LABORATORY REPORT

**Application Development Lab**

**(CS33002)**

**B.Tech Program in ECSc**

Submitted By

**Name:-** RITESH PARVAT

**Roll No:** 2230274



**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. | To design and develop a professional resume using HTML and CSS. | 07-01-2025 | 14-01-2025 |  |
| 2. | To design and develop Machine Learning model for Cat and Dog Classification | 14/01/2025 | 23/01/2025 |  |
| 3. |  |  |  |  |
| 4. |  |  |  |  |
| 5. |  |  |  |  |
| 6. |  |  |  |  |
| 7. |  |  |  |  |
| 8. |  |  |  |  |
| 9. | Open Ended 1 |  |  |  |
| 10. | Open Ended 2 |  |  |  |

|  |  |
| --- | --- |
| **Experiment Number** | 1 |
| **Experiment Title** | Design and develop a professional resume using HTML and CSS. |
| **Date of Experiment** | 07/01/2025 |
| **Date of Submission** | 14/01/2025 |

1. **Objective:-** To design and develop a professional resume using HTML and CSS.
2. **Procedure:- (Steps Followed)**
3. Open a new project folder and create an `index.html` file for the structure of your resume.
4. Write HTML code for sections: Header (Name, Photo, Contact), Skills, Projects, Work-experience, and Education.
5. Add hyperlinks to Linked In, GitHub, and other online portfolios.
6. Use CSS to apply design elements for proper alignment, spacing, and colors.
7. Make the layout responsive using Flexbox or Grid.
8. Integrate images and video links as required.

7. Test the design across devices.

1. **Code:-**
2. **index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Ritesh Parvat - Resume</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="a4-container">

<!-- Header Section -->

<header class="header">

<h1>Ritesh Parvat</h1>

<div class="contact-info">

<p class="email">

<i class="fas fa-envelope"></i>

<a href="mailto:ritesh.parvat17@gmail.com">ritesh.parvat17@gmail.com</a>

</p>

<p class="phone">

<i class="fas fa-phone-alt"></i>

+91 8800526625

</p>

<p class="linkedin">

<i class="fab fa-linkedin"></i>

<a href="https://www.linkedin.com/in/ritesh-parvat-a50738169/" target="\_blank">LinkedIn Profile</a>

</p>

</div>

<div class="image-container">

<img src="utils/profile.jpg" alt="Profile Picture">

</div>

</header>

<!-- Education Section -->

<section class="education">

<h2 class="heading">EDUCATION</h2>

<hr>

<div class="education-entry">

<div class="institution">

<span class="name">Kalinga Institute of Industrial Technology</span>

<span class="location">Bhubaneswar, India</span>

</div>

<div class="details">

<span class="degree">B.Tech in Electronics and Computer Science Engineering; CGPA: 8.51</span>

<span class="year">2022 - 2026</span>

</div>

</div>

<div class="education-entry">

<div class="institution">

<span class="name">Ashok Memorial Public School</span>

<span class="location">Faridabad, Haryana</span>

</div>

<div class="details">

<span class="degree">CBSE 12th; Percentage: 86%</span>

<span class="year">2021</span>

</div>

</div>

<div class="education-entry">

<div class="institution">

<span class="name">Ashok Memorial Public School</span>

<span class="location">Faridabad, Haryana</span>

</div>

<div class="details">

<span class="degree">CBSE 10th; Percentage: 77%</span>

<span class="year">2019</span>

</div>

</div>

</section>

<!-- Projects Section -->

<section class="projects">

<h2 class="heading">PROJECTS</h2>

<hr>

<ul>

<li>

<h3 class="project-heading">

House Price Prediction

<a href="https://github.com/Ritesh17-stack/House-Price-Prediction" target="\_blank">

<i class="fas fa-external-link-alt"></i>

</a>

</h3>

<p class="description">

Developed a pipeline regression model integrated with Flask to predict house prices based on key

features such as area, number of bedrooms and bathrooms, locality, and year built. Implemented

efficient data preprocessing techniques like scaling and one-hot encoding to improve model

accuracy.

</p>

</li>

<li>

<h3 class="project-heading">

Iris Classification

<a href="https://github.com/Ritesh17-stack/House-Price-Prediction" target="\_blank">

<i class="fas fa-external-link-alt"></i>

</a>

</h3>

<p class="description">

Developed a machine learning model for Iris Classification using Logistic Regression, applying

feature engineering, model training, and evaluation techniques to predict the species of Iris

flowers based on their petal and sepal measurements. Achieved high accuracy in classification,

demonstrating proficiency in data analysis and model optimization.

