|  |  |
| --- | --- |
| Team ID | PNT2022TMID52735 |
| Project Name | A Novel Method for Handwritten Digit  Recognition System |

**Bulid python PART-1**

**MODEL CREATION:**

from keras.datasets import mnist

import matplotlib.pyplot as plt

from keras.utils import np\_utils

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.optimizers import Adam

(X\_train,y\_train),(

X\_test,y\_test) =mnist.load\_data()

print(X\_train.shape)

print(X\_test.shape)

print(y\_test.shape)

print(y\_train.shape)

print("The label value is ",y\_test[10]) #Value in y\_test

plt.imshow(X\_test[10])

print("The label value is ",y\_test[65]) #Value in y\_test

plt.imshow(X\_test[65])

X\_train.shape

X\_test.shape

                 X\_train1 = X\_train.reshape(60000, 28, 28, 1).astype('float32')

                X\_test1 = X\_test.reshape(10000, 28, 28, 1).astype('float32')

                number\_of\_classes= 10

y\_train1 = np\_utils.to\_categorical(y\_train,number\_of\_classes)

y\_test1 = np\_utils.to\_categorical(y\_test,number\_of\_classes)

                print("After encoding the value",y\_test[10] ,"become", y\_test1[10])

                print("After encoding the value",y\_test[100] ,"become", y\_test1[100])

                 print("After encoding the value",y\_test[65] ,"become", y\_test1[65])

                model = Sequential()

model.add(Conv2D(64, (3, 3), input\_shape=(28, 28, 1), activation="relu"))

model.add(Conv2D(32, (3, 3), activation="relu"))

model.add(Flatten())

model.add(Dense(number\_of\_classes, activation="softmax"))

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

                 model.fit(X\_train1, y\_train1, batch\_size=32, epochs=5, validation\_data=(X\_test1,y\_test1))

                metrics = model.evaluate(X\_test1, y\_test1, verbose=0)

print("Metrics (Test Loss & Test Accuracy): ")

print(metrics)

prediction = model.predict(X\_test1[:4])

print(prediction)

                import numpy as np

print(np.argmax(prediction, axis=1))

print(y\_test1[:4])

                model.save("model.h5")

from tensorflow.keras.models import load\_model

model=load\_model("model.h5")

                 model.summary()

**FLASK APP:**

from flask import Flask, render\_template, request,redirect,session, url\_for

from flask\_mail import Mail, Message

from itsdangerous import URLSafeTimedSerializer, SignatureExpired

import mysql.connector

import os

from flask\_mysqldb import MySQL

from recognize import recognize

import requests

from io import BytesIO

from werkzeug.utils import secure\_filename

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

app.secret\_key=os.urandom(24)

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = ''

app.config['MYSQL\_DB'] = 'digit\_recognition'

       mysql = MySQL(app)

       @app.route('/')

def index():

    return render\_template('index.html')

       @app.route('/login')

def login():

    return render\_template('login.html')

@app.route('/register/')

def about():

    return render\_template('form.html')

       @app.route('/home')

def home():

    if 'email' in session:

        return render\_template('home.html')

    else:

        return redirect('/')

           @app.route('/login\_validation',methods=['POST'])

def login\_validation():

    if request.method == "POST":

                       email=request.form.get('email')

        password=request.form.get('password')

        error = None

        if mysql:

            print("Connection Successful!")

            cursor = mysql.connection.cursor()

            cursor.execute("""SELECT \* FROM `users` where `Email` LIKE '{}' """.format(email))

            users = cursor.fetchall()

            cursor.close()

            cursor1 = mysql.connection.cursor()

            cursor1.execute("""SELECT \* FROM `users` where `Email` LIKE '{}' and `Password` LIKE '{}'""".format(email, password))

            users1 = cursor1.fetchall()

            cursor1.close()

                       else:

            print("Connection Failed!")

        if len(users)>0:

            if len(users1)>0:

                session['email'] = users[0][1]

                return redirect('/home')

            else:

                error = "Wrong password"

        else:

            error = "Email not available"

    return render\_template('login.html',error=error)

@app.route('/add\_user',methods=['POST'])

def add\_user():

    username=request.form.get('username')

    email = request.form.get('email')

    password = request.form.get('password')

    phone = request.form.get('phone')

    gender = request.form.get('gender')

    if mysql:

        print("Connection Successful!")

        cursor = mysql.connection.cursor()

        cursor.execute(

            """INSERT INTO `users` (`FullName`,`Email`,`Password`,`PhoneNo`,`Gender`) VALUES ('{}','{}','{}','{}','{}')""".format(username,email, password,phone,gender))

        mysql.connection.commit()

        cursor.close()

    else:

        print("Connection Failed!")

    return redirect('/login')

@app.route('/logout')

def logout():

    return redirect('/')

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

def predictpage():

    return render\_template('prediction.html')

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

def submit():

    if request.method == 'POST':

        # Upload file flask

        uploaded\_img = request.files['image']

        # Upload file to database (defined uploaded folder in static path)

        uploaded\_img.save('./static/data/1.jpg')

        # Storing uploaded file path in flask session

        session['uploaded\_img\_file\_path'] = "./static/data/1.jpg"

        return render\_template('prediction.html')

@app.route('/prediction',methods=('POST', "GET"))

def predict():

        # Retrieving uploaded file path from session

        img\_file\_path = session.get('uploaded\_img\_file\_path', None)

        best, img1 = recognize(img\_file\_path)

        return render\_template("prediction.html", best=best, img\_name=img1)

if \_\_name\_\_=="\_\_main\_\_":

    app.run(debug=True)