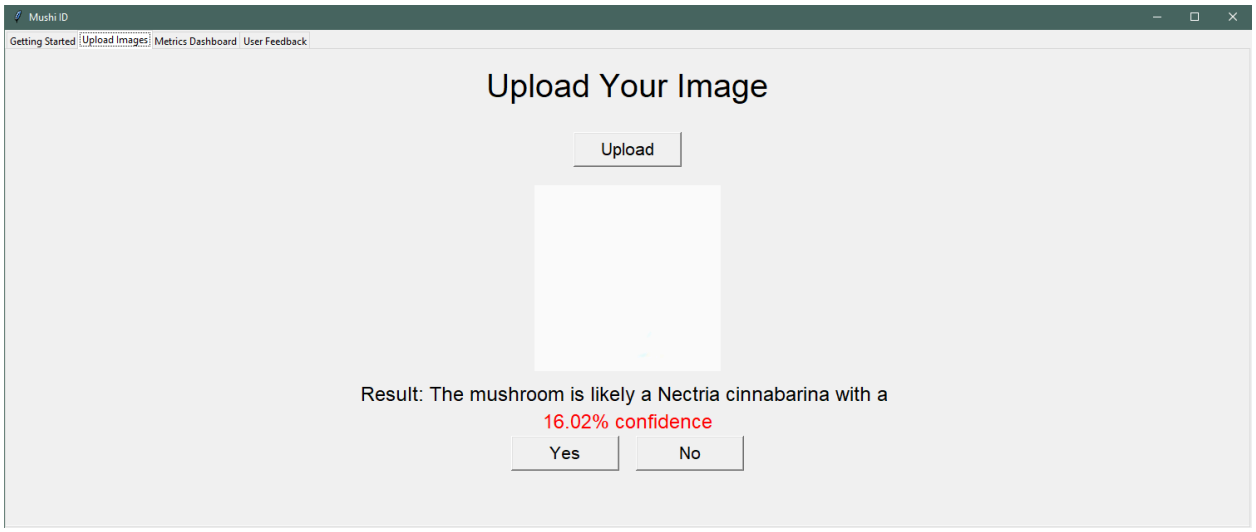
Note: The testData images were NOT sourced from the original datasets, therefore they are new to the model and are authentic predictions.


Mushi ID
Getting Started
Upload Images
Metrics Dashboard
User Feedback
Upload Your Image
Upload


Mushi ID
Getting Started
Upload Images
Metrics Dashboard
User Feedback
Upload Your Image
Upload
Open
data › testData › mushies
Search mushies
Organize
New folder
Pictures
modelDataVisualiza
mushies
testData
WGU_CS_Capstone
This PC
3D Objects
Desktop
Documents
Downloads
calycina_citrina.jpg
Chanterelle_Cantharellus_cibarius.jpg
gyromitra_esculenta.jpg
Select a file to preview.
File name:
Image files (*.jpg;*.jpeg;*.png)
Open
Cancel


Mushi ID
Getting Started
Upload Images
Metrics Dashboard
User Feedback
Upload Your Image
Upload
Result: The mushroom is likely a Gyromitra esculenta with a
100.00% confidence
Yes
No

Negative test expected results should reflect a low model prediction confidence percentage.



Mushroom Class List

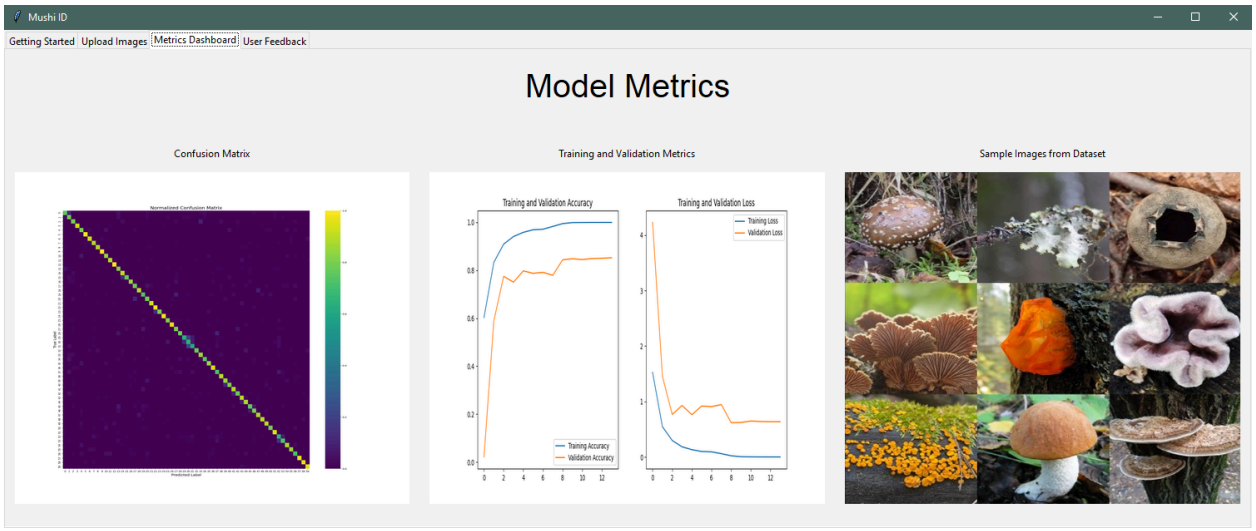
These are the mushroom classes that the model was trained on, you may try to use other mushroom images not present on this list, but the accuracy will not be correct.

