Task 1: Creating a neural network

For this task I Choose LeNet-5 Architecture. Because

- Simplicity: LeNet-5 is relatively simple and easiest one, making it a good starting point for understanding CNN architectures.
- Small Datasets: It works well with smaller datasets like MNIST, which contain low-resolution images (28x28).

Dataset Preprocessing

The MNIST dataset was loaded using the Keras library. The dataset was split into training and validation sets. Each set contains images of handwritten digits and their corresponding labels.

The images were preprocessed:

Reshaping: The images were reshaped to a standardized format of 28x28 pixels, suitable for the LeNet-5 architecture.

Normalization: Pixel values were scaled between 0 and 1 by dividing by 255, which aids in faster convergence during training.

LeNet-5 Architecture Configuration

The LeNet-5 architecture was implemented using Keras Sequential API:

Layer 1:

- Convolutional Layer 1: Applied 6 filters of size 5x5 with a ReLU activation function to capture low-level features.
- Max Pooling Layer 1: Performed 2x2 max pooling to downsample the feature maps.

Layer 2:

- Convolutional Layer 2: Utilized 16 filters of size 5x5 with a ReLU activation function to extract deeper features.
- Max Pooling Layer 2: 2x2 max pooling was applied to reduce spatial dimensions.
- Flatten Layer: Flattened the output from the previous layers to prepare for fully connected layers.

Layer 3:

Fully Connected Layer 1: Comprised of 120 neurons with a ReLU activation function to learn higher-level features.

Layer 4:

Fully Connected Layer 2: Included 84 neurons with a ReLU activation function. Dropout with a rate of 0.2 was used to prevent overfitting.

Layer 5(Output):

Output Layer with 10 neurons and a softmax activation function for 10-class classification.

3. Model Compilation and Training

The model was compiled with:

- **Optimizer**: 'adam' was chosen for its adaptive learning rate and efficiency in training.
- **Loss Function:** 'categorical_crossentropy' was used as it's well-suited for multi-class classification.
- **Metrics**: 'accuracy' was selected to monitor model performance during training.
- The model was trained for 20 epochs with a batch size of 128. Training and validation data were provided for assessing model performance and generalization. The 'verbose' parameter was set to 1 for real-time training updates.

4. Evaluation and Performance

After training, the model's performance was evaluated:

- Accuracy: The accuracy achieved on the validation set was calculated.
- Loss: The categorical cross-entropy loss value was determined

*****Last one Question you may ask . Why i use activation function relu on the hidden layer and softmax on the last layer.*****

- Softmax activation function is typically used in the output layer for multi-class classification tasks and Sigmoid for binary classification task.
- We use relu as a activation function in the hidden layer because of avoiding expoid or vanishing gradient problem.

In this field, It's all about experiment. 🙂

Task 2: Working with databases

Database Structure:

Table: Students

Columns:

- id (INTEGER AUTO_INCREMENT PRIMARY KEY): Unique identifier for each student. The AUTO_INCREMENT attribute ensures that each new entry automatically gets a unique ID.
- name (VARCHAR(255)): Holds the name of the student.
- age (INTEGER): Stores the age of the student.

Database Organization:

1. Table Structure:

- **Name and Age**: The table captures basic information about students, their names, and ages. These are common and essential details to have about individuals.
- **ID as Primary Key:** An auto-incremented ID serves as a primary key. This guarantees a unique identifier for each record and is useful for referencing specific individuals.
- **Data Types:** The use of appropriate data types (VARCHAR for names, INTEGER for age and ID) ensures efficient storage and retrieval.

2. Functions for CRUD Operations:

“I just follow the functional programing way”

Create Table Function: The create_table() function checks for the existence of the table and creates it if it doesn't exist. It ensures that the database structure is set up for storing student data.

Insert Data Function: The add_students() function inserts new student records, capturing their name and age.

Retrieve Data Function: The retrive_students() function retrieves all student records from the table, returning the results for further processing or display.

Update Data Function: The update_students() function allows for modifications to existing student records based on their ID.

Delete Data Function: The delete_students() function removes student records based on their ID.

Why This Structure:

Simplicity: The structure is straightforward, focusing on the essential details required for a student management system.
Efficient Data Handling: Using appropriate data types ensures efficient storage and retrieval, preventing data wastage or inefficiencies.
Normalization: The structure follows basic normalization principles, ensuring minimal data redundancy by separating data into distinct tables and using a primary key.
CRUD Operations: The provided functions allow for all fundamental operations (Create, Read, Update, Delete), enabling complete data management.

Task 3: Integration with a Google API

Introduction:

This Python script is designed to interact with the Google Sheets API to retrieve data from a specified range within a Google Spreadsheet. It employs Python, utilizing various Google API libraries, to authenticate, request data, and process the received information.

Prerequisites:

Python: Ensure Python is installed (Python 3.x).
Required Libraries: Install the necessary libraries using `pip install`:

google-auth
google-auth-oauthlib
google-auth-httplib2
google-api-python-client
Usage Instructions:

Set Up Google API Credentials:

Obtain a Service Account credentials JSON file from the Google Cloud Console. Rename the file to credentials.json and place it in the same directory as the script. Ensure the Service Account has access to the Google Sheet specified by the SPREADSHEET_ID.

Workflow:

Authentication:

The script checks for existing credentials in a file named token.json. If not present or if invalid, it initiates the authentication process:
Loads client secrets from credentials.json and runs the Installed App Flow to obtain and save valid credentials to token.json.

API Interaction:

It establishes a connection to the Google Sheets API using obtained credentials.
Retrieves data from the specified range (Sheet4!C1:J9) in the specified spreadsheet.

