UNIVERSITY OF CANBERRA
AUSTRALIA'S CAPITAL UNIVERSITY

*Faculty of Science and Technology*

**Assignment Coversheet**

| | |
|---|---|
| **Student ID number & Student Name** | U3259101, Francis Graceman Abakah |
| **Unit name** | Software Technology 1 |
| **Unit number** | 4483 |
| **Unit Tutor** | Gurpreet Kaur |
| **Assignment name** | ST1 Capstone Project – Semester 2 2023 |
| **Due date** | 20/12/23 |
| **Date submitted** | 20/12/23 |

**You must keep a photocopy or electronic copy of your assignment.**

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

**Signature of student** Abakah **Date:20/12/23**

# Table of Contents



# Introduction and background

This report describes the details of Python Capstone Project for ST1 unit within the scope of the project requirements provided in the assignment handout. I have decided to work on the project developing a pet facial expression classifier from the public available dataset in Kaggle repository.

Understanding the emotions and general wellbeing of animals through their facial expressions is crucial. Similar to us, pets convey a range of emotions through their facial expressions. Accurate interpretation of these expressions allows veterinarians and pet carers to understand the emotional states of their charges. This knowledge is essential for guaranteeing the animals' general wellbeing since it makes it easier to spot symptoms of stress, discomfort, or pain, which enables prompt treatments and improved care.

This analysis has the potential to improve the relationship between humans and animals. Pet owners who are able to read their pets' facial expressions are better able to comprehend and address their wants and feelings. As a result, the pet and owner develop a stronger, more sympathetic bond that improves both of their quality of life. Recognising particular expressions also helps with behaviour and pet training. It makes it possible to determine what makes pets feel good or bad, leading to more compassionate and successful training methods.

The study of facial expressions in pets makes a substantial contribution to the field of animal welfare as well. It offers insightful information about the emotional lives of animals, which can help develop rules and regulations for the humane treatment of animals in zoos, homes, and shelters, among other places.

Furthermore, this area of study is critical in the development of AI and machine learning applications. By training models to recognize and interpret pet facial expressions, innovative technologies can be developed for automated monitoring of pets' health and emotional states, enhancing pet care, strengthening human-animal bond and identifying the species and breed of a pet.

## Dataset description

The dataset is available publicly in Kaggle repository with a license specified as Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). One thousand faces of different pets, including dogs, cats, hamsters, rabbits, sheep, horses, and birds, are included in this dataset. The pictures depict the range of emotions these animals are capable of expressing, including joy, sorrow, rage, etc. Using pet face photos, you can make entertaining and imaginative projects, learn more about the feelings and personalities of your pets, and advance the fields of pet face recognition research and animal welfare.

Data Description:

- Data format: JPG
- Number of images: 1000
    - Contains various different types of images with around 250 images of each emotion being captured.
- The emotions are:
    - Angry
    - Sad
    - Happy
    - Other
- Number of classes: 4
- Applicability: Suitable for advancing the fields of pet face recognition research and animal welfare.

## Methodology

1. Stage 1: Exploratory data analysis for pet expressions images from the dataset (Google colab).
2. Stage 2: Predictive analytics development using machine learning platform/tools (using teachable machine with Google/Google colab).
3. Stage 3: Implementation and Deployment of the software technology tool for real world testing (using teachable machine with Google/Google colab).

## Stage 1: Exploratory Data Analysis Stage

The goal of exploratory data analysis, which is the most crucial initial step, is to fully comprehend the data, provide guidance on the predictive analytics algorithms to be employed, and project the software tool's projected performance in real-world scenarios.

Understanding the data, performing basic exploratory data analysis, and visualising the results were all part of the first phase of the software development. Google Colab was selected as the experimental environment. Several actions must be taken before the exploratory data analysis may start, including:

- Downloading data subsets from Kaggle repository (https://www.kaggle.com/datasets/anshtanwar/pets-facial-expression-dataset )
- Uploading the data to Google drive
- Linking a new Google Colab notebook to the Google drive
- Installing and importing several Python libraries to carry out EDA
- Relocating the project working folder to the project dataset's Google Drive location.

## Stage 1: EDA (Exploratory Data Analytics)

```
# Importing necessary libraries EDA in Google Colab
import pandas as pd

# Importing Google Colab's drive module to access Google Drive files
from google.colab import drive

# Mounting Google Drive to the current Colab session to access stored datasets
# The 'force_remount=True' option ensures that the drive is remounted for each execution of the cell
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

```
!pip install tensorflow==2.9.1
```

Requirement already satisfied: tensorflow==2.9.1 in /usr/local/lib/python3.10/dist-packages (2.9.1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.6.3)
Requirement already satisfied: flatbuffers<2,>=1.12 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.12)

Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.60.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (3.9.0)
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (2.9.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.1.2)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (16.0.6)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (23.2)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (3.19.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.16.0)
Requirement already satisfied: tensorboard<2.10,>=2.9 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (2.9.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (0.34.0)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (2.9.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.14.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow==2.9.1) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (0.3.0)

Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.2.2)

```python
# Import essential system libraries
import os
import time
import shutil
import pathlib
import itertools

# Importing libraries for data handling and visualization
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# Importing TensorFlow and Keras for Deep Learning
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras import regularizers

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")
```

```
print ('modules loaded')
```

```
modules loaded
```

```
# Reading and Organizing Data for Analysis

# Define the directory path to the dataset
data_dir = '/content/gdrive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)'

# Initialize lists to store file paths and corresponding labels
filepaths = []
labels = []

# Loop through each sub-folder in the dataset directory
folds = os.listdir(data_dir)
for fold in folds:
    foldpath = os.path.join(data_dir, fold)
    filelist = os.listdir(foldpath)
    for file in filelist:
        fpath = os.path.join(foldpath, file)
        filepaths.append(fpath)
        labels.append(fold)

