

Final Report: Mushroom Edibility Classifier

Problem Introduction

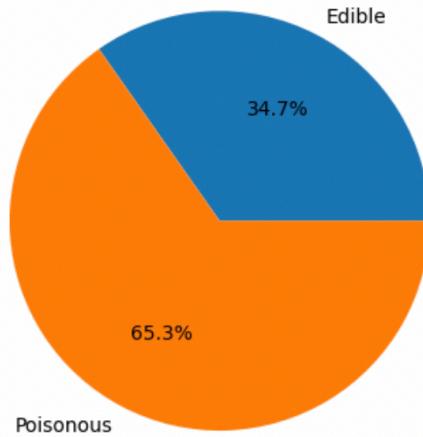
Much of the public opinion about wild mushrooms is that they are mostly poisonous and it's best to avoid them. Machine learning using image classification can provide reliable information about the toxicity potential of a mushroom using a digital picture.

In this project I used a pre-trained deep learning model from the Fastai library to predict a poisonous mushroom with 79% success. In addition, the model predicted 86% of all poisonous mushrooms in the dataset.

Data Wrangling and Exploratory Data Analysis

The raw image files for this project came from a [kaggle.com](#) user named Marcos Volpato which was built as part of his school's research project. The images were subset into four categories including edible sporocaps, edible mushroom sporocaps, poisonous sporocaps, and poisonous mushroom sporocaps. The first step in wrangling the dataset included renaming each image "poisonous" or "edible" depending on its directory name, and then regrouping the collection into the two classes.

To validate that each picture was neither corrupted or incomplete, I used Python Image Library's Image module and ran an iterated loop over each image with the verify method. All 3,400 images in the data passed the test. Moving forward with exploring the data, I compared the frequency of images in both classes:



About one third of the images are of edible mushrooms while the remaining two thirds are poisonous mushrooms.

Many images in the dataset were great quality that captured the top of the cap, the gills underneath, and the stem of the mushroom in question. For example:



This image is exceptional given that the mushroom to the right is still attached to the ground, and it validates the background habitat is exactly the area which this species grows in. However, from previously spot-checking other images in this dataset I observed a trend of watermarks and URL banners across the bottom edge. To investigate a solution for these, I experimented with another image:

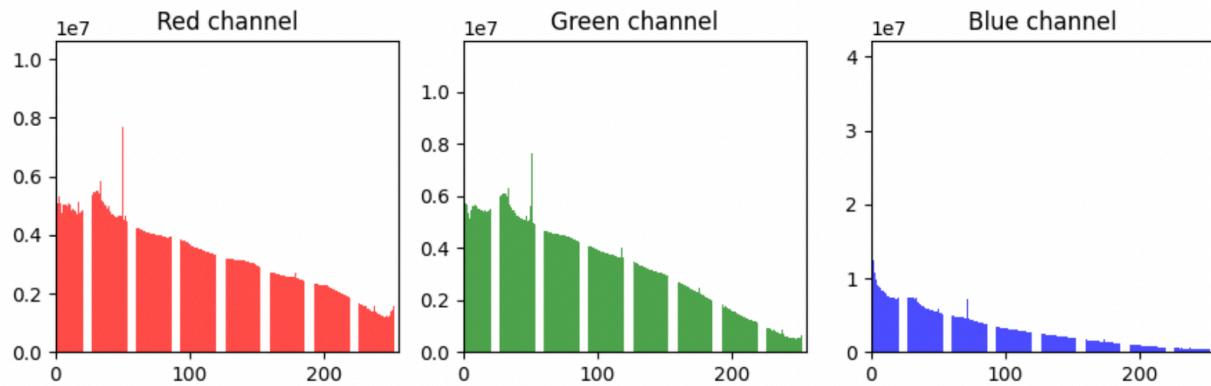


In this example, there are diagonal watermarks with the word “alamy” and a bar on the bottom edge with the text “alamy stock photo”. After I created a mask to cover each of those image issues, I used Sk-Image’s `inpaint_biharmonic` function to fill in those regions:



While this biharmonic_inpaint effort did help to remove some of the unrelated markups, I determined that the original image may be sufficient for my project. The mask corrections may take more important textures away from the image than the watermarks. Additional augmentation can be done after the model metrics had been revealed.

Exploring pixel intensities across RGB channels in a collection offers insights into the dataset's characteristics and quality. The distribution of pixel intensities within the RGB channels can reveal information about the lighting conditions, color balance, and potential biases within the dataset. Here are the results from all images in the dataset:



What's apparent is that the blue hues are a lot less prevalent compared to red and green. Also, there are some spikes in intensity within both the red and green channels, but it is missing from the blue channel.

Modeling

In preparation for modeling I converted all image file extensions to jpg, and subset the collection into training, validation, and testing groups. Next, I used Fastai's Datablock API to construct a Dataloaders object which serves as a container for the training dataloader and validation dataloader.

