LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in CSE

Submitted By

Name:- Rishi Banerjee

Roll No: 2205151

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Lab Number	4
Experiment Number	2
Experiment Title	Machine Learning for Cat and Dog Classification
Date of Experiment	23/01/2025
Date of Submission	30/01/2025

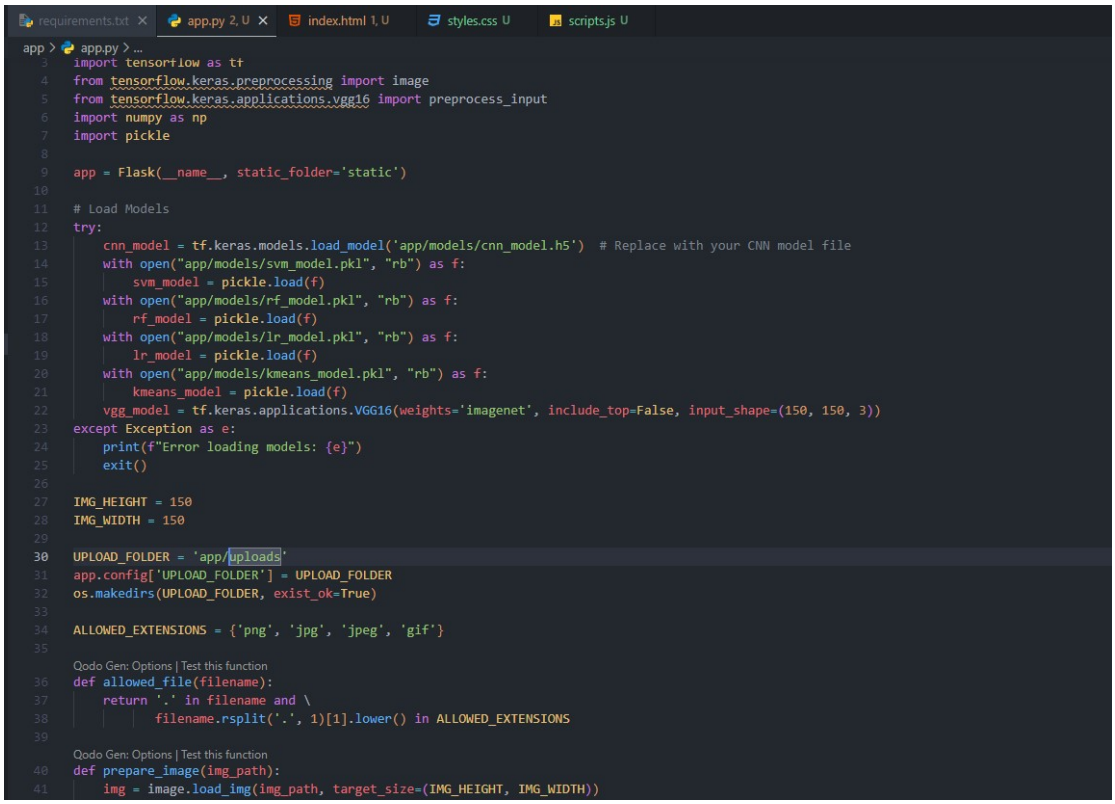
1. Objective:-

To classify images as cats or dogs using machine learning models.

2. Procedure:- (Steps Followed)

1. Collect a labeled dataset of cat and dog images.
2. Preprocess images using OpenCV (resize, flatten, etc.).
3. Train ML models: SVM, Random Forest, Logistic Regression, and K-means Clustering.
4. Save the trained models.
5. Build a Flask backend to load models and handle image uploads.
6. Create a frontend with HTML/CSS for uploading images and selecting models.
7. Display the classification result on the webpage.