# Creating a pandas DataFrame to hold file paths and labels
Fseries = pd.Series(filepaths, name= 'filepaths')
Lseries = pd.Series(labels, name='labels')
df = pd.concat([Fseries, Lseries], axis= 1)
```

```
# EDA Q1: How is the data distribution
df.head(5)
```

| | filepaths | labels |
|---|---|---|
| **0** | /content/gdrive/MyDrive/Capstone Project/PetFa... | Sad |
| **1** | /content/gdrive/MyDrive/Capstone Project/PetFa... | Sad |
| **2** | /content/gdrive/MyDrive/Capstone Project/PetFa... | Sad |
| **3** | /content/gdrive/MyDrive/Capstone Project/PetFa... | Sad |

| | filepaths | labels |
|---|---|---|
| **4** | /content/gdrive/MyDrive/Capstone Project/PetFa... | Sad |

```
df.tail(5)
```

| | filepaths | labels |
|---|---|---|
| **998** | /content/drive/MyDrive/Capstone Project/PetFac... | Other |
| **999** | /content/drive/MyDrive/Capstone Project/PetFac... | Other |
| **1000** | /content/drive/MyDrive/Capstone Project/PetFac... | Other |
| **1001** | /content/drive/MyDrive/Capstone Project/PetFac... | Other |
| **1002** | /content/drive/MyDrive/Capstone Project/PetFac... | Other |

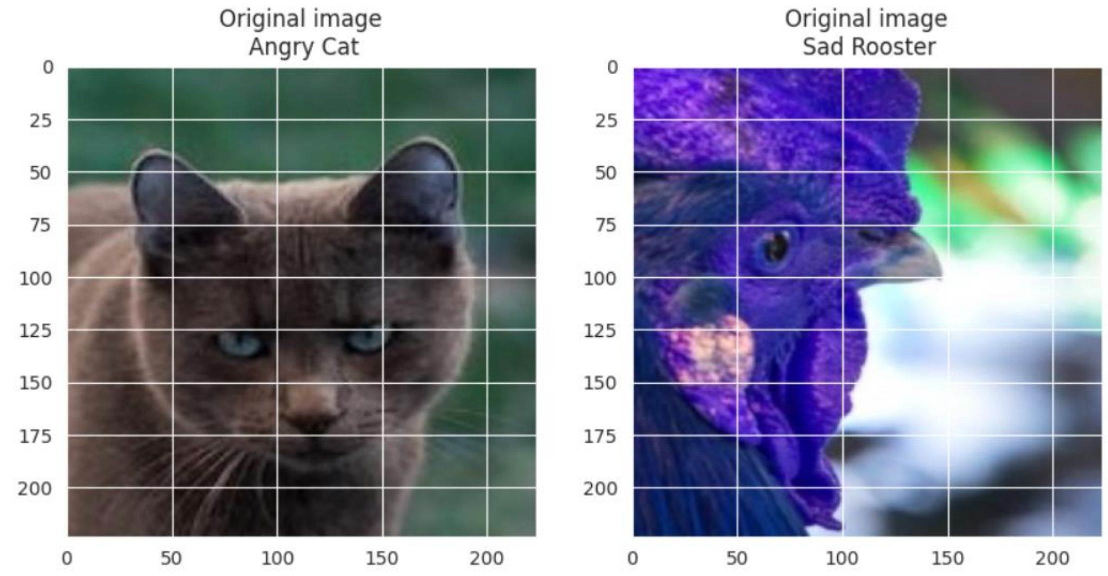
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1003 entries, 0 to 1002
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   filepaths  1003 non-null   object
 1   labels     1003 non-null   object
dtypes: object(2)
memory usage: 15.8+ KB
```

```
# EDA Q2: How do images from different classes look like (Read and
Display Images)

# Importing necessary libraries for image processing and visualization
import cv2
import matplotlib.pyplot as plt

# Configuring Matplotlib to display images inline
%matplotlib inline

# Defining the file paths for two images
img_path_1 = '/content/gdrive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Angry/04.jpg'
img_1 = cv2.imread(img_path_1)
```

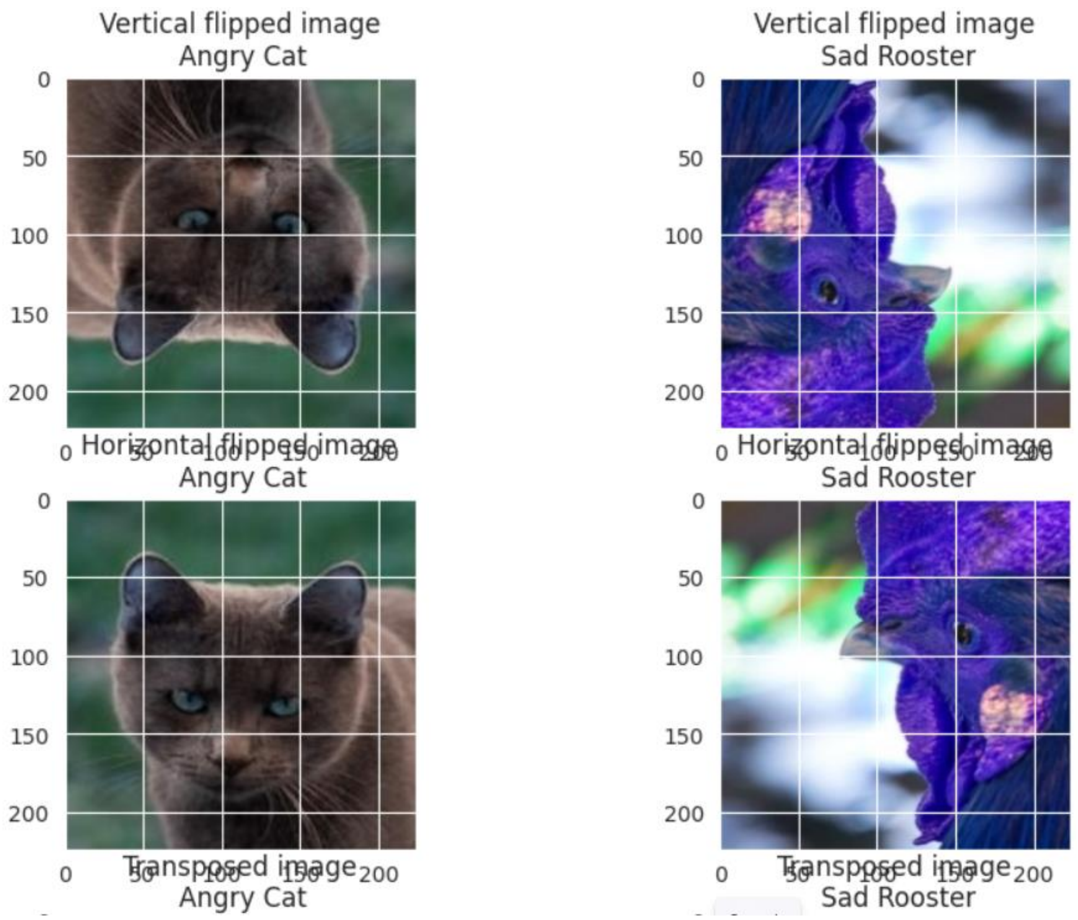
```
img_path_2 = '/content/gdrive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Sad/005.jpg'
img_2 = cv2.imread(img_path_2)

# Setting up the figure for displaying the images
plt.figure(figsize=(10, 10))

# Displaying Images
plt.subplot(121)
plt.imshow(img_1),plt.title('Original image\n Angry Cat')
plt.subplot(122)
plt.imshow(img_2),plt.title('Original image\n Sad Rooster')
```

