1. create_dataset_csv_file.py

```
import numpy as np
import sys
import os
import random

def create_dataset_file(myDir, is_tp_files):
    if is_tp_files:
        data_csv_file = open("F:\\HEMANT\\Task\\Image Forgery Detection\\IF1\\ForgedImages.csv", "w")
    else:
        data_csv_file = open("F:\\HEMANT\\Task\\Image Forgery Detection\\IF1\\AuthenticImages.csv", "w")
    format_ = ['.JPG' , '.jpg', 'jpeg', '.png', '.tiff', '.TIFF', '.Tiff', '.Tif', '.TIF', '.tif']
    print(myDir)
    for root, dirs, files in os.walk(myDir, topdown=False):
        for name in files:
            for type_ in format_:
                if name.endswith(type_):
                    fullName = os.path.join(root, name)
                    data_csv_file.write("%s\n" % (fullName))
                    break

    data_csv_file.close()

# load the original image
create_dataset_file('F:\\HEMANT\\Task\\Image Forgery Detection\\IF1\\dataset\\CASIA2\\Au\\', 0)
create_dataset_file('F:\\HEMANT\\Task\\Image Forgery Detection\\IF1\\dataset\\CASIA2\\Tp\\', 1)
```

2. Image Forgery Detection – GUI.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
np.random.seed(2)
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
#from keras.optimizers import RMSprop
from tensorflow.keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
sns.set(style='white', context='notebook', palette='deep')

from PIL import Image
import os
from pylab import *
import re
from PIL import Image, ImageChops, ImageEnhance

from tkinter import *
import PIL.Image
import PIL.ImageTk
from tkinter.filedialog import askopenfilename
from PIL import Image, ImageChops, ImageEnhance
```

```
def train():

  def get_imlist(path):
      return [os.path.join(path,f) for f in os.listdir(path) if f.endswith('.jpg') or f.endswith('.png')]

  def convert_to_ela_image(path, quality):
      filename = path
      resaved_filename = filename.split('.')[0] + '.resaved.jpg'
      ELA_filename = filename.split('.')[0] + '.ela.png'

      im = Image.open(filename).convert('RGB')
      im.save(resaved_filename, 'JPEG', quality=quality)
      resaved_im = Image.open(resaved_filename)

      ela_im = ImageChops.difference(im, resaved_im)

      extrema = ela_im.getextrema()
      max_diff = max([ex[1] for ex in extrema])
      if max_diff == 0:
          max_diff = 1
      scale = 255.0 / max_diff

      ela_im = ImageEnhance.Brightness(ela_im).enhance(scale)

      return ela_im

  #Read dataset
  dataset = pd.read_csv('dataset.csv')

  X = []
```

```
Y = []

for index, row in dataset.iterrows():
    X.append(array(convert_to_ela_image(row[0], 90).resize((128, 128))).flatten() / 255.0)
    Y.append(row[1])

X = np.array(X)
Y = to_categorical(Y, 2)
print(Y)
X = X.reshape(-1, 128, 128, 3)

X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2, random_state=5)

model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                 activation ='relu', input_shape = (128,128,3)))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                 activation ='relu'))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Dropout(0.25))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)
```

```
 model.add(Flatten())
 model.add(Dense(256, activation = "relu"))
 model.add(Dropout(0.5))
 model.add(Dense(2, activation = "softmax"))

 model.summary()

 optimizer = RMSprop(lr=0.0005, rho=0.9, epsilon=1e-08, decay=0.0)

 model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])

 early_stopping = EarlyStopping(monitor='val_acc',
                        min_delta=0,
                        patience=2,
                        verbose=0, mode='auto')

 #Epochs
 epochs = 30
 batch_size = 100

 history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
         validation_data = (X_val, Y_val), verbose = 2, callbacks=[early_stopping])

 #Model save
 #model.save('model.h5')

 print('---------------Saved Model---------------')


