| | |
|---|---|
| Team ID | PNT2022TMID26315 |
| 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)
```