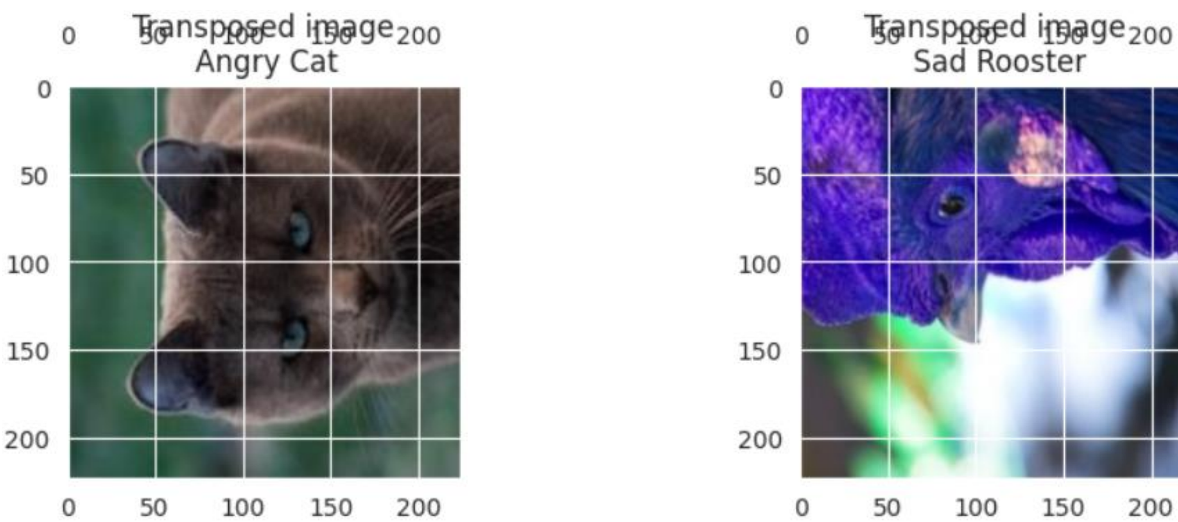
```
(<matplotlib.image.AxesImage at 0x7fe9ad9f7190>,
 Text(0.5, 1.0, 'Original image\n Sad Rooster'))
```




```
# EDA Q3 - How does the images from different classes look like with
geometrical transformations (vertical flipping, horizontal flipping,
transposing)

# Importing libraries for image processing and visualization
import cv2
import matplotlib.pyplot as plt

# Configuring Matplotlib to display images inline
%matplotlib inline

# Defining the file paths for two images
img_path_1 = '/content/gdrive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Angry/04.jpg'
img_1 = cv2.imread(img_path_1)
```

```
img_path_2 = '/content/gdrive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Sad/005.jpg'
img_2 = cv2.imread(img_path_2)

# Performing basic image manipulations: flipping and transposing
flip_img_v1=cv2.flip(img_1,0) # Vertically flipping the image
flip_img_v2=cv2.flip(img_2,0) # Vertically flipping the image
flip_img_h1=cv2.flip(img_1,1) # Horizontally flipping the image
flip_img_h2=cv2.flip(img_2,1) # Horizontally flipping the image
transp_img_1=cv2.transpose(img_1,1) # Transposes the image
transp_img_2=cv2.transpose(img_2,1) # Transposes the image

# Setting up the figure for displaying the images
plt.figure(figsize=(10,10))

# Displaying the vertically flipped images
plt.subplot(321)
plt.imshow(flip_img_v1),plt.title('Vertical flipped image\n Angry Cat')
plt.subplot(322)
plt.imshow(flip_img_v2),plt.title('Vertical flipped image\n Sad
Rooster')

# Displaying the horizontally flipped images
plt.subplot(323)
plt.imshow(flip_img_h1), plt.title('Horizontal flipped image\n Angry
Cat')
plt.subplot(324)
plt.imshow(flip_img_h2), plt.title('Horizontal flipped image\n Sad
Rooster')

# Displaying the transposed images
plt.subplot(325)
plt.imshow(transp_img_1),plt.title('Transposed image\n Angry Cat')
plt.subplot(326)
plt.imshow(transp_img_2),plt.title('Transposed image\n Sad Rooster')
```




```
# EDA Q4: How much data can be used for training, validation and
testing?

# Splitting the DataFrame into Train, Validation, and Test Sets

# Train dataframe
```