#predict the dataset
def predict():
```

```
def get_imlist(path):
    return [os.path.join(path,f) for f in os.listdir(path) if f.endswith('.jpg') or f.endswith('.png')]


def convert_to_ela_image(path, quality):
    filename = path
    resaved_filename = filename.split('.')[0] + '.resaved.jpg'
    ELA_filename = filename.split('.')[0] + '.ela.png'


    im = Image.open(filename).convert('RGB')
    im.save(resaved_filename, 'JPEG', quality=quality)
    resaved_im = Image.open(resaved_filename)


    ela_im = ImageChops.difference(im, resaved_im)


    extrema = ela_im.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff


    ela_im = ImageEnhance.Brightness(ela_im).enhance(scale)


    return ela_im


#file_name = 'authentic0.jpg'
#file_name = 'F:\\HEMANT\\Task\\Image Forgery Detection\\IF1\\examples\\Au\\authentic6'


from tkinter.filedialog import askopenfilename
file_name = askopenfilename(title='Select image file for analysis ',filetypes=[('image files', '.jpg')])
```

```
 X = []
 X.append(array(convert_to_ela_image(file_name, 90).resize((128, 128))).flatten() / 255.0)
 X = np.array(X)
 X = X.reshape(-1, 128, 128, 3)

 #Prediction
 from keras.models import load_model
 model = load_model('model.h5')

 Y_pred = model.predict(X)
 #print(Y_pred)
 print('------------------------------Prediction Started-------------------------------------')
 print('\nImage is:\n')
 if(Y_pred[0][0]>=Y_pred[0][1]):
   print("Authentic")
 else:
   print("Forged")

#added content for the frontend
from tkinter import *
from tkinter.filedialog import askopenfilename
from PIL import Image, ImageChops, ImageEnhance, ImageTk
import numpy as np
from keras.models import load_model

# Function to convert image to ELA format for forgery detection
def convert_to_ela_image(path, quality):
  filename = path
  resaved_filename = filename.split('.')[0] + '.resaved.jpg'
  ELA_filename = filename.split('.')[0] + '.ela.png'
```

```
    im = Image.open(filename).convert('RGB')
    im.save(resaved_filename, 'JPEG', quality=quality)
    resaved_im = Image.open(resaved_filename)

    ela_im = ImageChops.difference(im, resaved_im)

    extrema = ela_im.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff

    ela_im = ImageEnhance.Brightness(ela_im).enhance(scale)

    return ela_im

# Predict function with updated GUI result display
def predict():
    # Ask user to select an image
    file_name = askopenfilename(title='Select image file for analysis', filetypes=[('image files', '.jpg')])

    if file_name:
        # Convert the selected image to ELA format
        X = []
        X.append(np.array(convert_to_ela_image(file_name, 90).resize((128, 128))).flatten() / 255.0)
        X = np.array(X)
        X = X.reshape(-1, 128, 128, 3)

        # Load the trained model
        model = load_model('model.h5')
        # Predict the image class (Authentic or Forged)
        Y_pred = model.predict(X)

        # Display the result in a pop-up window
        result = "Authentic" if Y_pred[0][0] >= Y_pred[0][1] else "Forged"

        # Create a new pop-up window for displaying the result
        result_window = Toplevel(window9)
        result_window.geometry('500x600')
        result_window.title("Prediction Result")

        # Display the selected image in the result window
        img = Image.open(file_name)
        img = img.resize((300, 300))
        img_tk = ImageTk.PhotoImage(img)

        img_label = Label(result_window, image=img_tk)
        img_label.image = img_tk  # Keep a reference to avoid garbage collection
        img_label.pack(pady=20)

        # Display the prediction result in the result window
        result_label = Label(result_window, text=f"Prediction: {result}", font=('Times New Roman', 20), bg="white", fg="black")
        result_label.pack(pady=10)

# Function to open Contact Us page
def open_contact_us():
    # Create a new window for Contact Us page
    contact_window = Toplevel(window9)
    contact_window.geometry('800x600')
    contact_window.title("Contact Us")
    # Add a label with contact information
    label = Label(contact_window, text="Developed by : ", font=('algerian', 30, 'bold'))
    label.pack(pady=20)

    # Display some contact information
    contact_info = Label(contact_window, text="Vinod S - 1DB22CS417\n Sanjay S M - 1DB22CS411\n Vishal N - 1DB21CS171\n Sahana Devi - 1DB21CS185", font=('algerian', 16))
    contact_info.pack(pady=10)

    # Add a button to close the contact window
    close_button = Button(contact_window, text="Close", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue", command=contact_window.destroy)
    close_button.pack(pady=20)

def open_vision():
    # Create a new window for Vision page
    vision_window = Toplevel(window9)
    vision_window.geometry('800x600')