</p>

</li>

</ul>

</section>

<!-- Experience Section -->

<section class="experience">

<h2 class="heading">EXPERIENCE</h2>

<hr>

<div class="experience-entry">

<div class="company">

<span class="name">

AI-ML Virtual Internship

<a href="https://drive.google.com/file/d/1yDmjpdOW7\_zJhcskaRPqlD8ov8sr6Moy/view"

target="\_blank">

<i class="fas fa-external-link-alt"></i>

</a>

</span>

<span class="location">Virtual</span>

</div>

<div class="about">

<span class="company-name">AICTE - Google for Developers</span>

<span class="year">July 2024 - Sep 2024</span>

</div>

<div class="details">

<ul>

<li>Gained hands-on experience in developing AI/ML solutions by working on projects involving

image recognition, object detection, and classification.</li>

<li>Built practical skills in TensorFlow, machine learning workflows, and AI tools.</li>

<li>Enhanced my ability to design, train, and deploy models for real-world applications.</li>

</ul>

</div>

</div>

<div class="experience-entry">

<div class="company">

<span class="name">

Internship Trainee

<a href="https://drive.google.com/file/d/1yDmjpdOW7\_zJhcskaRPqlD8ov8sr6Moy/view"

target="\_blank">

<i class="fas fa-external-link-alt"></i>

</a>

</span>

<span class="location">Virtual</span>

</div>

<div class="about">

<span class="company-name">Bharat Intern</span>

<span class="year">June 2024 - July 2024</span>

</div>

<div class="details">

<ul>

<li>Gained hands-on experience in feature engineering, model training, and evaluation for

classification tasks.</li>

<li>Designed and implemented robust machine learning pipelines with efficient preprocessing

techniques, including scaling

and one-hot encoding.</li>

<li>Built regression models for predictive analysis and integrated them into web applications

using Flask for real-time

deployment.</li>

<li>Strengthened expertise in data preprocessing, algorithm optimization, and deploying AI/ML

solutions for real-world

scenarios</li>

</ul>

</div>

</div>

<div class="experience-entry">

<div class="company">

<span class="name">

Internship Trainee

<a href="https://drive.google.com/file/d/1yDmjpdOW7\_zJhcskaRPqlD8ov8sr6Moy/view"

target="\_blank">

<i class="fas fa-external-link-alt"></i>

</a>

</span>

<span class="location">Ahmedabad, Gujarat</span>

</div>

<div class="about">

<span class="company-name">Nippon Signal Co., Ltd.</span>

<span class="year">April 2024 - June 2024</span>

</div>

<div class="details">

<ul>

<li>Acquired in-depth knowledge of CBTC system architecture, functionality, and equipment, including Point Machines,

KLCR, ESP, SPK, Axle Counters, and Balises.</li>

<li>Gained practical understanding of wayside, on-board, and station equipment roles and operations.</li>

<li>Trained in safety protocols, emergency procedures, and effective CBTC system integration with rail networks.</li>

</ul>

</div>

</div>

</section>

<!-- Core Subject Proficiency Section -->

<section class="core">

<h2 class="heading">CORE SUBJECT PROFICIENCY</h2>

<hr>

<ul>

<li>Design and Analysis of Algorithms</li>

<li>Database Management System</li>

<li>Microprocessor & Embedded Systems</li>

<li>Operating Systems</li>

</ul>

</section>

<!-- Skills Section -->

<section class="skills">

<h2 class="heading">SKILLS</h2>

<hr>

<ul>

<li>Programming Languages: C, Java, Python, HTML, CSS, SQL</li>

<li>Developer Tools: Git, VS Code, Jupyter Notebook, MySQL</li>

<li>Frameworks: Flask</li>

<li>Libraries: Pandas, NumPy, Matplotlib, scikit-learn, tensorflow</li>

</ul>

</section>

<!-- Achievements Section -->

<section class="achievements">

<h2 class="heading">ACHIEVEMENTS</h2>

<hr>

<ul>

<li>Built my first AI model during a Generative AI workshop by NxtWave, gaining hands-on experience and

practical knowledge in AI concepts.</li>

</ul>

</section>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"

crossorigin="anonymous"></script>

</body>

</html>

1. **styles.css**

html {

height: 80%;

margin: 0;

}

body {

display: flex;

justify-content: center;

min-height: 100vh;

}

.a4-container {

width: 210mm;

min-height: 297mm;

margin: 20px auto;

padding: 15mm;

background-color: white;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

box-sizing: border-box;

font-size: 10pt;

}

.header {

margin-bottom: 15px;

display: flex;

justify-content: space-between;

gap: 15px;

}

.header-content {

flex: 1;

}

.header h1 {

margin: 0 0 8px 0;

font-size: 18pt;

font-weight: bold;

}

.contact-info {

font-size: 9pt;

margin-left: -557px;

margin-top: 40px;

gap: 4px;

}

.contact-info p {

margin: 0;

display: flex;

align-items: center;

gap: 4px;

}

.image-container {

width: 100px;

height: 100px;

flex-shrink: 0;

}

.image-container img {

width: 100%;

height: 100%;

object-fit: cover;

object-position: center;

}

a {

color: inherit;

text-decoration: none;

}

a:hover {

text-decoration: underline;

}

.heading {

font-size: 12pt;

margin: 8px 0 4px;

font-weight: bold;

}

hr {

border: 0;

border-top: 1px solid black;

margin: 4px 0;

}

ul {

padding-left: 12px;

margin: 5px 0;

}

ul li {

font-size: 9pt;

margin-bottom: 6px;

line-height: 1.3;

}

.education,

.experience,

.projects,

.skills,

.core,

.achievements {

margin-top: 12px;

}

.education-entry,

.experience-entry {

margin-bottom: 10px;

}

.institution,

.company {

display: flex;

justify-content: space-between;

font-size: 10pt;

font-weight: bold;

margin-bottom: 5px;

}

.institution .location,

.company .location {

font-style: italic;

font-weight: lighter;

color: #555;

}

.details,

.about {

display: flex;

justify-content: space-between;

font-size: 9pt;

color: #333;

