LABORATORY REPORT

**Application Development Lab**

**(CS33002)**

**B.Tech Program in ECSc**

Submitted By

**Name:- ABHISHEK ADARSH**

**Roll No:** 2230141



**Kalinga Institute of Industrial Technology**

**(Deemed to be University)**

**Bhubaneswar, India**

Spring 2024-2025

**Table of Content**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exp No.** | **Title** | **Date of**  **Experiment** | **Date of**  **Submission** | **Remarks** |
| 1. | 1.Build a Resume using HTML/CSS |  |  |  |
| 2. | 1.Machine Learning for Cat and Dog Classification |  |  |  |
| 3. |  |  |  |  |
| 4. |  |  |  |  |
| 5. |  |  |  |  |
| 6. |  |  |  |  |
| 7. |  |  |  |  |
| 8. |  |  |  |  |
| 9. | Open Ended 1 |  |  |  |
| 10. | Open Ended 2 |  |  |  |

|  |  |
| --- | --- |
| **Experiment Number** | 2 |
| **Experiment Title** | Machine Learning for Cat and Dog Classification |
| **Date of Experiment** | 14/01/2025 |
| **Date of Submission** | 20/01/2025 |

1. **Objective:-** To classify images as cats or dogs using machine learning models.
2. **Procedure:-**
3. Collect a labeled dataset of cat and dog images.
4. Preprocess images using OpenCV (resize, flatten, etc.).
5. Train ML models: SVM, Random Forest, Logistic Regression, CNN, and K-means Clustering.
6. Save the trained models.
7. Build a Flask backend to load models and handle image uploads.
8. Create a frontend with HTML/CSS for uploading images and selecting models.

**Code:-**

**Frontend code (html):**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cat and Dog Classifier</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f9f9f9;

margin: 0;

padding: 0;

}

header {

background-color: #4CAF50;

color: white;

padding: 20px;

}

header h1 {

margin: 0;

font-size: 2.5em;

}

main {

margin: 50px auto;

max-width: 600px;

}

.btn {

background-color: #4CAF50;

color: white;

padding: 15px 30px;

font-size: 1.2em;

border: none;

border-radius: 5px;

cursor: pointer;

text-decoration: none;

}

.btn:hover {

background-color: #45a049;

}

footer {

margin-top: 50px;

color: #888;

}

</style>

</head>

<body>

<header>

<h1>Cat and Dog Classifier</h1>

</header>

<main>

<p>Click the button below to open the classification interface and classify your images!</p>

<a href="https://9bc8d64f5b726f4346.gradio.live" target="\_blank" class="btn">Open Classifier</a>

</main>

</body>

</html>

**Backend Code**:

!pip install gradio opencv-python scikit-learn tensorflow

import gradio as gr

import numpy as np

import cv2

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.cluster import KMeans

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.optimizers import Adam

import joblib

from tensorflow.keras.datasets import cifar10

(X, y), (\_, \_) = cifar10.load\_data()

filter\_indices = np.where((y == 3) | (y == 5))[0][:2000]

X = X[filter\_indices]

y = y[filter\_indices]

y = np.where(y == 3, 0, 1)

X = X / 255.0

X = np.array([cv2.resize(img, (32, 32)) for img in X])

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

X\_train\_flat = X\_train.reshape(X\_train.shape[0], -1)

# Load previously saved models or train them if not saved

# Assuming the models are saved or need to be loaded here

# Define the prediction function

def preprocess\_image(image, size=(32, 32)):

if len(image.shape) == 2:

image = cv2.cvtColor(image, cv2.COLOR\_GRAY2RGB)

image = cv2.resize(image, size)

image = image / 255.0

return image

def predict(image, model\_name):

models = {

"SVM": svm\_model,

"Random Forest": rf\_model,

"Logistic Regression": lr\_model,

"CNN": cnn\_model,

"K-Means": kmeans

}

try:

image = preprocess\_image(image)

except Exception as e:

return f"Error in preprocessing the image: {str(e)}"

image\_flat = image.flatten().reshape(1, -1)

try:

if model\_name == "CNN":

image\_reshaped = np.expand\_dims(image, axis=0)

predictions = cnn\_model.predict(image\_reshaped)

confidence = predictions[0][np.argmax(predictions)]

label = np.argmax(predictions)

elif model\_name == "K-Means":

label = kmeans.predict(image\_flat)[0]

confidence = 1.0

else:

label = models[model\_name].predict(image\_flat)[0]

confidence = 1.0

prediction = "Cat" if label == 0 else "Dog"

return f"Prediction: {prediction}\nConfidence: {confidence:.2f}"

except Exception as e:

return f"Error during prediction: {str(e)}"

