Programming Project 2 Real and Fake Detection

Jasmine Roberts

Bowie State University

CTEC-445-101

Dr.Haydar

September 17, 2024

**Programming Project 2 Real and Fake Detection**

Programming Project 2 consisted of programming and plotting the accuracy and loss results. We downloaded a dataset from [Real and Fake Detection | Kaggle.](https://www.kaggle.com/datasets/ciplab/real-and-fake-face-detection) We used the images to train and test the model. They were resized and saved to a folder in the C drive of our computer. Below are the numerous steps we took to construct many lines of code to detect real and fake.

**Lines of code**

First step is to downloading the images from Kraggle and save them to a folder in your C driveA screenshot of a computer

Description automatically generatedMove these two files into your C drive ( in the header “Windows (C:) ) drag your files to there.

# Downloading Algorithms

Second step is downloading the recommended algorithms for the project:

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

The algorithms we needed were:

1. Native Bayes
2. K-Nearest Neighbor
3. Decision Tree
4. Logistic Regression

**Python Program to resize all of the images**

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

import cv2

import os

import numpy as np

from sklearn.model\_selection import train\_test\_split

*Import cv2* is a function to import OpenCV libraries for the detection and processing

*Import os* is importing the program and/or module

*Import numpy as np* is for number inside of the python program

*from sklearn.model\_selection import train\_test\_spli*t is for testing the data inside of the program

# Define paths to the dataset folders

training\_real\_folder = r'C:\training\_real'

training\_fake\_folder = r'C:\training\_fake'

def load\_images\_from\_folder(folder, label):

images = []

labels = []

for filename in os.listdir(folder):

img\_path = os.path.join(folder, filename)

if os.path.isfile(img\_path):  #check the file name

*training\_real\_folder* = this file name inside of my C drive

*r’C:\training\_real’* is the C file path of the folder inside of my C drive

*training\_fake\_folder* = this file name inside of my C drive

*r’C:\training\_fake’* is the C file path inside of my C drive

*Def load images\_from\_ folder(folder, label):* is the load images function to load the images from selected folder

*Images[] and labels[]* is the selected image and labels of the folder

*For filename in os.listdr(folder):*

*Img\_path = os.path.join(folder, filename)*

*If os.path.isfile(imag\_path):* is the function to check the file name of each folder

img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)

img = cv2.resize(img, (64, 64))  #the photo resize

if img is not None:

images.append(img.flatten())

labels.append(label)

return images, labels

*Img = cv2.imread(img\_path, cv2.IMREAD\_GRADSCALE)*

*Img = cv2.resize(img, (64, 64)*)

*If image is not None:*

*images .append(img.flatten())*

*labels.append(label)*

*Return images, labels*

This is the function to resize the images

#Loading the images

training\_real, real\_labels = load\_images\_from\_folder(training\_real\_folder, 1)

training\_fake, fake\_labels = load\_images\_from\_folder(training\_fake\_folder, 0)

This is the function to load the images from each folder (real and fake)

#Combing both images together

X = np.array(training\_real + training\_fake)

y = np.array(real\_labels + fake\_labels)

#Split into training and testing sets

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

The first function is to combine both images together, and the second function is to split both into training and testing sets.

**Python Program for Real and Fake Detection**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a graph

Description automatically generated

*from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score, log\_loss  
import matplotlib.pyplot as plt*

These are all of the required algorithms in order to start the program, these are the functions of importing them

*#The required algorithms  
models = {  
    "1.Naive Bayes": GaussianNB(),  
    "2. K-Nearest Neighbor": KNeighborsClassifier(),  
    "3. Decision Tree": DecisionTreeClassifier(),  
    "4. Logistic Regression": LogisticRegression(max\_iter=3000)  
}  
#Plotting the accuracy scores and the losses  
accuracy\_scores = []  
losses = []*

These are the functions to import the required algorithms, and also the function to plot the accuracy and the losses  
  
*for name, model in models.items():  
    model.fit(X\_train, y\_train)  
    y\_pred = model.predict(X\_test)  
  
    accuracy = accuracy\_score(y\_test, y\_pred)  
    accuracy\_scores.append(accuracy)  
  
    if hasattr(model, 'predict\_proba'): #predict proba is the columns and arrays  
        loss = log\_loss(y\_test, model.predict\_proba(X\_test))  
        losses.append(loss)  
    else:  
        losses.append(None)  
  
#Plots for each result  
fig, ax1 = plt.subplots()*

These functions is to plot the accuracy scores and the losses, and predict proba is for the columns and the arrays with the detection. And to plot each result.

*#Blue is for the bars on the grpah and titles of the bars  
color = 'tab:blue'  
ax1.set\_xlabel('Model')  
ax1.set\_ylabel('Accuracy', color=color)  
ax1.bar(models.keys(), accuracy\_scores, color=color, alpha=0.6, label="Accuracy")  
ax1.tick\_params(axis='y', labelcolor=color)*

This function is for the bars on the graph, and titles of the bars on the graph (blue)

*#Green is for the line on the graph  
ax2 = ax1.twinx()  
color = 'tab:green'  
ax2.set\_ylabel('Log Loss', color=color)  
ax2.plot(models.keys(), losses, color=color, alpha=0.6, marker='o', label="Log Loss")  
ax2.tick\_params(axis='y', labelcolor=color)  
  
#The title of the figure  
fig.tight\_layout()  
plt.title( ' The Model Accuracy and Loss Chart')  
plt.show()*

 These functions is for the line on the graph which is green, and the last function is for the title of the figure