Calycina citrina	Daedaleopsis confragosa	Hericium coralloides	Leccinum aurantiacum	Lobaria pulmonaria	Tremella mesenterica
Urnula craterium	Crucibulum laeve	Stereum hirsutum	Pleurotus ostreatus	Cladonia fimbriata	Gyromitra infula
Nectria cinnabarina	Chondrostereum purpureum	Phellinus tremulae	Pleurotus pulmonarius	Amanita rubescens	Phellinus igniarius
Stropharia aeruginosa	Graphis scripta	Pholiota squarrosa	Amanita pantherina	Chlorociboria aeruginascens	Lactarius torminosus
Clitocybe nebularis	Leccinum versipelle	Macrolepiota procera	Paxillus involutus	Apioperdon pyriforme	Artomyces pyxidatus
Bjerkandera adusta	Calocera viscosa	Cantharellus cibarius	Cetraria islandica	Cladonia stellaris	Coprinellus disseminatus
Coprinellus micaceus	Evernia mesomorpha	Flammulina velutipes	Ganoderma applanatum	Gyromitra esculenta	Hypholoma lateritium
Inonotus obliquus	Kuehneromyces mutabilis	Leccinum albostipitatum	Lycoperdon perlatum	Merulius tremellosus	Phallus impudicus
Pholiota aurivella	Physcia adscendens	Platismatia glauca	Pseudevernia furfuracea	Sarcoscypha austriaca	Schizophyllum commune
Suillus granulatus	Suillus luteus	Trametes versicolor	Trichaptum biforme	Verpa bohemica	Vulpicida pinastri

6. Metrics Dashboard Tab

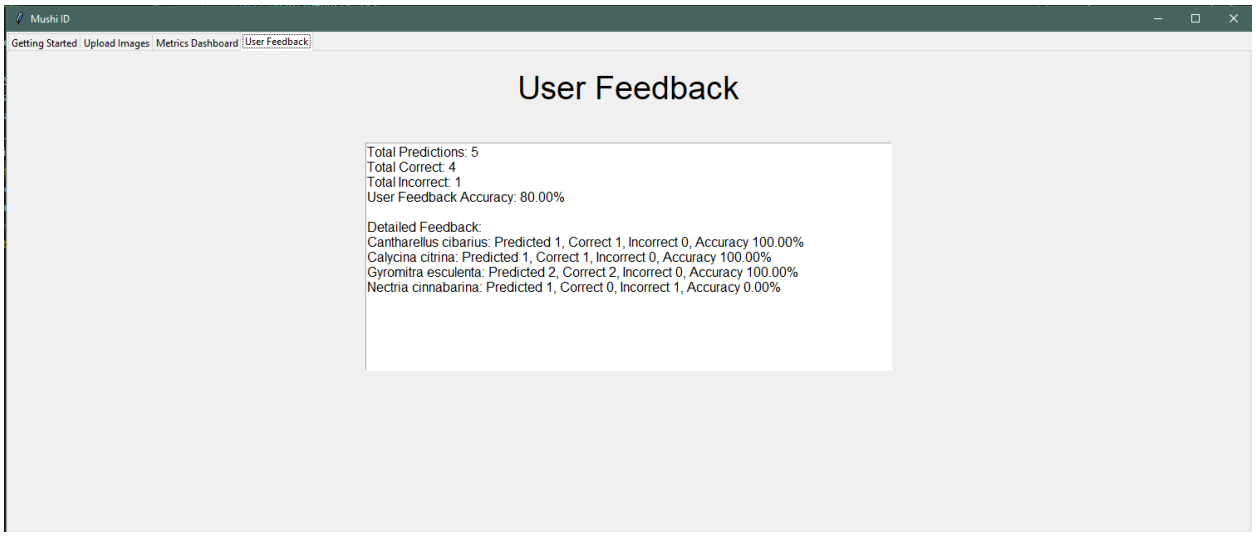
The Metrics Dashboard tab is designed for the user to see how well the model was performing during and after training. The tab also includes a sample image of the data for understanding the dataset variety.

The model metrics include a Confusion Matrix and Training and Validation Accuracy/Loss graphs over the span of training.



7. User Feedback Tab

The User Feedback tab displays the user feedback metrics. When a user submits feedback the prediction and whether or not it was correct is logged into a CSV file and uploaded on the User Feedback tab. This allows the user to analyze and maintain how well the model is actually doing in real-life situations.



Attribution for Data Usage

The mushroom species dataset used in this project was sourced from Kaggle and is licensed under the Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license.

- Dataset Source: https://www.kaggle.com/datasets/thehir0/mushroom-species
- Authors: Danil Kuchukov, Artyom Makarov, Damir Abdulayev, thehir0
- License: CC BY-NC 4.0

Modifications: Images were deleted to balance the classes.

References

Local Binary Pattern for texture classification — skimage v0.19.2 docs. (n.d.). Scikit-Image.org. https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html

Marcelino, P. (2018, October 23). Transfer learning from pre-trained models. Medium; Towards Data Science. https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

Potrimba, P. (2024, March 13). What is ResNet-50? Roboflow Blog. https://blog.roboflow.com/what-is-resnet-50/

Python OpenCV - Canny() Function. (2021, November 20). GeeksforGeeks. https://www.geeksforgeeks.org/python-opencv-canny-function/