}

.details .year,

.about .year {

font-style: italic;

}

.project-heading {

font-size: 10pt;

font-weight: bold;

margin-bottom: 3px;

}

.description {

font-size: 9pt;

line-height: 1.3;

}

.projects ul,

.skills ul,

.core ul,

.achievements ul {

list-style-type: none;

padding-left: 0;

}

.skills li,

.core li,

.achievements li {

margin-bottom: 5px;

}

.experience-entry .details ul {

margin-top: 3px;

margin-left:19px;

}

@media print {

body {

background: none;

}

.a4-container {

margin: 0;

padding: 15mm;

box-shadow: none;

width: 210mm;

height: 100px;

}

}

@media screen and (max-width: 210mm) {

.a4-container {

margin: 0;

width: 100%;

min-height: auto;

}

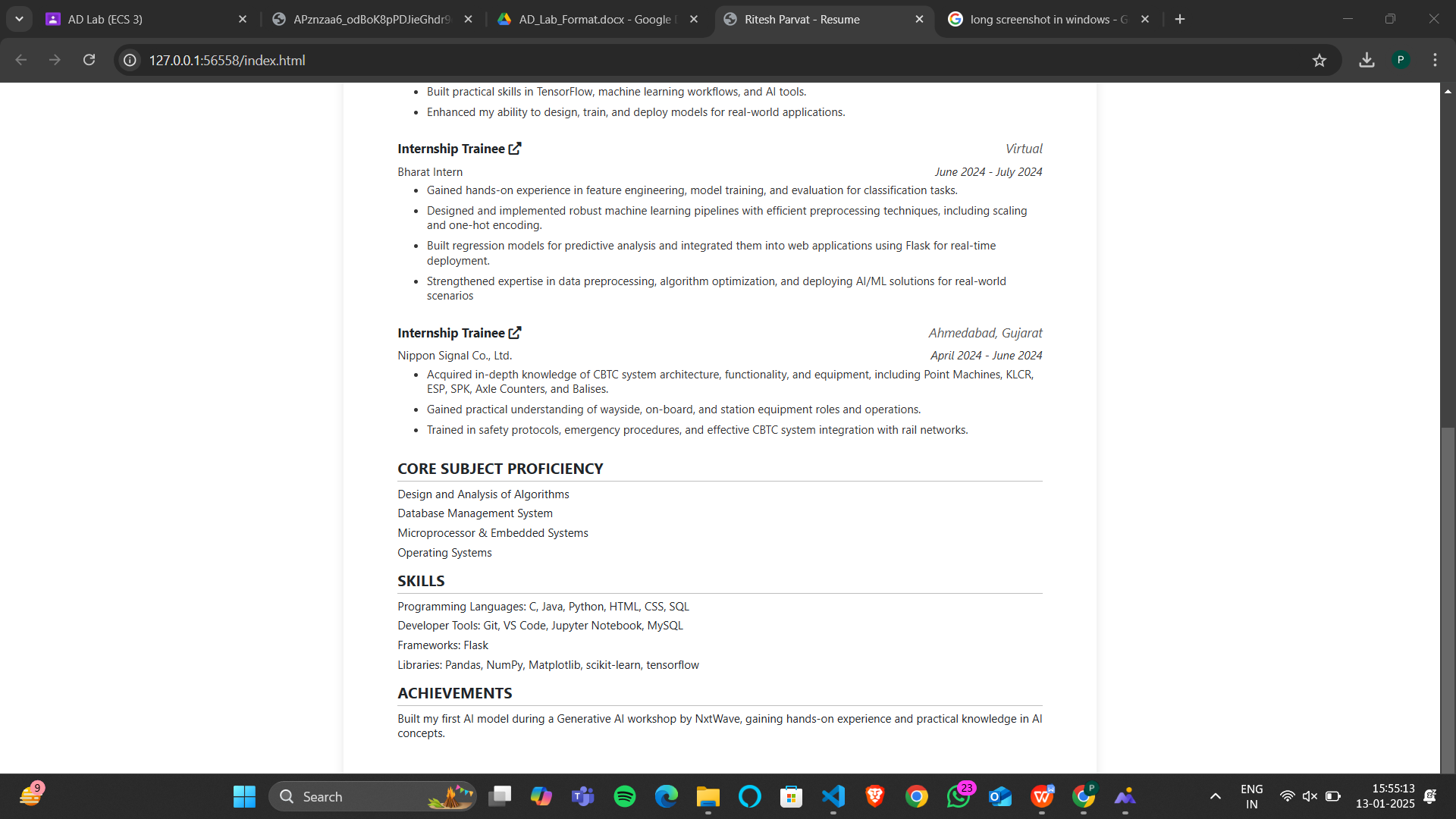
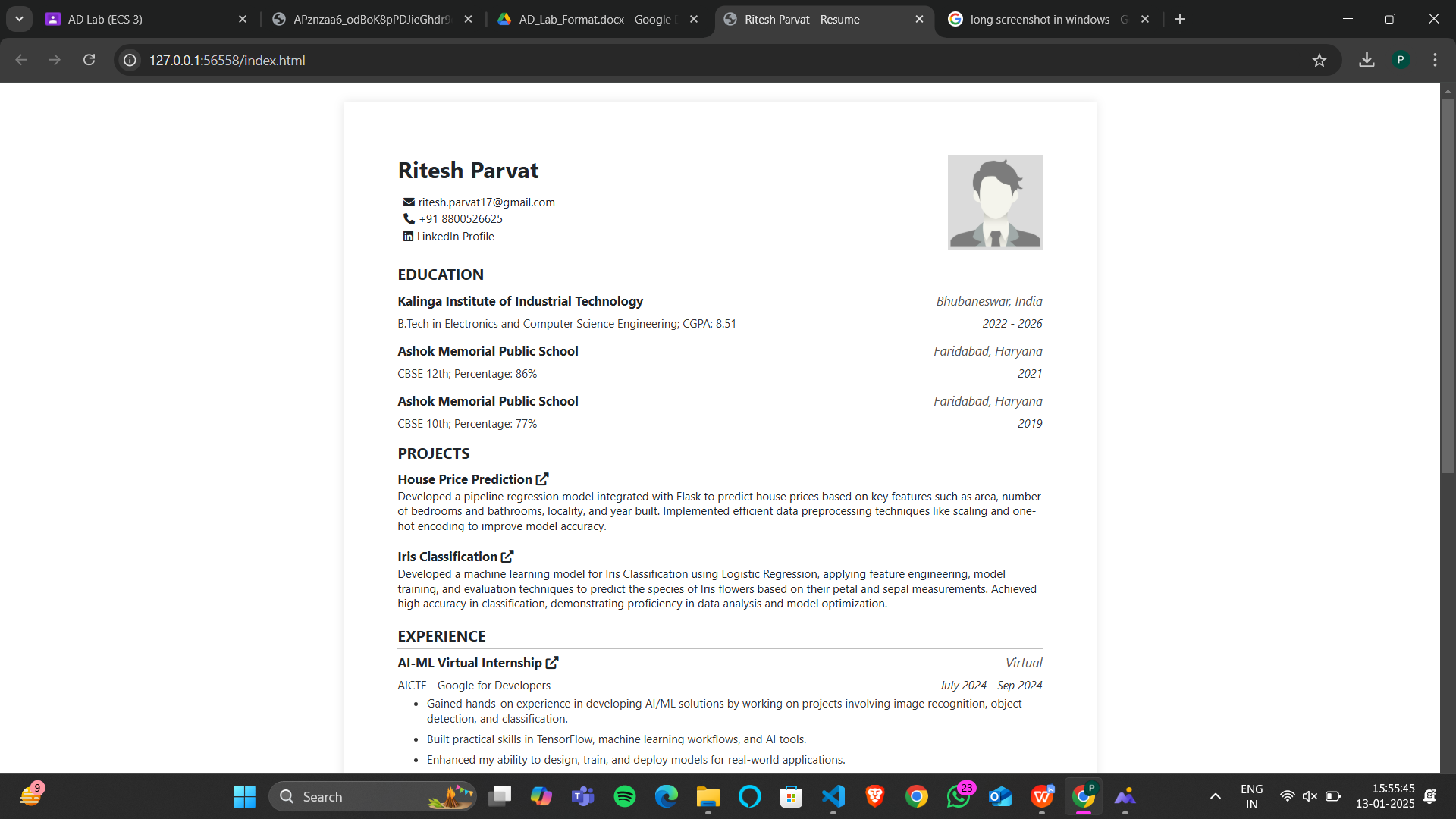
}