3. Code:-



```
requirements.txt   app.py 2, U   index.html 1, U   styles.css U   scripts.js U
app > app.py > ...
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np
import pickle

app = Flask(__name__, static_folder='static')

# Load Models
try:
    cnn_model = tf.keras.models.load_model('app/models/cnn_model.h5')  # Replace with your CNN model file
    with open("app/models/svm_model.pkl", "rb") as f:
        svm_model = pickle.load(f)
    with open("app/models/rf_model.pkl", "rb") as f:
        rf_model = pickle.load(f)
    with open("app/models/lr_model.pkl", "rb") as f:
        lr_model = pickle.load(f)
    with open("app/models/kmeans_model.pkl", "rb") as f:
        kmeans_model = pickle.load(f)
    vgg_model = tf.keras.applications.VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
except Exception as e:
    print(f"Error loading models: {e}")
    exit()

IMG_HEIGHT = 150
IMG_WIDTH = 150

UPLOAD_FOLDER = 'app/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

Qodo Gen: Options | Test this function
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

Qodo Gen: Options | Test this function
def prepare_image(img_path):
    img = image.load_img(img_path, target_size=(IMG_HEIGHT, IMG_WIDTH))
```

requirements.txt | app.py 2, U | index.html 1, U | styles.css U | scripts.js U

app > templates > index.html > ...

```html
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Cat-Dog Classifier</title>
    <link rel="stylesheet" href="/static/styles.css" />
  </head>
  <body>
    <div class="container">
      <h1>Cat-Dog Classifier</h1>
      <input type="file" id="fileInput" />
      <button onclick="uploadImage()">Upload and Predict</button>
      <div id="result"></div>
      <img
        id="uploadedImage"
        src=""
        alt="Uploaded Image"
        style="display: none"
      />
    </div>
    <script src="/static/scripts.js"></script>
  </body>
</html>

```

AD Lab 4 Exp 2.ipynb ☆ Save failed

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

Reconnect T4 ✦ Gemini

```
[ ] try:
      # This command only in Colab.
      %tensorflow_version 2.x
    except Exception:
      pass
    import tensorflow as tf

    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
    from tensorflow.keras.preprocessing.image import ImageDataGenerator

    import os
    from numpy import round
    import matplotlib.pyplot as plt
```

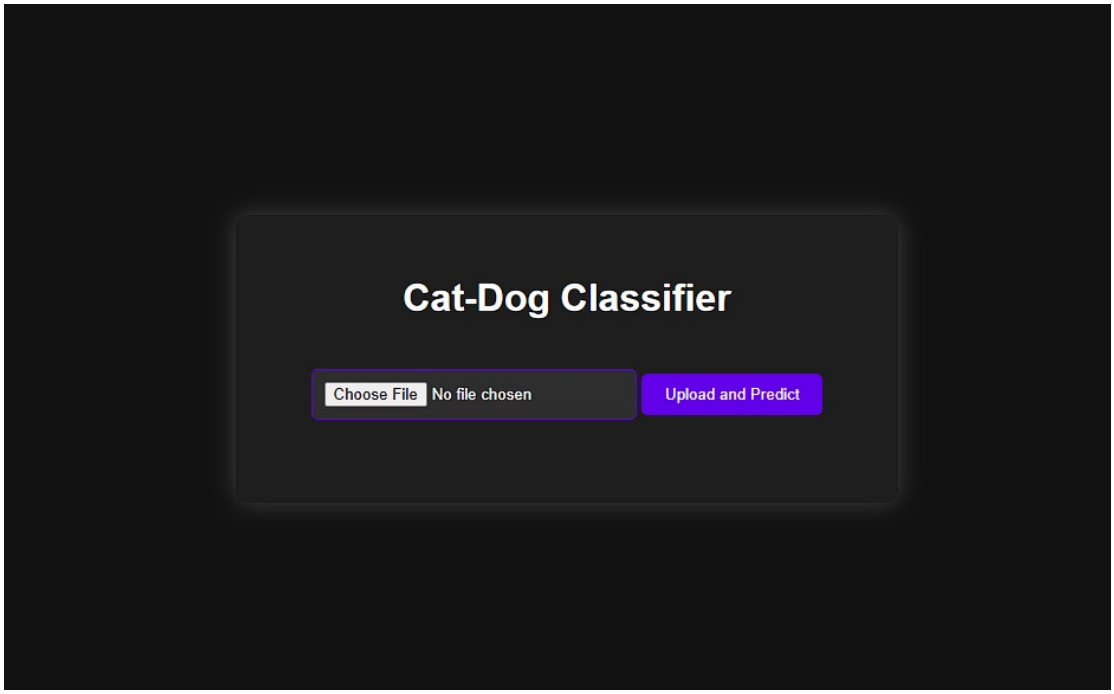
```
Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
```

