A.I. BINGO

Gotten a weather forecast from a website or used a weather app Dataset: Prediction:	Sent a voice-to-text message Dataset: Prediction:	Used an online search engine like Google or Bing Dataset: Prediction:	Seen a Google autofill search result Dataset: Prediction:	Had a writing assignment graded by a computer Dataset: Prediction:
Used “safe search” on Google Dataset: Prediction:	Seen a suggested response on Gmail to an email Dataset: Prediction:	Used a Snapchat filter (what’s your favorite?) Dataset: Prediction:	Played a motion-sensitive video game e.g. Mario Party, Nintendo, Wii U, etc. Dataset: Prediction:	Had an Emoji suggested instead of a word e.g. “lol” is replaced for an Emoji smiley face Dataset: Prediction:
Seen a sponsored product on Google or Amazon e.g. *”since you bought __, we thought you might like...”* Dataset: Prediction:	Had an email go to your spam folder (was it actually spam?) Dataset: Prediction:	**FREE**	Clicked on an Instagram ad (what kinds of ads do you normally see on the app compared to your partner?) Dataset: Prediction:	Seen news articles suggested in a news app (what kinds of articles do you normally see compared to your partner?) Dataset: Prediction:
Had an email labeled as “important” Dataset: Prediction:	Seen a suggested ad on Snapchat (if so, what for? How does this compare to what ads your partner sees?) Dataset: Prediction:	Had a text auto-completed or used autocorrect Dataset: Prediction:	Listened to a recommended song on Spotify (what kind of music do you usually get recommended compared to your partner?) Dataset: Prediction:	Seen a recommended product on Facebook (if so, what for?) Dataset: Prediction:
Seen a “nudge” reminder on Gmail to respond to an email Dataset: Prediction:	Used a fingerprint to unlock a device or opened a device with your face Dataset: Prediction:	Used a map app to find a path to a destination Dataset: Prediction:	Used an app to recognize a song playing Dataset: Prediction:	Communicated with a customer service bot Dataset: Prediction:

Name: _____ Date: _____

Introduction to Supervised Machine Learning Activity

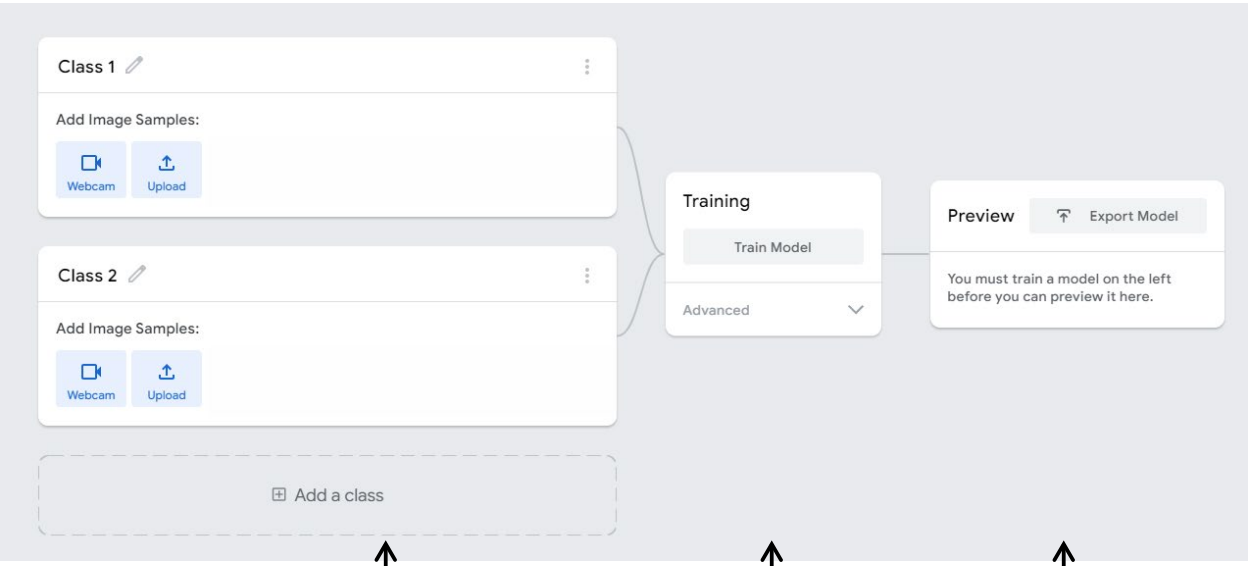
Description

In this exercise, you will learn about the three components of an artificial intelligence (AI) system.

Instructions

1. Go to https://drive.google.com/drive/u/1/folders/1PGO3lu_4hztRAyCPK4g8AVQtk79A79j3, download the cat-dog data to your computer.
2. Go to: https://teachablemachine.withgoogle.com/
3. Click "Get Started → Image Project → Standard image model"

Identify the three parts of an AI system in the teachable machine as discussed in class:



_____ _____ _____

Steps to add training dataset:

1. Click “Class 1 → Upload → Choose images from your files…”
2. Locate the training dataset in your hard drive.
3. Click “dog” and select all images. Click open.
4. Repeat with cat images for Class 2.