.fa-phone-alt {

transform: rotate(90deg);

}

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



1. **Remarks:-**

|  |  |
| --- | --- |
| Signature of the Student  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ritesh Parvat) | Signature of the Lab Coordinator  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Name of the Coordinator) |

|  |  |
| --- | --- |
| **Experiment Number** | 2 |
| **Experiment Title** | To design and develop Machine Learning model for Cat and Dog Classification |
| **Date of Experiment** | 14/01/2025 |
| **Date of Submission** | 23/01/2025 |

**Objective:-** To design and develop Machine Learning model for Cat and Dog Classification

**Procedure:-**

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, CNN, 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.

**Code:-**

1. **CNN**

### Importing libraries

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

import os

from keras.utils import to\_categorical

### Loading the dataset

# Paths

train\_path = r'C:\Personal\AD LAB\Cats&Dogs\training\_set'

test\_path = r'C:\Personal\AD LAB\Cats&Dogs\test\_set'

# Function to load images and labels

def load\_data(data\_path):

images, labels = [], []

for label, folder\_name in enumerate(['cats', 'dogs']): # 0 for cats, 1 for dogs

folder = os.path.join(data\_path, folder\_name)

for file\_name in os.listdir(folder):

file\_path = os.path.join(folder, file\_name)

image = cv2.imread(file\_path)

if image is not None:

images.append(cv2.resize(image, (100, 100)))

labels.append(label)

return np.array(images), np.array(labels)

# Load training and test data

X\_train, y\_train = load\_data(train\_path)

X\_test, y\_test = load\_data(test\_path)

# Print the shapes for verification

X\_train=X\_train.astype('float32')

X\_test=X\_test.astype('float32')

print(f"X\_train shape: {X\_train.shape}, y\_train shape: {y\_train.shape}")

print(f"X\_test shape: {X\_test.shape}, y\_test shape: {y\_test.shape}")