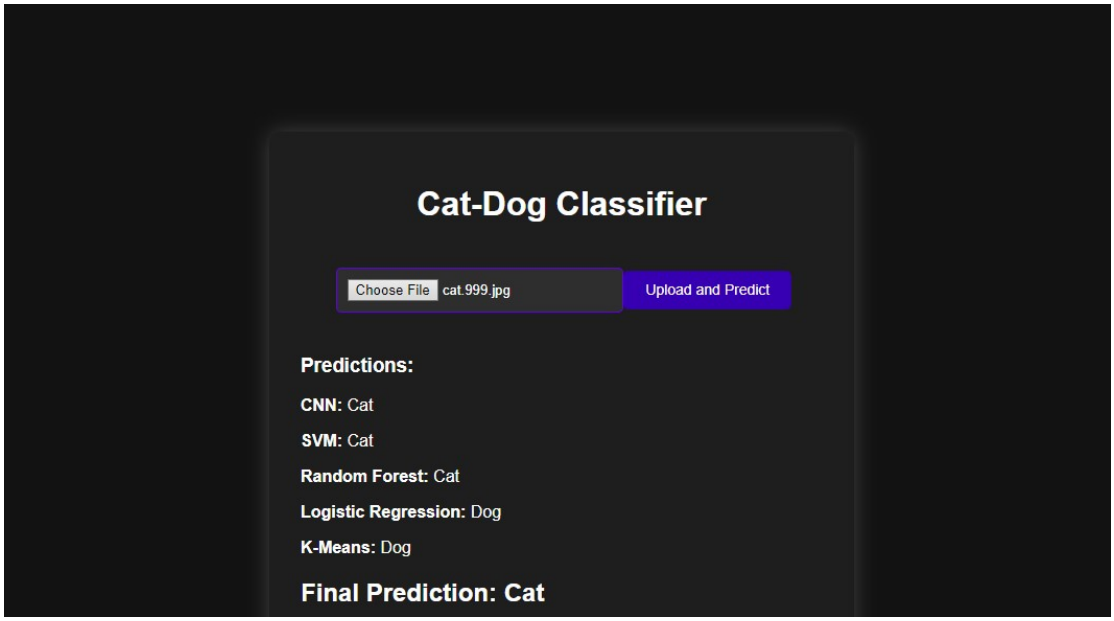
```
[ ] # Get project files
    !wget https://cdn.freecodecamp.org/project-data/cats-and-dogs/cats_and_dogs.zip

    !unzip cats_and_dogs.zip

    PATH = 'cats_and_dogs'

    train_dir = os.path.join(PATH, 'train')
    validation_dir = os.path.join(PATH, 'validation')
    test_dir = os.path.join(PATH, 'test')

    # Get number of files in each directory. The train and validation directories
```

```
# 3
train_image_generator = ImageDataGenerator(rescale = 1./255)
validation_image_generator = ImageDataGenerator(rescale = 1./255)
test_image_generator = ImageDataGenerator(rescale = 1./255)

train_data_gen = train_image_generator.flow_from_directory(
        train_dir,
        batch_size=batch_size,
        target_size=(IMG_HEIGHT, IMG_WIDTH),
        class_mode='binary')

val_data_gen = validation_image_generator.flow_from_directory(
        validation_dir,
        batch_size=batch_size,
        target_size=(IMG_HEIGHT, IMG_WIDTH),
        class_mode='binary')

test_data_gen = test_image_generator.flow_from_directory(
        '/content/cats_and_dogs',
        classes=['test'],
        batch_size=batch_size,
        target_size=(IMG_HEIGHT, IMG_WIDTH),
        class_mode='binary',
        shuffle=False)
```

```
# 4
def plotImages(images_arr, probabilities = False):
    fig, axes = plt.subplots(len(images_arr), 1, figsize=(5,len(images_arr) * 3))
    if probabilities is False:
      for img, ax in zip( images_arr, axes):
          ax.imshow(img)
          ax.axis('off')
    else:
      for img, probability, ax in zip( images_arr, probabilities, axes):
          ax.imshow(img)
          ax.axis('off')
          if probability > 0.5:
```

4. Results/Output:- Entire Screen Shot including Date & Time





5. Remarks:-

Signature of the Student

_____

(Name of the Student)

Signature of the Lab Coordinator

_____

(Name of the Coordinator)