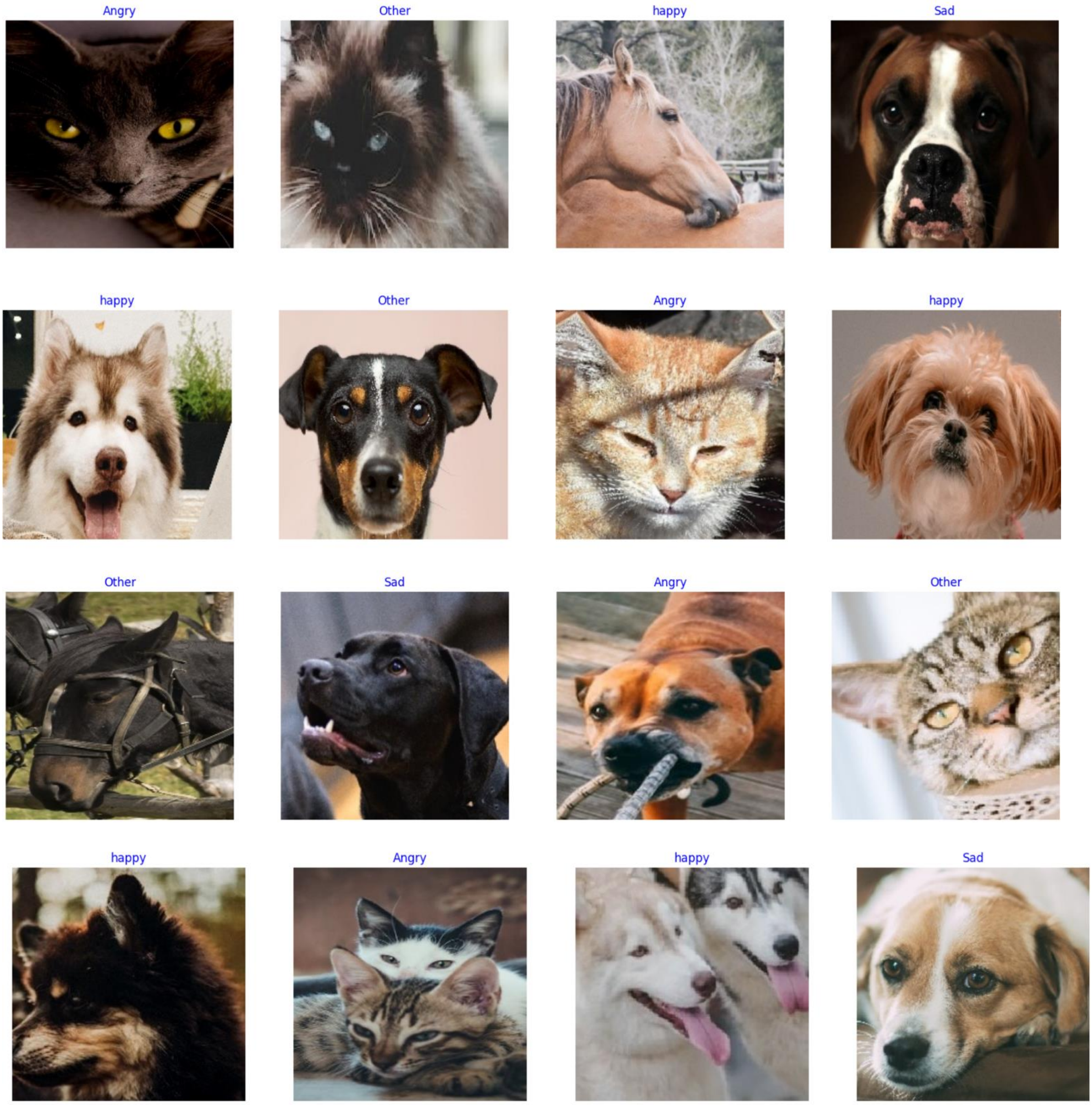
```
train_df, dummy_df = train_test_split(df,  train_size= 0.8, shuffle=
True, random_state= 123)

# Valid and test dataframe
valid_df, test_df = train_test_split(dummy_df,  train_size= 0.6,
shuffle= True, random_state= 123)
```

```
# EDA Q5: Can we visualise the data?

# Setting Up Image Data Generators for Visualizing and Preprocessing
the Data

# Defining basic parameters for image processing
batch_size = 16
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)

# This ensures efficient processing during testing and limits batch
size to 80.
ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1,
ts_length + 1) if ts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size

# Function for the ImageDataGenerator,returns the input image
unmodified
def scalar(img):
    return img

# Creating ImageDataGenerator instances for training and testing
# These generators apply the custom preprocessing function 'scalar'
during image augmentation
tr_gen = ImageDataGenerator(preprocessing_function= scalar)
ts_gen = ImageDataGenerator(preprocessing_function= scalar)

# Setting up the training data generator
train_gen = tr_gen.flow_from_dataframe( train_df, x_col= 'filepaths',
y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                    color_mode= 'rgb', shuffle= True,
batch_size= batch_size)

# Setting up the validation data generator
valid_gen = ts_gen.flow_from_dataframe( valid_df, x_col= 'filepaths',
y_col= 'labels', target_size= img_size, class_mode= 'categorical',
```