X\_train

class\_label = np.unique(y\_train)

num\_classes = class\_label.shape[0]

print(num\_classes)

y\_train = to\_categorical(y\_train, num\_classes)

y\_test = to\_categorical(y\_test, num\_classes)

y\_train

### Normalizing

X\_train /= 255

X\_test /= 255

X\_train

### Model Training

import keras

from keras.models import Sequential

from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout

model = Sequential()

# Convolutional layer

model.add(Conv2D(100, kernel\_size=(3,3), strides=(1,1),padding='valid',activation='relu', input\_shape=(200,200,3)))

#Pooling layer

model.add(MaxPool2D(pool\_size=(2,2)))

# Convolutional Layer

model.add(Conv2D(50, kernel\_size=(3,3), strides=(1,1), padding='valid', activation='relu'))

# Pooling Layer

model.add(MaxPool2D(pool\_size=(2,2)))

# Flatten the output

model.add(Flatten())

model.add(Dropout(0.5))

# Fully Connected Layer

model.add(Dense(50, activation='relu'))

# Output Layer

model.add(Dense(2, activation='softmax'))

model.summary()

# compiling the sequential model

model.compile(loss='categorical\_crossentropy', metrics=['accuracy'], optimizer='adam')

model.fit(X\_train,y\_train, batch\_size=128,epochs=20,validation\_split=0.2)

model.save('model.h5')

from keras.models import load\_model

model=load\_model('conv.h5')

(loss, accuracy) = model.evaluate(X\_test, y\_test, batch\_size=50, verbose=1)

print("[INFO] loss={:.4f}, accuracy: {:.4f}%".format(loss,accuracy \* 100))

model\_df = pd.DataFrame(model.model)

model\_df.loc[:, ['loss', 'val\_loss']].plot()

model\_df.loc[:, ['accuracy', 'val\_accuracy']].plot()

plt.show()

# Test one image from test set

test\_img = X\_test[112,:,:]

# print(img1.shape)

img1 = np.reshape(test\_img, (1, 100,100,3))

#print(img1)

import matplotlib.pyplot as plt

plt.imshow(test\_img)

prediction = model.predict(img1)

class\_labels = ['Cat', 'Dog']

# Get the index of the highest probability

predicted\_index = np.argmax(prediction)

# Get the corresponding label

predicted\_label = class\_labels[predicted\_index]

# Print the result

print(f"Predicted Label: {predicted\_label}")

print(f"Probabilities: Cat={prediction[0][0]:.2f}, Dog={prediction[0][1]:.2f}")

**2)K-means**

### Importing libraries

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

import os

from keras.utils import to\_categorical

from sklearn.cluster import KMeans

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

import numpy as np

### Loading the dataset

# Paths

train\_path = r'C:\Personal\AD LAB\Cats&Dogs\training\_set'

test\_path = r'C:\Personal\AD LAB\Cats&Dogs\test\_set'

# Function to load images and labels

def load\_data(data\_path):

images, labels = [], []

for label, folder\_name in enumerate(['cats', 'dogs']): # 0 for cats, 1 for dogs

folder = os.path.join(data\_path, folder\_name)

for file\_name in os.listdir(folder):

file\_path = os.path.join(folder, file\_name)

image = cv2.imread(file\_path)

if image is not None:

images.append(cv2.resize(image, (100, 100)))

labels.append(label)

return np.array(images), np.array(labels)

# Load training and test data

X\_train, y\_train = load\_data(train\_path)

X\_test, y\_test = load\_data(test\_path)

# Print the shapes for verification

X\_train=X\_train.astype('float32')

X\_test=X\_test.astype('float32')

print(f"X\_train shape: {X\_train.shape}, y\_train shape: {y\_train.shape}")

print(f"X\_test shape: {X\_test.shape}, y\_test shape: {y\_test.shape}")

# Flatten the image data

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

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

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train\_flat)

X\_test\_scaled = scaler.transform(X\_test\_flat)

# Initialize and train K-Means

kmeans = KMeans(n\_clusters=2, random\_state=0) # 2 clusters for binary classification

kmeans.fit(X\_train\_scaled)

# Map cluster assignments to original labels

def map\_cluster\_labels(cluster\_labels, true\_labels):

"""Maps cluster labels to true labels based on majority class in each cluster"""

mapping = {}

for cluster in np.unique(cluster\_labels):

mask = cluster\_labels == cluster

if len(true\_labels[mask]) > 0:

mapping[cluster] = mode(true\_labels[mask])[0]

return np.array([mapping[label] for label in cluster\_labels])

# Evaluate clustering performance

def evaluate\_clustering(kmeans, X\_train\_scaled, X\_test\_scaled, y\_train, y\_test):

"""Evaluates K-means clustering performance"""

# Get cluster predictions

train\_clusters = kmeans.predict(X\_train\_scaled)

test\_clusters = kmeans.predict(X\_test\_scaled)

# Map clusters to true labels

train\_pred = map\_cluster\_labels(train\_clusters, y\_train)

test\_pred = map\_cluster\_labels(test\_clusters, y\_test)

# Calculate accuracy

train\_acc = accuracy\_score(y\_train, train\_pred)

test\_acc = accuracy\_score(y\_test, test\_pred)

return train\_acc, test\_acc

# Use the functions

train\_accuracy, test\_accuracy = evaluate\_clustering(

kmeans, X\_train\_scaled, X\_test\_scaled, y\_train, y\_test

)

print(f"Training Accuracy: {train\_accuracy:.3f}")

print(f"Test Accuracy: {test\_accuracy:.3f}")

def predict\_image(img\_index):