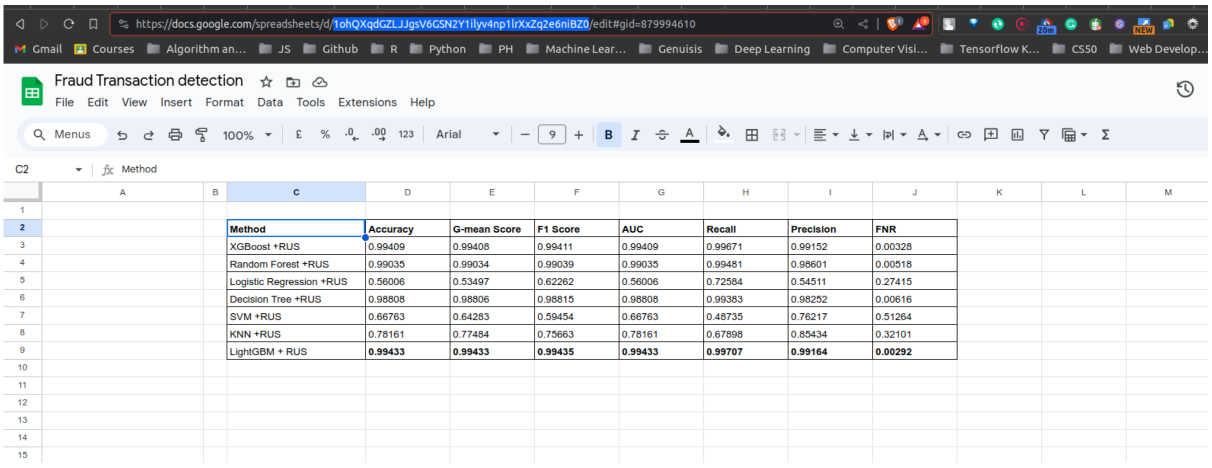
Error Handling:

In case of any errors during the API interaction, it catches and handles HttpError exceptions.

Sample Output:

The script fetches data from the specified range and prints the retrieved values to the console.

Spreadsheet and SPREADSHEET_ID



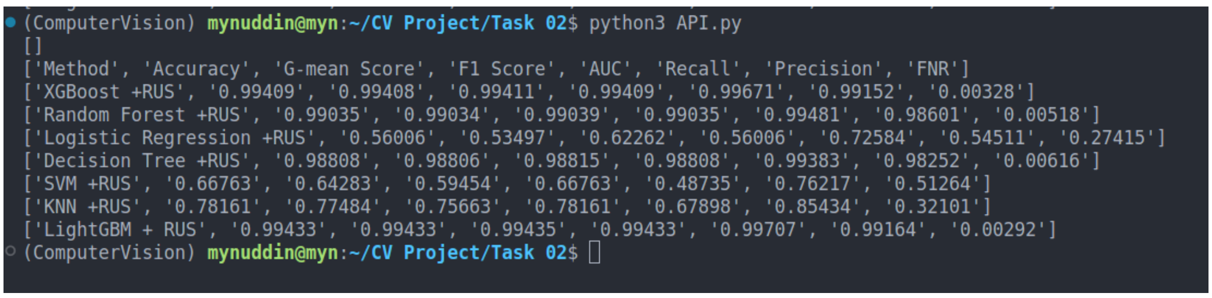
https://docs.google.com/spreadsheets/d/1ohQXqdGZLJJgsV6GSN2Y1ilyv4np1lrXxZq2e6niBZ0/edit#gid=879994610

Fraud Transaction detection

C2 Method

Method	Accuracy	G-mean Score	F1 Score	AUC	Recall	Precision	FNR
XGBoost +RUS	0.99409	0.99408	0.99411	0.99409	0.99671	0.99152	0.00328
Random Forest +RUS	0.99035	0.99034	0.99039	0.99035	0.99481	0.98601	0.00518
Logistic Regression +RUS	0.56006	0.53497	0.62262	0.56006	0.72584	0.54511	0.27415
Decision Tree +RUS	0.98808	0.98806	0.98815	0.98808	0.99383	0.98252	0.00616
SVM +RUS	0.66763	0.64283	0.59454	0.66763	0.48735	0.76217	0.51264
KNN +RUS	0.78161	0.77484	0.75663	0.78161	0.67898	0.85434	0.32101
LightGBM + RUS	**0.99433**	**0.99433**	**0.99435**	**0.99433**	**0.99707**	**0.99164**	**0.00292**

output:



```
(ComputerVision) mynuddin@myn:~/CV Project/Task 02$ python3 API.py
[]
['Method', 'Accuracy', 'G-mean Score', 'F1 Score', 'AUC', 'Recall', 'Precision', 'FNR']
['XGBoost +RUS', '0.99409', '0.99408', '0.99411', '0.99409', '0.99671', '0.99152', '0.00328']
['Random Forest +RUS', '0.99035', '0.99034', '0.99039', '0.99035', '0.99481', '0.98601', '0.00518']
['Logistic Regression +RUS', '0.56006', '0.53497', '0.62262', '0.56006', '0.72584', '0.54511', '0.27415']
['Decision Tree +RUS', '0.98808', '0.98806', '0.98815', '0.98808', '0.99383', '0.98252', '0.00616']
['SVM +RUS', '0.66763', '0.64283', '0.59454', '0.66763', '0.48735', '0.76217', '0.51264']
['KNN +RUS', '0.78161', '0.77484', '0.75663', '0.78161', '0.67898', '0.85434', '0.32101']
['LightGBM + RUS', '0.99433', '0.99433', '0.99435', '0.99433', '0.99707', '0.99164', '0.00292']
(ComputerVision) mynuddin@myn:~/CV Project/Task 02$
```

Credits : Google Workspace [Google Sheet For Python]