## 15-110 F26 - Machine Learning Mini-Project

## Name:

## AndrewID:

---

For this project, we will be working with the classic "chihuahua vs. blueberry muffin" image classification problem, a popular toy example in machine learning. Our goal is to train a model that can predict whether a given image shows a chihuahua or a muffin.



Most questions on this assignment allow you to use GenAI (unless stated otherwise). The goal is to ensure you understand how the code works and can explain it.

Start by downloading the following files from our website. Make sure the **.py** files and the **data folder** remain at the same directory level.

1. A zip file, **data.zip**, which contains the data for the project.
    a. Once you've downloaded **data.zip**, extract its contents (a **data** folder). It should have two subfolders, **training** and **testing,** which contain images.
2. A **`given.py`** file containing functions we have already written for you, which you will use in your implementation below.
    a. Note that this file will require you to install several external libraries: **pillow**, **numpy**, and **requests**.
3. A **`project2.py`** file, which you will complete. It includes comments to guide you through each section.

For this assignment there are two submission links on Gradescope.

- **Project2 - Reflection** - upload your responses to questions in this file as a .pdf
- **Project2 - Code and Process** - upload your code in the given **`project2.py`** starter file and a **.pdf** log of any communication you had with a GenAI. If you wrote the code yourself with no GenAI support, upload a short .pdf saying that.


Step 1: Featurize the Image Data
Step 2: Train/Test Split
Step 3: Train the Model
Step 4: Evaluate the Model
Step 5: Let's Predict!


Data source: https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification

## Step 1: Featurize the Image Data

In class, we discussed how the features for the seeds dataset were columns of the given dataset, such as *area, perimeter,* and *compactness*. We used these features to train the model so that we could then predict the type of kernel. In the seeds example, the features were given to us by the creators of the dataset, and they were also all numbers (numerical features).

What about when we work with images? We need a way to convert the images into features somehow. Specifically, we want to turn the images into some numerical form, and those numbers then will be used as our features.

How do we do that? There are multiple ways, some more complex and effective than others, which you would explore in more depth in a computer vision class.

For our purposes, we will use a simple, and maybe not super efficient approach. The method is called a **color histogram**, and it aims to compute summary statistics per image. A histogram is essentially a counting exercise: you scan every pixel, ask "which bucket does this value fall into?", and increment that bucket's counter. The final counts become the features.

We have already written this function for you in the **`given.py`** `file`, called `extract_color_histogram`. Read through the code (without using GenAI) and, in your own words, explain what the code is doing.

Now ask either GenAI or someone who is really good with code (like a TA, course instructor, or CS major friend) to explain the function to you line by line. Based on the new explanation, did your understanding of the code change? What was something new you learned?

| |
|---|
| |

Finally, we have also written a function called `load_one_folder` in **`given.py`**. Read the code and/or use GenAI to understand what it does, then answer the following questions.

| | |
|---|---|
| What does this function take as input? | |
| In 1-2 sentences what does this function do? | |
| What does this function return? | |

## Step 2: Train/Test Split

We have already split the training and test data into separate folders for you. You only need to load the training and test data into your code by following the steps below.

First, let's review your understanding of train/test splitting based on the lecture.

What happens to a ML model if you do not have a testing set?

Similarly, what happens to an ML model if your testing and training set are the same?

From the lecture, we used some code to do a 20% train-test split. Change that lecture code and write it below to set the testing set to be **35%** of the data.

Update **`project2.py`** with code that uses the functions we have already written for you in **`given.py`** to load the training and testing data. Fill in the blanks below with the code you write as well.

```
# load train data

X_train_chihuahua, y_train_chihuahua = _____________________________________________

X_train_muffin, y_train_muffin = _____________________________________________

# load test data

X_test_chihuahua, y_test_chihuahua = _____________________________________________

X_test_muffin, y_test_muffin = _____________________________________________
```

*Note:* it may take a while for your code to process all the data. This is normal.

Finally, we need to combine `X_train_chihuahua` and `X_train_muffin` into `X_train`, and `y_train_chihuahua` and `y_train_muffin` into `y_train`. We will do the same for the testing data. Find a function from the `NumPy` library which would let you do this.

| | |
|---|---|
| What function could you use? | |
| What does this function take as input? | |
| What does this function return as output? | |

Add code to **`project2.py`** to combine these datasets in the appropriate place.

At this point, all the image data has been turned into numbers by `extract_color_histogram` and we have split these numbers into train and test features and labels. We are ready to train the model now!

## Step 3: Train the Model

For our project, we need to predict whether an image shows a "chihuahua" or a "blueberry muffin". To do this, we will use **logistic regression**.

In 1–2 sentences, describe what kind of learning and reasoning this is, and why (i.e., what data you use for training and for predicting, and how that determines the type of model you will use).

Now, add code in Thonny that trains the model using the approach we showed in class. Use 1000 iterations for training. Paste the code you used below as well.

## Step 4: Evaluate the Model

Congratulations on training your first model! Go ahead and calculate the following metrics in your code, and answer the questions below.

**Accuracy**

| | |
|---|---|
| How would you calculate the accuracy score of your model based on the test data? Write the code to the side and in Thonny. | |
| What is the test accuracy of your model? | |

**Confusion matrix**

First, write code in Thonny to generate a confusion matrix for your model based on the test set. You can use GenAI or look up example code online to do this.

| | |
|---|---|
| How many images were predicted correctly? | |
| How many images were predicted incorrectly? | |

Paste your confusion matrix into the box on the next page (either as an image or text).

**Click to upload image**

**Or paste text here**

## Step 5: Let’s Predict!

Now that we have trained the model, we want to have some fun with it. We will continuously ask the user for an image URL from the internet and predict whether it is a chihuahua or not.

We're written a function `predict_image_from_url` in **`given.py`** to help with this process. Explain in 1–2 sentences what it is doing and why we need it.

Now, in Step 5 of the starter file, write code that:

1. Continuously gets URL input from the user, until the user presses q to quit.
2. Predicts what the image is.

You might want to add some error handling around processing the image in case it can't be loaded, to keep the whole file from crashing.

In the table below, enter some URLs you used to test your model, along with the probabilities the model found for each class.

| **URL:** | | |
|---|---|---|
| Predicted Class | Chihuahua Probability | Muffin Probability |
| | | |
| | | |
| **URL:** | | |
| Predicted Class | Chihuahua Probability | Muffin Probability |
| | | |
| | | |
| **URL:** | | |
| Predicted Class | Chihuahua Probability | Muffin Probability |
| | | |

What do you notice from the table above in terms of the probabilities? What does this say about this model, and ML models more generally?

| |
|---|
| |