"""Predict single image classification"""

# Prepare image

test\_img = X\_test[img\_index]

test\_data = test\_img.reshape(1, -1)

test\_data\_scaled = scaler.transform(test\_data)

# Get prediction

cluster\_pred = kmeans.predict(test\_data\_scaled)

# Map cluster to label

train\_clusters = kmeans.predict(X\_train\_scaled)

cluster\_mask = train\_clusters == cluster\_pred[0]

true\_label = mode(y\_train[cluster\_mask])[0]

# Display results

plt.figure(figsize=(6, 6))

plt.imshow(test\_img)

plt.axis('off')

plt.title('Test Image')

print("\nPrediction:", "cat" if true\_label == 0 else "dog")

# Example usage

predict\_image(1102)

X\_test[100]

from joblib import dump, load

# Save the model

dump(kmeans, 'kmeans.joblib')

print("Kmeans saved as kmeans.joblib")

**3)Logistic Regression**

### Importing libraries

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

import os

from keras.utils import to\_categorical

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

### Loading the dataset

# Paths

train\_path = r'C:\Personal\AD LAB\Cats&Dogs\training\_set'

test\_path = r'C:\Personal\AD LAB\Cats&Dogs\test\_set'

# Function to load images and labels

def load\_data(data\_path):

images, labels = [], []

for label, folder\_name in enumerate(['cats', 'dogs']): # 0 for cats, 1 for dogs

folder = os.path.join(data\_path, folder\_name)

for file\_name in os.listdir(folder):

file\_path = os.path.join(folder, file\_name)

image = cv2.imread(file\_path)

if image is not None:

images.append(cv2.resize(image, (100, 100)))

labels.append(label)

return np.array(images), np.array(labels)

# Load training and test data

X\_train, y\_train = load\_data(train\_path)

X\_test, y\_test = load\_data(test\_path)

# Print the shapes for verification

X\_train=X\_train.astype('float32')

X\_test=X\_test.astype('float32')

print(f"X\_train shape: {X\_train.shape}, y\_train shape: {y\_train.shape}")

print(f"X\_test shape: {X\_test.shape}, y\_test shape: {y\_test.shape}")

# Flattening the image data for SVM

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

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

X\_train.shape

# Initializing and training Logistic Regression model

model = LogisticRegression(penalty='l2',

C=1.0,

solver='lbfgs',

max\_iter=1000,

random\_state=42)

model.fit(X\_train, y\_train)

# Making predictions on the test set

y\_pred = model.predict(X\_test)

# Calculating accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Test Accuracy with Logistic\_Regression: {accuracy \* 100:.2f}%")

1. **Random Forest**

### Importing libraries

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

import os

from keras.utils import to\_categorical

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

### Loading the dataset

# Paths

train\_path = r'C:\Personal\AD LAB\Cats&Dogs\training\_set'

test\_path = r'C:\Personal\AD LAB\Cats&Dogs\test\_set'

# Function to load images and labels

def load\_data(data\_path):

images, labels = [], []

for label, folder\_name in enumerate(['cats', 'dogs']): # 0 for cats, 1 for dogs

folder = os.path.join(data\_path, folder\_name)

for file\_name in os.listdir(folder):

file\_path = os.path.join(folder, file\_name)

image = cv2.imread(file\_path)

if image is not None:

images.append(cv2.resize(image, (100, 100)))

labels.append(label)

return np.array(images), np.array(labels)

# Load training and test data

X\_train, y\_train = load\_data(train\_path)

X\_test, y\_test = load\_data(test\_path)

# Print the shapes for verification

X\_train=X\_train.astype('float32')

X\_test=X\_test.astype('float32')

print(f"X\_train shape: {X\_train.shape}, y\_train shape: {y\_train.shape}")

print(f"X\_test shape: {X\_test.shape}, y\_test shape: {y\_test.shape}")

# Flattening the image data for SVM

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

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

X\_train.shape

y\_train

X\_train /= 255

X\_test /= 255

X\_train

# Initialize and train Random Forest model

rf\_model = RandomForestClassifier(

n\_estimators=100, # Number of trees in the forest

max\_depth=None, # Let the trees grow until pure or max features split

random\_state=42, # Reproducibility

n\_jobs=-1 # Use all available processors

)

rf\_model.fit(X\_train, y\_train)

# Calculating accuracy

y\_pred = rf\_model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Test Accuracy with Random Forest: {accuracy \* 100:.2f}%")

from joblib import dump, load

# Save the model

dump(rf\_model, 'rf\_model.joblib')

print("RF model saved as rf\_model.joblib")

import joblib

from joblib import dump, load

model = joblib.load('rf\_model.joblib')

# Select the test image

test\_img = X\_test[12, :, :]

# Reshape the image for prediction (same as your input shape)

test\_data = np.reshape(test\_img, (1, 100, 100, 3))

# Display the image

plt.imshow(test\_img)

# Flatten the image for model prediction

test\_data\_flattened = test\_data.reshape(1, -1)

def print\_animal(prediction):

"""Prints "cat" for 0 and "dog" for 1."""

if prediction == 0:

print("cat")

elif prediction == 1:

print("dog")

else:

print("Invalid prediction value.")