    # Add a label with Vision information
    label = Label(vision_window, text="Vision : ", font=('algerian', 30, 'bold'))
    label.pack(pady=20)

    # Display some Vision information
    vision_info = Label(vision_window, text="The Main Vision of Our Project is to find the image is authentic or forged", font=('algerian', 16))
    vision_info.pack(pady=10)

    # Add a button to close the vision window
    close_button = Button(vision_window, text="Close", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue", command=vision_window.destroy)
    close_button.pack(pady=20)



# Function to update button appearance when mouse hovers
def on_enter(e):
    e.widget['bg'] = 'white'  # Change background color on hover

def on_leave(e):
    e.widget['bg'] = 'skyblue'  # Revert background color when mouse leaves

# GUI setup
window9 = Tk()
window9.geometry('1920x1080')

# Change the window title
window9.title("Image Forgery Detection")

# Background image
fp = open("back2.jpeg", "rb")
image = Image.open(fp)
image = image.resize((1920, 1080))
photo_image = ImageTk.PhotoImage(image)
label = Label(window9, image=photo_image)
label.place(x=0, y=0)



# Navbar setup
navbar = Frame(window9, bg="lightblue", height=50)
navbar.pack(fill="x", side="top")
```

```
# Create a frame for button container
button_frame = Frame(navbar, bg="lightblue")
button_frame.pack(side="left", padx=50)  # Adjust the padx to give space at the beginning

# Navbar buttons
btn_home = Button(navbar, text="Home", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue")
btn_home.pack(side="left", padx=15, pady=15)

#btn_train = Button(navbar, text="Train", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue")
#btn_train.pack(side="left", padx=15, pady=15)

btn_predict = Button(navbar, text="Predict", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue", command=predict)
btn_predict.pack(side="left", padx=30, pady=15)

# New "Contact Us" button
#btn_contact_us = Button(navbar, text="Contact Us", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue", command=open_contact_us)
#btn_contact_us.pack(side="left", padx=30, pady=15)

# New button added to the navbar
btn_vision = Button(navbar, text="Visions", width=15, height=2, font=('algerian', 14, 'bold'), bg="skyblue", command=open_vision)
btn_vision.pack(side="left", padx=30, pady=15)


# Load a small image for the right side of the navbar (for example, "logo.png")
right_image = Image.open("logofinal.png")  # Replace with the actual path to your image file
right_image = right_image.resize((100, 100))  # Resize the image to a small size
right_photo = ImageTk.PhotoImage(right_image)
```

```
# Add the image to the right side of the navbar as a Label
right_label = Label(navbar, image=right_photo, bg="lightblue")
right_label.pack(side="right", padx=30)  # This places the image on the right side



# Bind hover events for each button
btn_home.bind("<Enter>", on_enter)
btn_home.bind("<Leave>", on_leave)

#btn_train.bind("<Enter>", on_enter)
#btn_train.bind("<Leave>", on_leave)

btn_predict.bind("<Enter>", on_enter)
btn_predict.bind("<Leave>", on_leave)

#btn_contact_us.bind("<Enter>", on_enter)
#btn_contact_us.bind("<Leave>", on_leave)

btn_vision.bind("<Enter>", on_enter)
btn_vision.bind("<Leave>", on_leave)

# Main window labels
lb1 = Label(window9, text="Welcome to Image Forgery Detection", font=('algerian', 50, 'bold'), justify='center', fg="BLUE", bg="lightblue")
lb1.place(relx=0.3, rely=0.1, anchor='center')
lb1.place(x=300, y=150)

intro_text = """
In today’s digital age, the authenticity of images has become increasingly difficult to verify.
With the advent of powerful image manipulation tools, the risk of fake and doctored images being circulated
```

```
is higher than ever. Whether it's for news, social media, or legal purposes, ensuring that images are genuine is crucial.


Our Image Forgery Detection System uses advanced Convolutional Neural Networks (CNNs) to automatically identify altered images

and detect forgeries with high accuracy. CNNs, a powerful tool in deep learning, can analyze image features to distinguish real content

from manipulated ones, providing a reliable method to verify the authenticity of images.