Train Model

Add test data

What is the output?

What happens when you select a test image other than cat and dog?

What happens if you only train on one class (say dog)?

What happens if you decrease the number of images in the training dataset?

Name: _____ Date: _____

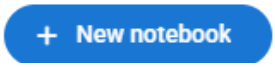
Machine Learning with Python Activity

Description

In this exercise, you will learn how to do clustering and classification using Python.

Instructions

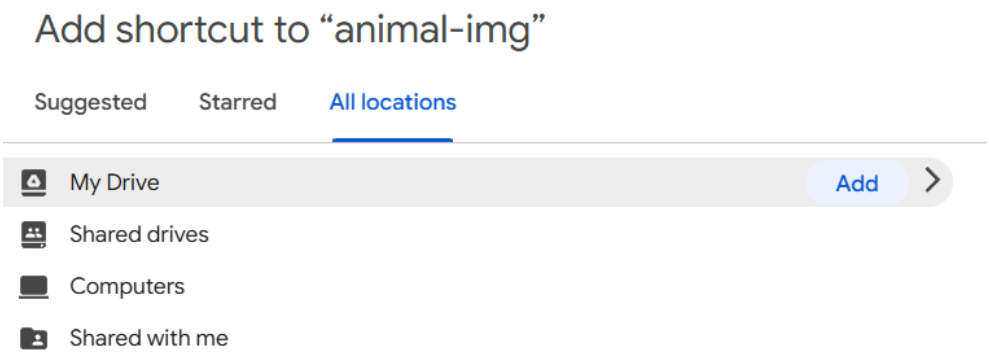
A. Iris Clustering and Classification

1. Go to Google Colab (https://colab.research.google.com/) and create a new notebook 

2. Follow the tutorial in slides to build k-means clustering model and logistic regression model on the Iris data.
    a. What results do you have?

B. Animal Image Classification

1. Click the shared link (https://drive.google.com/drive/folders/1wCgMIwlguegGmxBvhbxdcJE1vRHeOwPT?usp=drive_link) and sign in to your Google Drive.
2. Click the folder name animal-img, then go to **Organize** → **Add shortcut**, and add the shortcut to **My Drive**.



3. Build the image classifier:
    a. Go to Google Colab (https://colab.research.google.com/) and create a new notebook.
    b. Import required libraries

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

c. Mount Google Drive and load the datasets

```
from google.colab import drive
drive.mount('/content/drive')
train_dir = '/content/drive/MyDrive/animal-img/train'
test_dir  = '/content/drive/MyDrive/animal-img/test'
IMG_SIZE = (128, 128)
BATCH_SIZE = 32
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)

test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)

class_names = train_ds.class_names
print("Detected Classes:", class_names)
```

d. Build the model

```
base_model = tf.keras.applications.MobileNetV2(
    input_shape=IMG_SIZE + (3,),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = False  # Freeze for now

model = tf.keras.Sequential([
    layers.Rescaling(1./255, input_shape=IMG_SIZE + (3,)),
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(len(class_names), activation='softmax')
])
```

e. Set optimizer and loss function

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

f. Train the model

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

4. Test the model

```
test_images = []
test_labels = []

for image, label in test_ds.unbatch():
    test_images.append(image)
    test_labels.append(label)

test_images = np.array(test_images)
test_labels = np.array(test_labels)

# Prompt user for a class name
print("Available classes:", class_names)
selected_class = input("Enter a class name from the list above: ").strip()

# Get class index and filter images
class_index = class_names.index(selected_class)
indices = np.where(test_labels == class_index)[0]

# Predict labels for all selected images
predicted_labels = []
for idx in indices:
    img = tf.expand_dims(test_images[idx], axis=0)
    pred = model.predict(img, verbose=0)
    predicted_labels.append(np.argmax(pred))

# Plot the images
plt.figure(figsize=(15, 8))
for i, idx in enumerate(indices):
    plt.subplot(3, int(np.ceil(len(indices) / 3)), i + 1)
    plt.imshow(test_images[idx].astype("uint8"))
    plt.axis('off')
    true_label = class_names[test_labels[idx]]
    pred_label = class_names[predicted_labels[i]]
    plt.title(f"T: {true_label}\nP: {pred_label}", fontsize=9)
plt.tight_layout()
plt.show()
```

a. Which classes have lower prediction accuracy?
b. Find the misclassified images in the test data. Can you guess why they were incorrectly predicted?

Name: _____ Date: _____

Can AI Really Code?

Description

In this exercise, you will ask ChatGPT to write simple Python code.

Instructions

You can work in group or individually and ask ChatGPT to write simple Python programs to solve following problems:

1. Convert temperatures from Fahrenheit to Celsius.
2. Calculates the sum of numbers from 1 to 100.
3. Create a text-based adventure game.