# Predict using the model

prediction = model.predict(test\_data\_flattened)

print\_animal(prediction)

1. **SVM**

### Importing libraries

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

import os

from sklearn.decomposition import PCA

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

import joblib

### Loading the dataset

# Paths

train\_path = r'C:\Personal\AD LAB\Cats&Dogs\training\_set'

test\_path = r'C:\Personal\AD LAB\Cats&Dogs\test\_set'

# Function to load images and labels

def load\_data(data\_path):

images, labels = [], []

for label, folder\_name in enumerate(['cats', 'dogs']): # 0 for cats, 1 for dogs

folder = os.path.join(data\_path, folder\_name)

for file\_name in os.listdir(folder):

file\_path = os.path.join(folder, file\_name)

image = cv2.imread(file\_path)

if image is not None:

images.append(cv2.resize(image, (100, 100)))

labels.append(label)

return np.array(images), np.array(labels)

# Load training and test data

X\_train, y\_train = load\_data(train\_path)

X\_test, y\_test = load\_data(test\_path)

X\_train.shape

y\_train

# Normalize the data

X\_train = X\_train.astype('float32') / 255.0

X\_test = X\_test.astype('float32') / 255.0

X\_train

# Flatten the data for SVM

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

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

# Dimensionality reduction using PCA

print("Performing PCA for dimensionality reduction...")

pca = PCA(n\_components=100) # Reduce to 100 principal components

X\_train\_pca = pca.fit\_transform(X\_train)

X\_test\_pca = pca.transform(X\_test)

# Train SVM model

print("Training SVM...")

svm = SVC(kernel='linear', probability=True) # Linear kernel for simplicity

svm.fit(X\_train\_pca, y\_train)

# Evaluate the model

y\_pred = svm.predict(X\_test\_pca)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Model accuracy: {accuracy \* 100:.2f}%")

from joblib import dump, load

# Save the model

dump(svm\_model, 'svm\_model.joblib')

print("SVM model saved as svm\_model.joblib")

import joblib

from joblib import dump, load

model = joblib.load('svm\_model.joblib')

# Select the test image

test\_img = X\_test[110, :, :]

# Reshape the image for prediction (same as your input shape)

test\_data = np.reshape(test\_img, (1, 100, 100, 3))

# Display the image

plt.imshow(test\_img)

# Flatten the image for model prediction

test\_data\_flattened = test\_data.reshape(1, -1)

def print\_animal(prediction):

"""Prints "cat" for 0 and "dog" for 1."""

if prediction == 0:

print("cat")

elif prediction == 1:

print("dog")

else:

print("Invalid prediction value.")

# Predict using the model

prediction = model.predict(test\_data\_flattened)

print\_animal(prediction)

**Website using flask**

1. **app.py**

import os

os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = '0'

from flask import Flask, render\_template, request

import os

from werkzeug.utils import secure\_filename

from PIL import Image

import numpy as np

import joblib

from keras.models import load\_model

# Load models

cnn\_model = load\_model('conv.h5')

kmeans\_model = joblib.load('kmeans.joblib')

lr\_model = joblib.load('lr\_model.joblib')

rf\_model = joblib.load('rf\_model.joblib')

svm\_model = joblib.load('svm\_model.joblib')

# Placeholder functions for model predictions

def predict\_with\_cnn(image\_path):

image = Image.open(image\_path).resize((100, 100))

image\_array = np.array(image).astype('float32') / 255.0

image\_array = image\_array.reshape(1, 100, 100, 3)

prediction = cnn\_model.predict(image\_array)

return "Cat" if prediction[0][0] > 0.5 else "Dog"

def predict\_with\_logistic\_regression(image\_path):

image = Image.open(image\_path).resize((300, 100)).convert('L') # Resize to 300x100

image\_array = np.array(image).reshape(1, 30000).astype('float32') / 255.0 # Flatten to match 30000 features

prediction = lr\_model.predict(image\_array)

return "Cat" if prediction[0] == 0 else "Dog"

def predict\_with\_kmeans(image\_path):

image = Image.open(image\_path).resize((100, 100)).convert('L')

image\_array = np.array(image).reshape(1, -1).astype('float32') / 255.0

cluster = kmeans\_model.predict(image\_array)

return "Cat" if cluster[0] == 0 else "Dog"

def predict\_with\_random\_forest(image\_path):

image = Image.open(image\_path).resize((100, 100)).convert('L')

image\_array = np.array(image).reshape(1, -1).astype('float32') / 255.0

prediction = rf\_model.predict(image\_array)

return "Cat" if prediction[0] == 0 else "Dog"

def predict\_with\_svm(image\_path):

image = Image.open(image\_path).resize((100, 100)).convert('L')

image\_array = np.array(image).reshape(1, -1).astype('float32') / 255.0

prediction = svm\_model.predict(image\_array)

return "Cat" if prediction[0] == 0 else "Dog"

# Mapping model names to prediction functions

model\_functions = {

"cnn": predict\_with\_cnn,

"logistic\_regression": predict\_with\_logistic\_regression,

"kmeans": predict\_with\_kmeans,

"random\_forest": predict\_with\_random\_forest,

"svm": predict\_with\_svm

}

# Initialize Flask app

app = Flask(\_\_name\_\_)