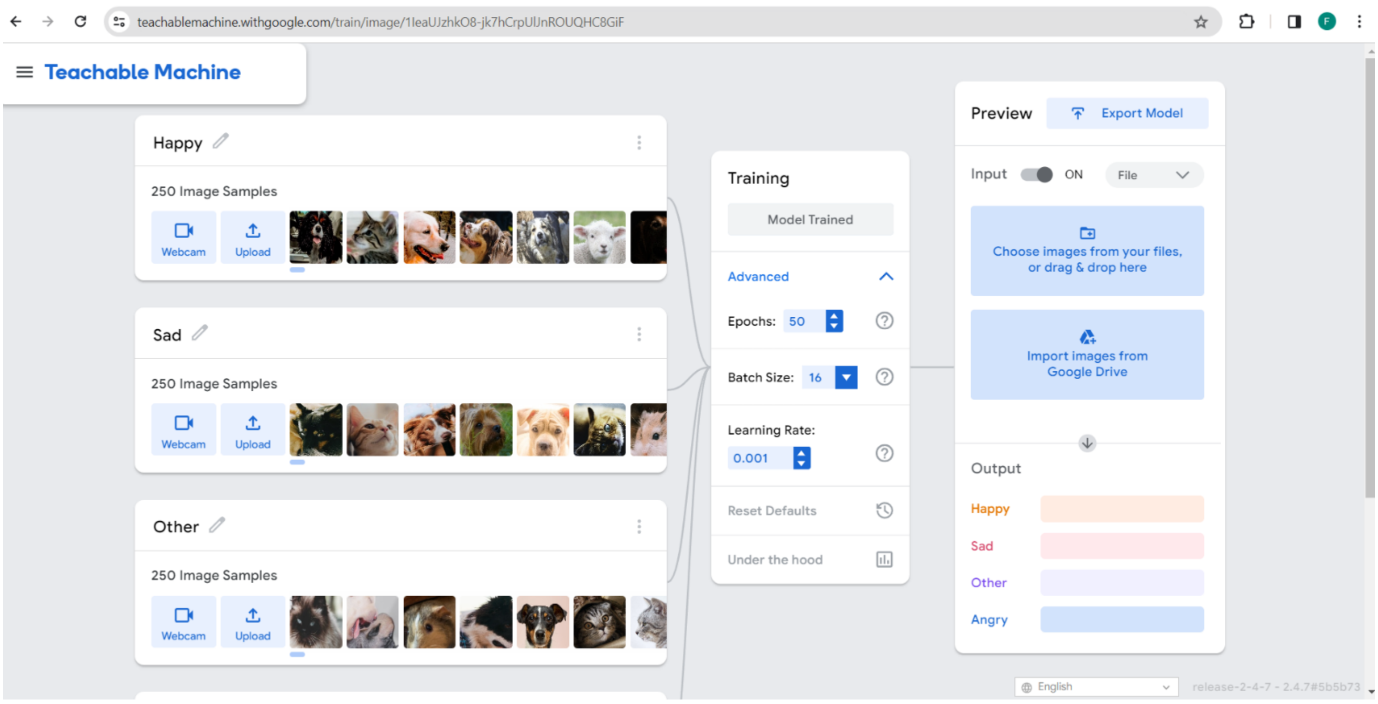
```
                                     color_mode= 'rgb', shuffle= True,
batch_size= batch_size)

# Setting up the test data generator with the custom test_batch_size
and shuffle set to False
test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths',
y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                     color_mode= 'rgb', shuffle= False,
batch_size= test_batch_size)
```

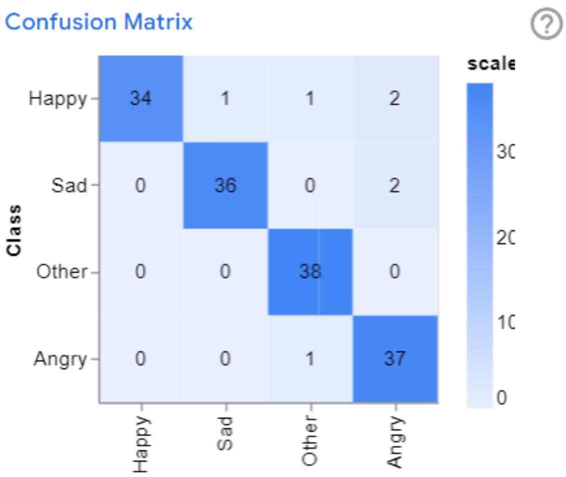
```
Found 799 validated image filenames belonging to 4 classes.
Found 120 validated image filenames belonging to 4 classes.
Found 81 validated image filenames belonging to 4 classes.
```

```
# EDA Q6: How does the samples from the training subset look like ?

g_dict = train_gen.class_indices      # Retrieves a dictionary mapping
class names to their numeric indices
classes = list(g_dict.keys())      # Extracts class names as a list of
strings
images, labels = next(train_gen)       # retrieves the next batch of
images and labels

# Setting up the figure for displaying the images
plt.figure(figsize= (20, 20))

# Looping through the first 16 images in the batch to display them
for i in range(16):
    plt.subplot(4, 4, i + 1)
    image = images[i] / 255       # Scales data to range (0 - 255)
    plt.imshow(image)
    index = np.argmax(labels[i])  # Get the index of the image
    class_name = classes[index]   # Get the class of the image

    # Adding the class name as the title of the subplot
    plt.title(class_name, color= 'blue', fontsize= 12)

    # Turning off the axis for a cleaner look
    plt.axis('off')
plt.show()
```


Angry
Other
happy
Sad
happy
Other
Angry
happy
Other
Sad
Angry
Other
happy
Angry
happy
Sad

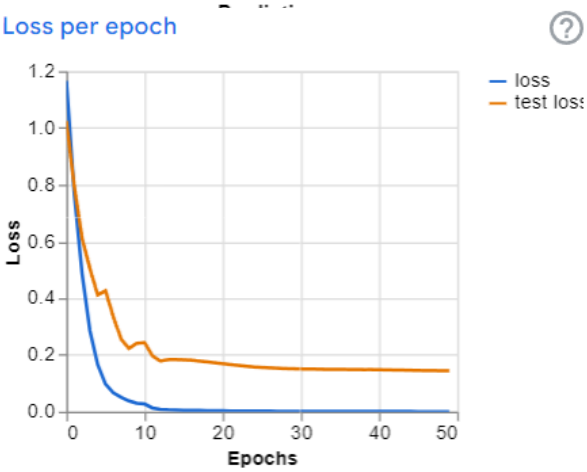
# Stage 2: PDA (Predictive Data Analytics) With Teachable Machine with Google