"""


# Create a label with the introductory text

intro_label = Label(window9, text=intro_text, font=('Times New Roman', 20), justify='center', fg="black", bg="lightblue", padx=20, pady=20)

intro_label.place(x=10, y=350)  # Adjust position as needed


window9.mainloop()
```

3. program.py

```
n=3

result=1

for i in range(1,n+1):

    result=result*i


print(result)
```

4. test.py

```
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import seaborn as sns

np.random.seed(2)

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

from keras.models import Sequential

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

from keras.optimizers import RMSprop

from keras.preprocessing.image import ImageDataGenerator

from keras.callbacks import ReduceLROnPlateau, EarlyStopping

sns.set(style='white', context='notebook', palette='deep')

from PIL import Image

import os

from pylab import *

import re

from PIL import Image, ImageChops, ImageEnhance

from tkinter.filedialog import askopenfilename


def get_imlist(path):

    return [os.path.join(path,f) for f in os.listdir(path) if f.endswith('.jpg') or f.endswith('.png')]


def convert_to_ela_image(path, quality):

    filename = path

    resaved_filename = filename.split('.')[0] + '.resaved.jpg'

    ELA_filename = filename.split('.')[0] + '.ela.png'


    im = Image.open(filename).convert('RGB')

    im.save(resaved_filename, 'JPEG', quality=quality)
    resaved_im = Image.open(resaved_filename)

    ela_im = ImageChops.difference(im, resaved_im)

    extrema = ela_im.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff

    ela_im = ImageEnhance.Brightness(ela_im).enhance(scale)

    return ela_im

#file_name = 'authentic0.jpg'
#file_name = 'F:\\HEMANT\\Task\\Image Forgery Detection\\IF1\\examples\\Au\\authentic6'

from tkinter.filedialog import askopenfilename
file_name = askopenfilename(title='Select image file for analysis ',filetypes=[('image files', '.jpg')])

X = []
X.append(array(convert_to_ela_image(file_name, 90).resize((128, 128))).flatten() / 255.0)
X = np.array(X)
X = X.reshape(-1, 128, 128, 3)

#Prediction
from keras.models import load_model
model = load_model('model.h5')

#model.summary()
```

```
Y_pred = model.predict(X)
#print(Y_pred)
print('------------------------------Prediction Started------------------------------------')
print('\nImage is:\n')
if(Y_pred[0][0]>=Y_pred[0][1]):
   print("Authentic")
else:
   print("Forged")
```

5.train.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
np.random.seed(2)
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
sns.set(style='white', context='notebook', palette='deep')

from PIL import Image
import os
```

```
from pylab import *
import re
from PIL import Image, ImageChops, ImageEnhance




def get_imlist(path):
    return [os.path.join(path,f) for f in os.listdir(path) if f.endswith('.jpg') or f.endswith('.png')]


def convert_to_ela_image(path, quality):
    filename = path
    resaved_filename = filename.split('.')[0] + '.resaved.jpg'
    ELA_filename = filename.split('.')[0] + '.ela.png'

    im = Image.open(filename).convert('RGB')
    im.save(resaved_filename, 'JPEG', quality=quality)
    resaved_im = Image.open(resaved_filename)

    ela_im = ImageChops.difference(im, resaved_im)

    extrema = ela_im.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff

    ela_im = ImageEnhance.Brightness(ela_im).enhance(scale)

    return ela_im
```

```
#Read dataset
dataset = pd.read_csv('dataset.csv')

X = []
Y = []

for index, row in dataset.iterrows():
   X.append(array(convert_to_ela_image(row[0], 90).resize((128, 128))).flatten() / 255.0)
   Y.append(row[1])

X = np.array(X)
Y = to_categorical(Y, 2)
print(Y)
X = X.reshape(-1, 128, 128, 3)

X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2, random_state=5)

model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                activation ='relu', input_shape = (128,128,3)))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                activation ='relu'))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(2, activation = "softmax"))

model.summary()

optimizer = RMSprop(lr=0.0005, rho=0.9, epsilon=1e-08, decay=0.0)

model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])

early_stopping = EarlyStopping(monitor='val_acc',
                 min_delta=0,
                 patience=2,
                 verbose=0, mode='auto')

#Epochs
epochs = 30
batch_size = 100

history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
      validation_data = (X_val, Y_val), verbose = 2, callbacks=[early_stopping])

#Model save
model.save('model.h5')
```

```
print('---------------Saved Model---------------')
```