# Configure upload folder

UPLOAD\_FOLDER = "static/uploads"

os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

if 'file' not in request.files:

return "No file part"

file = request.files['file']

if file.filename == '':

return "No selected file"

if file:

filename = secure\_filename(file.filename)

filepath = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

file.save(filepath)

# Get selected model

selected\_model = request.form.get('model')

if selected\_model not in model\_functions:

return "Invalid model selected"

# Predict using the selected model

prediction\_function = model\_functions[selected\_model]

prediction = prediction\_function(filepath)

return render\_template('result.html', prediction=prediction, image\_path=filepath)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

1. **Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dog and Cat Classifier</title>

<link rel="stylesheet" href="/static/styles.css">

</head>

<body>

<div class="container">

<h1>Dog and Cat Classifier</h1>

<form action="/predict" method="post" enctype="multipart/form-data">

<label for="model">Select Model:</label>

<select name="model" id="model">

<option value="cnn">CNN</option>

<option value="logistic\_regression">Logistic Regression</option>

<option value="kmeans">KMeans</option>

<option value="random\_forest">Random Forest</option>

<option value="svm">SVM</option>

</select>

<br><br>

<label for="file">Upload an Image:</label>

<input type="file" name="file" id="file" accept="image/\*" required>

<br><br>

<button type="submit">Classify</button>

</form>

</div>

</body>

</html>

1. **result.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Classification Result</title>

<link rel="stylesheet" href="/static/styles.css">

</head>

<body>

<div class="container">

<h1>Classification Result</h1>

<p><strong>Prediction:</strong> {{ prediction }}</p>

<img src="{{ image\_path }}" alt="Uploaded Image" class="result-image">

<br><br>

<a href="/">Back to Home</a>

</div>

</body>

</html>

1. **styles.css**

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f4f4f9;

color: #333;

}

.container {

margin: 50px auto;

padding: 20px;

max-width: 500px;

background: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 8px;

}

h1 {

color: #555;

}

label {

display: block;

margin: 10px 0 5px;

}

input, select, button {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

background-color: #5cb85c;

color: white;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #4cae4c;

}

.result-image {

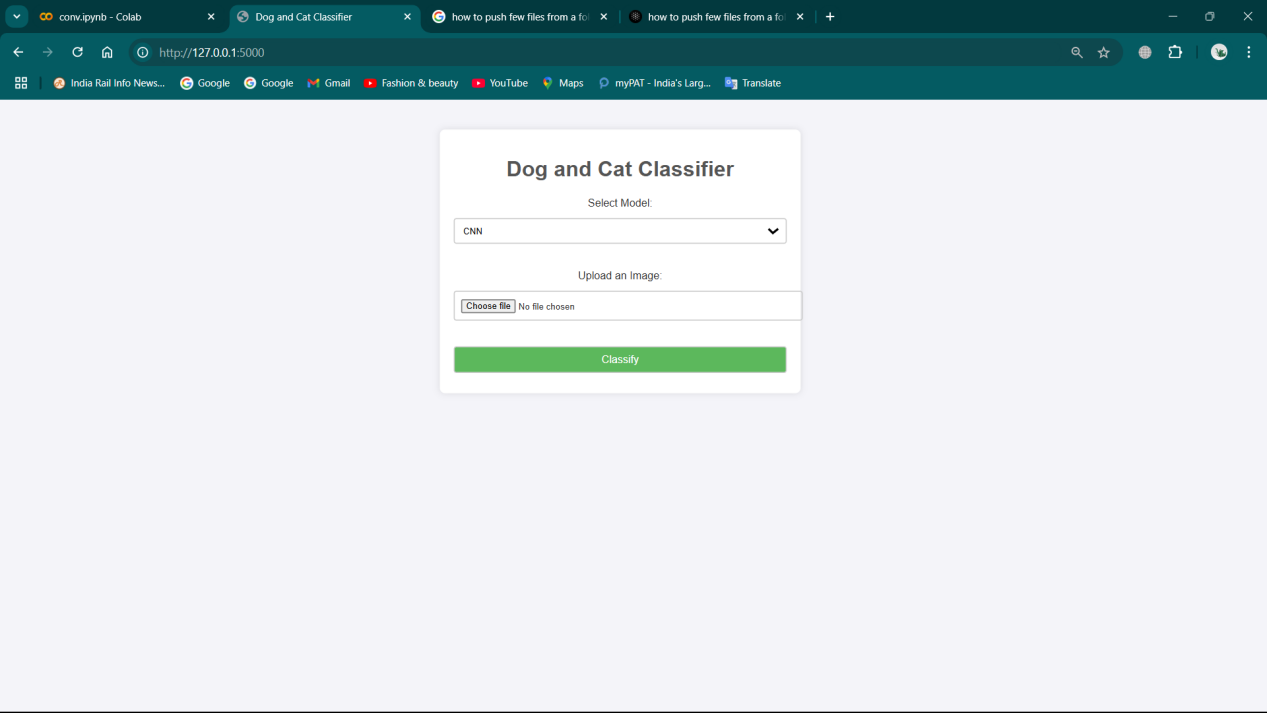
max-width: 100%;

height: auto;

margin-top: 20px;

}

**Results/Output:-**



**Remarks:-**

From this experiment I learned how to perform classification using machine learning models and host the using flask.

|  |  |
| --- | --- |
| Signature of the Student  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ritesh Parvat) | Signature of the Lab Coordinator  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Name of the Coordinator) |