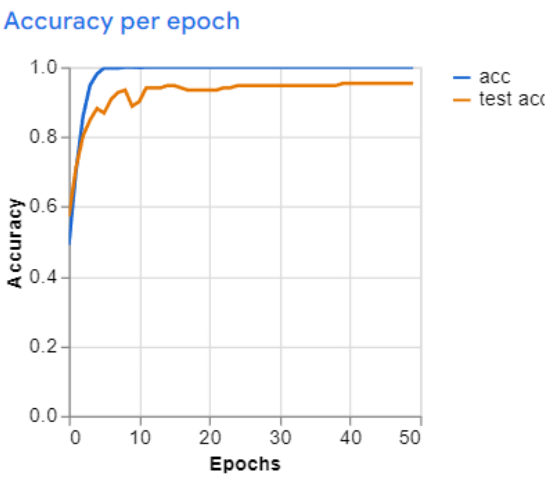



Accuracy per class

| CLASS | ACCURACY | # SAMPLES |
|---|---|---|
| Happy | 0.89 | 38 |
| Sad | 0.95 | 38 |
| Other | 1.00 | 38 |
| Angry | 0.97 | 38 |



## Stage 3: Deployment/ Implementation with Teachable Machine with Google

```
# Importing necessary libraries for model loading and image processing
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Setting NumPy print options for better readability
# Suppresses scientific notation in NumPy arrays
np.set_printoptions(suppress=True)

# Load the model
model = load_model("/content/drive/MyDrive/Capstone
Project/converted_keras/keras_model.h5", compile=False)

# Read class names
class_names = open("/content/drive/MyDrive/Capstone
Project/converted_keras/labels.txt", "r").readlines()

# Preparing a NumPy array with the shape required by the Keras model
for input
# Only 1 image can go into the array
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Loading and processing the image for model prediction
image = Image.open("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Angry/02.jpg").convert("RGB")

#  Resizing the image to 224x224 to be at least 224x224 and then
cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# Converting the image into a numpy array
image_array = np.asarray(image)

# Normalizing the image data
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Makes a prediction using the model
prediction = model.predict(data)
```

```
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)
```

```
1/1 [==============================] - 1s 1s/step
Class: Angry
Confidence Score: 0.9999914
```

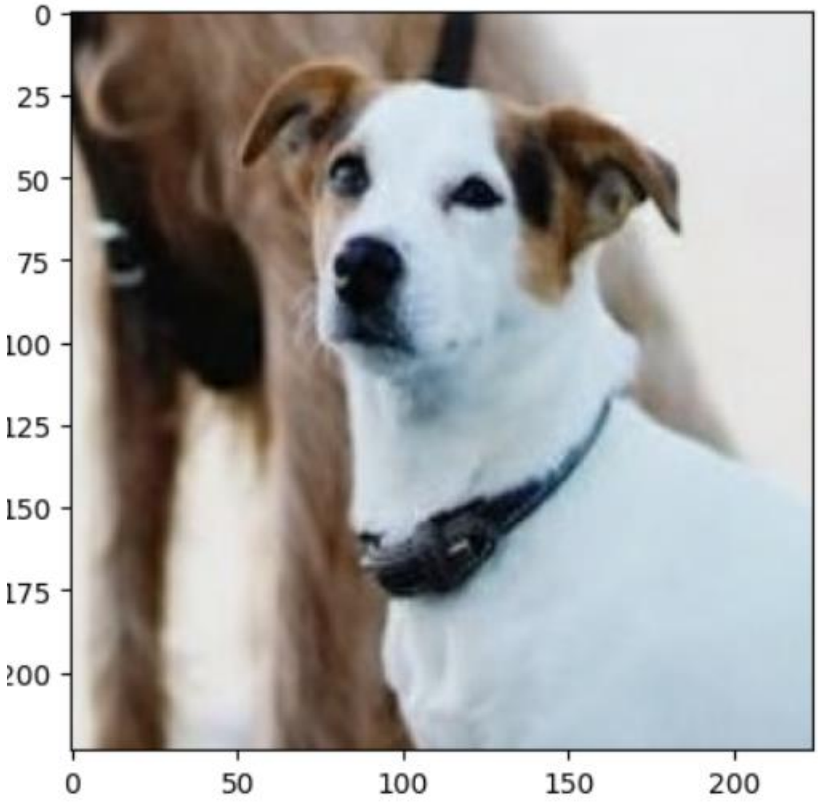
```
# Importing necessary libraries and modules
from warnings import filterwarnings
import tensorflow as tf
from tensorflow import io
from tensorflow import image
from matplotlib import pyplot as plt

# Suppressing warnings for cleaner output
filterwarnings("ignore")

# Reading an image file using TensorFlow
tf_img = io.read_file("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Angry/02.jpg")

# Decoding the image file (PNG format) into a tensor and setting the
number of color channels to 3 (RGB)
tf_img = image.decode_png(tf_img, channels=3)

print(tf_img.dtype)
plt.imshow(tf_img)
# plt.show()
```


```
# Importing necessary libraries for model loading and image processing
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Setting NumPy print options for better readability
# Suppresses scientific notation in NumPy arrays
np.set_printoptions(suppress=True)

# Load the model
model = load_model("/content/drive/MyDrive/Capstone
Project/converted_keras/keras_model.h5", compile=False)

# Read class names
class_names = open("/content/drive/MyDrive/Capstone
Project/converted_keras/labels.txt", "r").readlines()

# Preparing a NumPy array with the shape required by the Keras model
for input
# Only 1 image can go into the array
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Loading and processing the image for model prediction
image = Image.open("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Other/16.jpg").convert("RGB")

#  Resizing the image to 224x224 to be at least 224x224 and then
cropping from the center
```

```python
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# Converting the image into a numpy array
image_array = np.asarray(image)

# Normalizing the image data
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Makes a prediction using the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)
```

```
1/1 [==============================] - 1s 898ms/step
Class: Other
Confidence Score: 0.9995234
```

```python
# Importing necessary libraries and modules
from warnings import filterwarnings
import tensorflow as tf
from tensorflow import io
from tensorflow import image
from matplotlib import pyplot as plt

# Suppressing warnings for cleaner output
filterwarnings("ignore")

# Reading an image file using TensorFlow
tf_img = io.read_file("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Other/16.jpg")

# Decoding the image file (PNG format) into a tensor and setting the
number of color channels to 3 (RGB)
tf_img = image.decode_png(tf_img, channels=3)

print(tf_img.dtype)
plt.imshow(tf_img)
# plt.show()
```