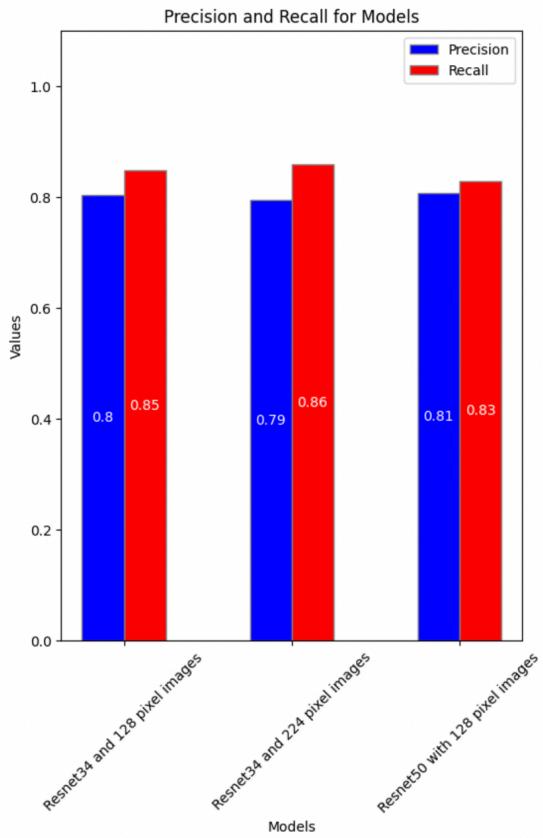
I trained and validated three models using Fastai's vision_learner model:

Model 1. This model used Resnet34 and resized images to 128 pixels, 4 epochs.

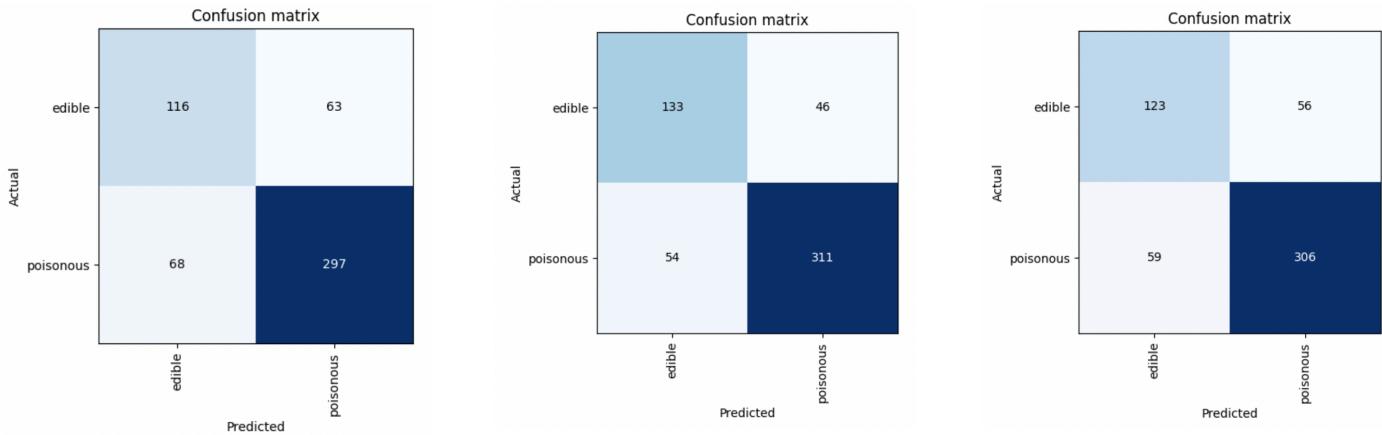
Model 2. This model used Resnet34 and resized images to 224 pixels, 4 epochs.

Model 3. This model used Resnet50 and resized images to 128 pixels, 4 epochs.

Following each model's train and validation step, I followed up with a precision and recall analysis on the test set:



In addition, I also computed the confusion matrices of each model:



Conclusions

If our goal is to have the best success over predictions of poisonous mushrooms (precision), the model with Resnet50 and 128 pixel images performed best with 81%. If our goal is to have the most coverage of correct predictions of poisonous mushrooms (recall), the model with Resnet34 and 224 pixel images performed best with 86%. Between the three confusion matrices, the second model performed the best with across all elements. In all, the second model is my choice for the top performer in my project.

Further Research

In an effort to improve my model's performance, more high quality pictures in the dataset of accurately labeled either poisonous or edible is the top of the list. When I mention high quality, I'm referring to a few properties of the image beyond resolution. The framing of the mushroom needs to include the top of the cap, the underside, and the stem. The background of the mushroom is also an important part of its habitat.

Another idea to explore is by using more powerful hardware on the existing data I've prepared. This includes using a GPU system to handle Resnet50 and 224 pixel image sizes. My current machine could not complete the task.

I'd also like to augment the existing dataset's watermark issues by exploring thresholding and edge detection with biharmonic_inpainting. This effort would need additional validation that the changes aren't removing important features of an image by mistaking them for markups. In addition, I'd like to try scaling the blue pixel distribution so that it follows a similar curve to that of the red and green channel distributions.