# Improved Gradio Interface

iface = gr.Interface(

fn=predict,

inputs=[

gr.Image(type="numpy", label="Upload an Image (Cat or Dog)"),

gr.Radio(

["SVM", "Random Forest", "Logistic Regression", "CNN", "K-Means"],

label="Select a Model",

info="Choose a machine learning model for prediction"

)

],

outputs="text",

title="Cat or Dog Classifier",

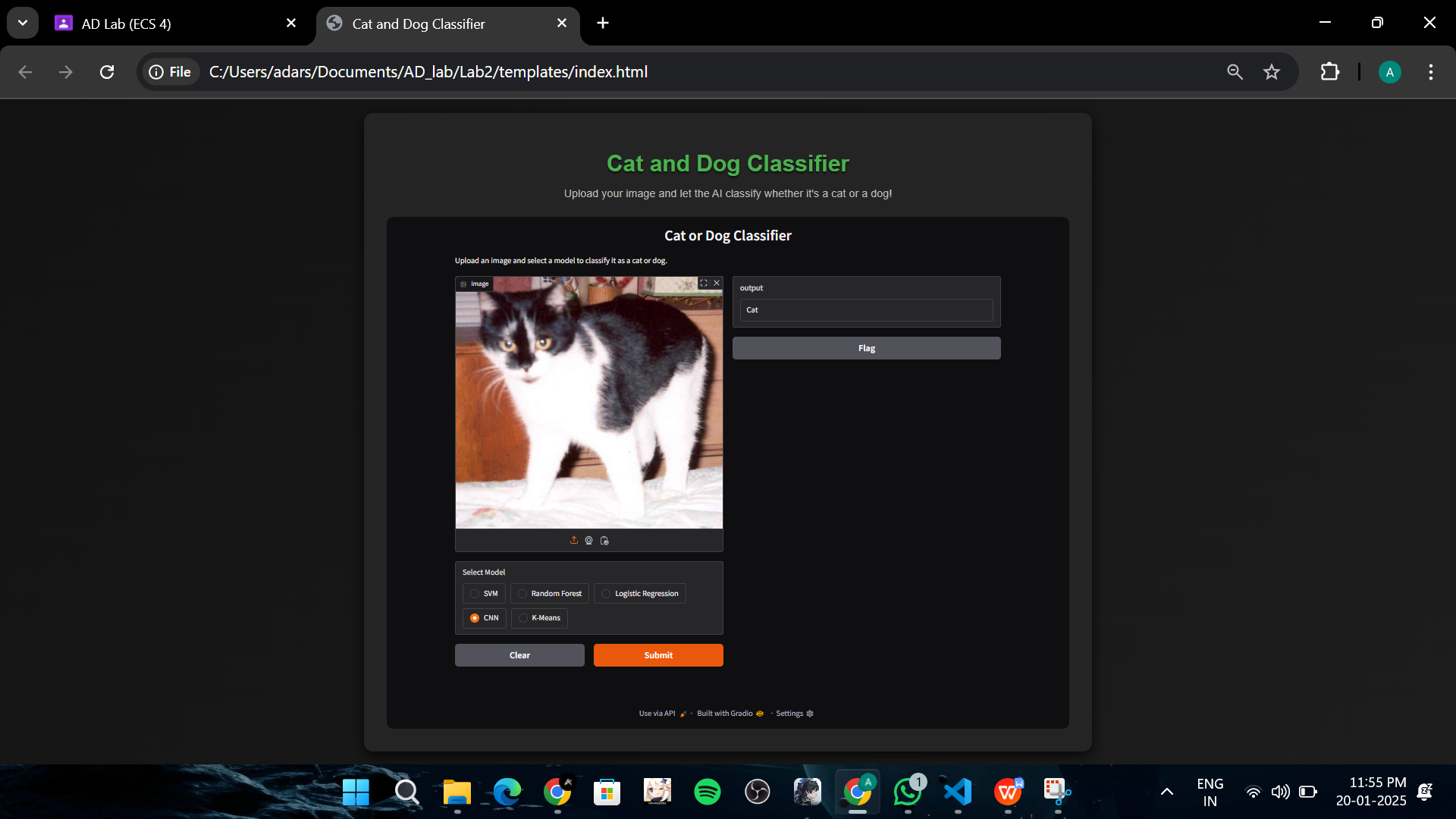
description="Upload an image of a cat or dog and select a model to classify it.",

live=True

)

iface.launch()

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



1. **Remarks:-**

|  |  |
| --- | --- |
| Abhishek Adarsh  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Name of the Student) | Signature of the Lab Coordinator  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Name of the Coordinator) |