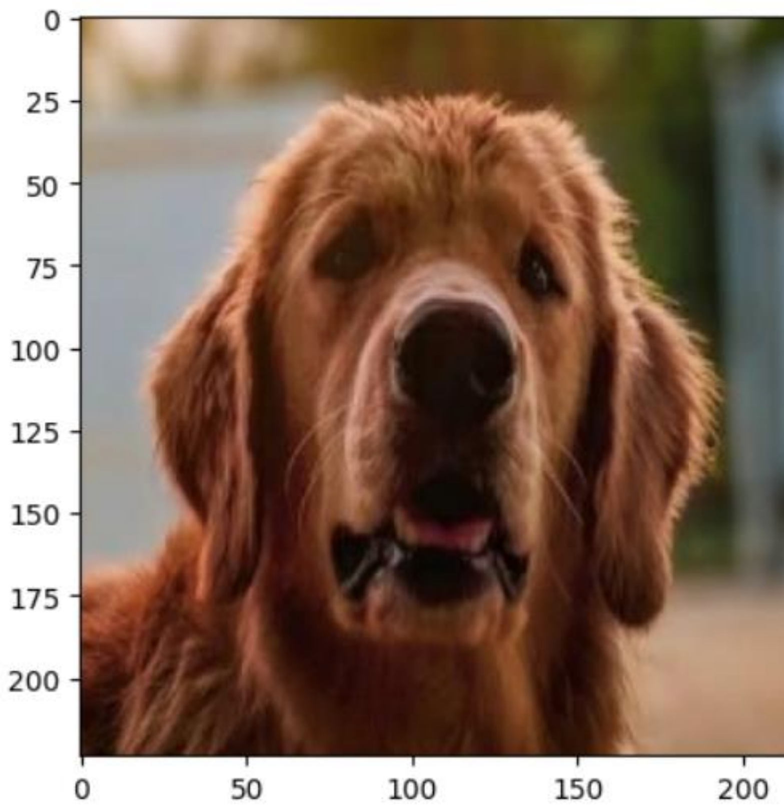

```
# Importing necessary libraries for model loading and image processing
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Setting NumPy print options for better readability
# Suppresses scientific notation in NumPy arrays
np.set_printoptions(suppress=True)

# Load the model
model = load_model("/content/drive/MyDrive/Capstone
Project/converted_keras/keras_model.h5", compile=False)

# Read class names
class_names = open("/content/drive/MyDrive/Capstone
Project/converted_keras/labels.txt", "r").readlines()

# Preparing a NumPy array with the shape required by the Keras model
for input
# Only 1 image can go into the array
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Loading and processing the image for model prediction
image = Image.open("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Sad/026.jpg").convert("RGB")
```

```
#  Resizing the image to 224x224 to be at least 224x224 and then
cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# Converting the image into a numpy array
image_array = np.asarray(image)

# Normalizing the image data
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Makes a prediction using the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)
```

```
1/1 [==============================] - 1s 900ms/step
Class: Sad
Confidence Score: 0.99895346
```

```
# Importing necessary libraries and modules
from warnings import filterwarnings
import tensorflow as tf
from tensorflow import io
from tensorflow import image
from matplotlib import pyplot as plt

# Suppressing warnings for cleaner output
filterwarnings("ignore")

# Reading an image file using TensorFlow
tf_img = io.read_file("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/Sad/026.jpg")

# Decoding the image file (PNG format) into a tensor and setting the
number of color channels to 3 (RGB)
tf_img = image.decode_png(tf_img, channels=3)

print(tf_img.dtype)
plt.imshow(tf_img)
```

```
# plt.show()
```



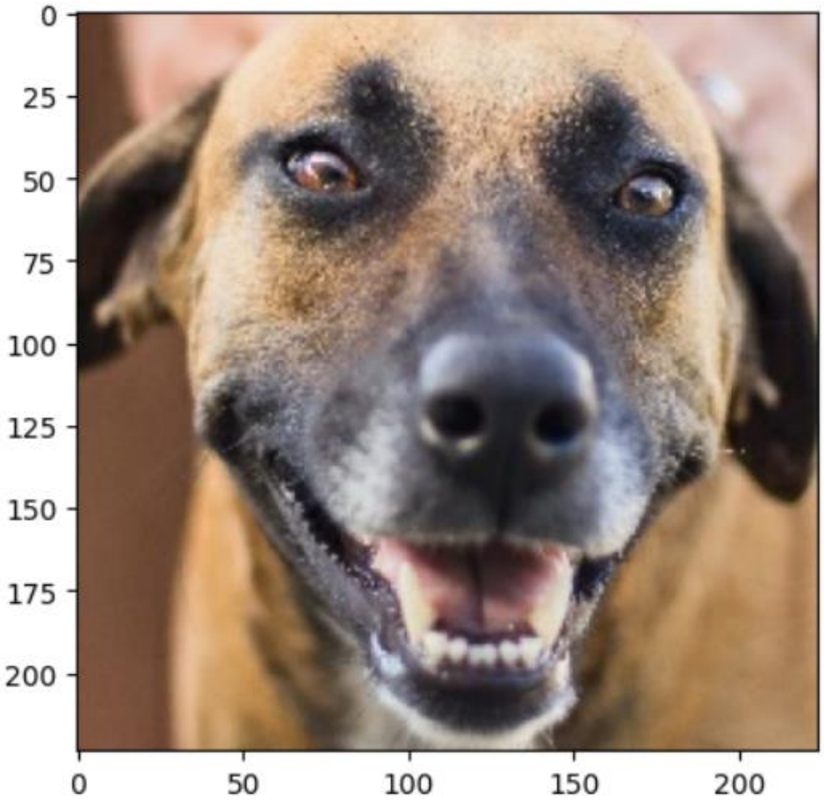

```
# Importing necessary libraries for model loading and image processing
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Setting NumPy print options for better readability
# Suppresses scientific notation in NumPy arrays
np.set_printoptions(suppress=True)

# Load the model
model = load_model("/content/drive/MyDrive/Capstone
Project/converted_keras/keras_model.h5", compile=False)

# Read class names
class_names = open("/content/drive/MyDrive/Capstone
Project/converted_keras/labels.txt", "r").readlines()

# Preparing a NumPy array with the shape required by the Keras model
for input
# Only 1 image can go into the array
```

```python
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Loading and processing the image for model prediction
image = Image.open("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/happy/001.jpg").convert("RGB")

#  Resizing the image to 224x224 to be at least 224x224 and then
cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# Converting the image into a numpy array
image_array = np.asarray(image)

# Normalizing the image data
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Makes a prediction using the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)
```

```
1/1 [==============================] - 1s 998ms/step
Class: Happy
Confidence Score: 0.99953306
```

```python
# Importing necessary libraries and modules
from warnings import filterwarnings
import tensorflow as tf
from tensorflow import io
from tensorflow import image
from matplotlib import pyplot as plt

# Suppressing warnings for cleaner output
filterwarnings("ignore")

# Reading an image file using TensorFlow
tf_img = io.read_file("/content/drive/MyDrive/Capstone
Project/PetFacialDataset/archive (1)/happy/001.jpg")
```

```
# Decoding the image file (PNG format) into a tensor and setting the
number of color channels to 3 (RGB)
tf_img = image.decode_png(tf_img, channels=3)

print(tf_img.dtype)
plt.imshow(tf_img)
# plt.show()
```




## Conclusions

The work completed for the ST1 capstone project, which involved designing, developing, implementing, and deploying a Python data-driven pet facial expression classifier, is presented in this paper. In step 1, a thorough exploratory data analysis is conducted; in stage 2, a predictive model is developed using Tensorflow-Keras packages and Google's teachable machine; in stage 3, the model is deployed and implemented in using teachable machine with Google and Google colab. The successful implementation of the predictive model demonstrates the feasibility and potential of using artificial intelligence to interpret animal emotions. However, the project also highlights the need for further research, particularly in expanding the dataset and refining the model for greater accuracy. Overall, this project underscores the growing importance and capabilities of technology in improving the lives of both humans and animals, paving the way for more empathetic and informed interactions with our pets.

## References


https://www.kaggle.com/datasets/anshtanwar/pets-facial-expression-dataset

Kaur, G (2023) Mango leaf disease classifier [Google Colab File]. Canvas.

Kaur, G (2023) ST_Capstone_Project_Final_Report_S2_2023_Template.docx [Word Document]. Canvas.

Kaur, G(2023) 4483 UCC-BRUCE COLTR3-2023 ON-CAMPUS LECTURE_A 01 Software Technology [Recorded Lecture], *Week 9 Data Analysis & Visualization (Matplotlib & NumPy)*. University of Canberra Canvas. 20 January. Available at: https://uclearn.canberra.edu.au/courses/14917/external_tools/197 Accessed 14 December 2023

Kaur, G(2023) 4483 UCC-BRUCE COLTR3-2023 ON-CAMPUS LECTURE_B 01 Software Technology, [Recorded Lecture], *Week 9 Data Analysis & Visualization (Matplotlib & NumPy).* University of Canberra. Canvas. 20 January. Available at: https://uclearn.canberra.edu.au/courses/14917/external_tools/197 Accessed 15 December 2023

Kaur, G(2023) 4483 UCC-BRUCE COLTR3-2023 ON-CAMPUS LECTURE_A 01 Software Technology [Recorded Lecture], *Week 10 Capstone Project Stage2*. University of Canberra Canvas. 20 January. Available at: https://uclearn.canberra.edu.au/courses/14917/external_tools/197 Accessed 16 December 2023

Kaur, G(2023) 4483 UCC-BRUCE COLTR3-2023 ON-CAMPUS LECTURE_B 01 Software Technology, [Recorded Lecture], *Week 10 Capstone Project Stage2.* University of Canberra. Canvas. 20 January. Available at: https://uclearn.canberra.edu.au/courses/14917/external_tools/197 Accessed 17 December 2023

Kaur, G(2023) 4483 UCC-BRUCE COLTR3-2023 ON-CAMPUS LECTURE_A 01 Software Technology [Recorded Lecture], *Week 11 Project Assignment Workshop*. University of Canberra Canvas. 20 January. Available at: https://uclearn.canberra.edu.au/courses/14917/external_tools/197 Accessed 18 December 2023

Kaur, G(2023) 4483 UCC-BRUCE COLTR3-2023 ON-CAMPUS LECTURE_B 01 Software Technology, [Recorded Lecture], *Week 11 Project Assignment Workshop.* University of Canberra. Canvas. 20 January. Available at: https://uclearn.canberra.edu.au/courses/14917/external_tools/197 Accessed 19 December 2023