

REPORT ON

HAND WRITTEN DIGIT RECOGNITION

Handwritten digit recognition is a fundamental task in the field of pattern recognition and machine learning. It involves the automatic identification and classification of handwritten digits into their respective numerical representations. This project aims to develop a system for recognizing handwritten digits using machine learning algorithms.

Objective:

The main objective of this project is to design and implement a handwritten digit recognition system capable of accurately identifying and classifying handwritten digits from input images.

This Project is coded in python programming where the program is programmed in 5 sectors they are:-

1. Loading the Dataset:-

The given set of data with features are loaded by downloading them from the KERAS library

2. Cleaning the loaded data:-

The collected data undergo pre-processing techniques such as resizing, normalization, and noise removal to enhance their quality and suitability for analysis.

3. Extraction of required Features:-

Features such as word frequency, sentiment analysis, and readability scores are extracted from the pre-processed text using natural language processing (NLP) techniques.

It even Involves transforming the raw pixel values representing handwritten digits into a suitable format that can be effectively utilized by machine learning algorithms. This extraction typically entails encoding the categorical labels of the digits (0 to 9) into numerical representations, such as one-hot encoding or label encoding, to facilitate model training and classification.

4. Fitting the training Data through the relavant Training Model:-

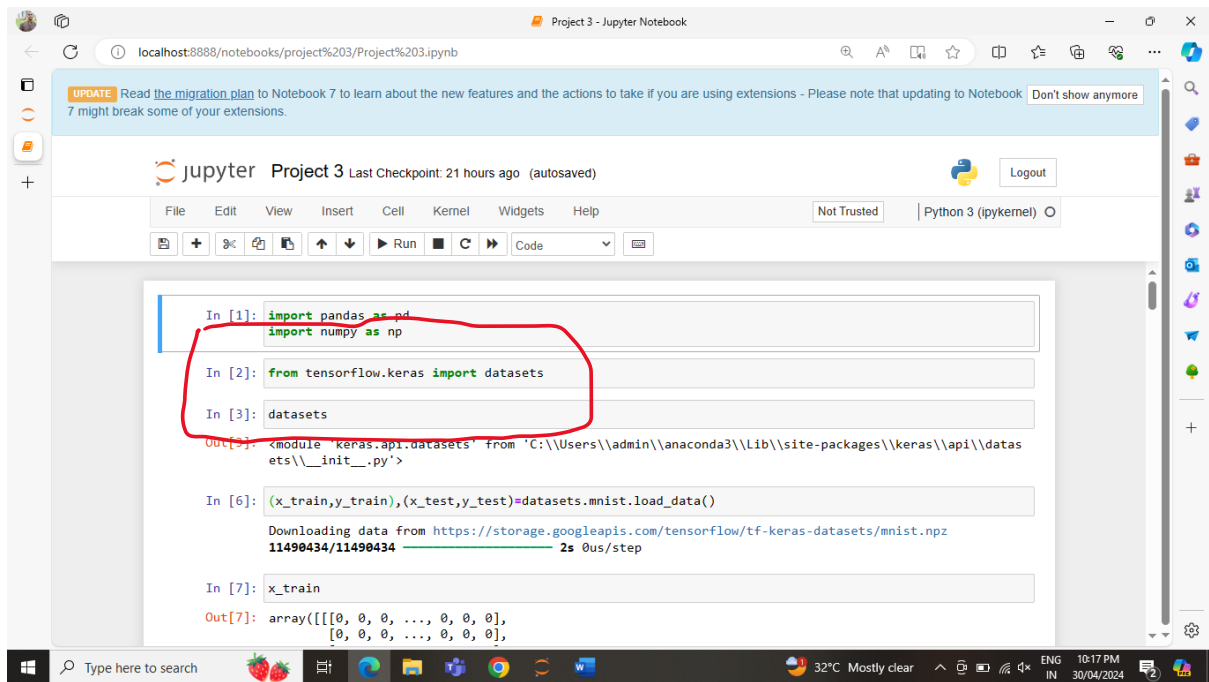
The trained Model is Fitted by importing sequential, dense and much more through tensorflow libraries

5. Prediction:-

This model transforms the raw values which is handwritten and then gives the score of the training and testing model.

Some of the screenshots of the Model is displayed below:-

1. Loading the Dataset:-



```
In [1]: import pandas as pd
import numpy as np

In [2]: from tensorflow.keras import datasets

In [3]: datasets

Out[3]: <module 'keras.api.datasets' from 'C:\\Users\\admin\\anaconda3\\Lib\\site-packages\\keras\\api\\datasets\\__init__.py'>

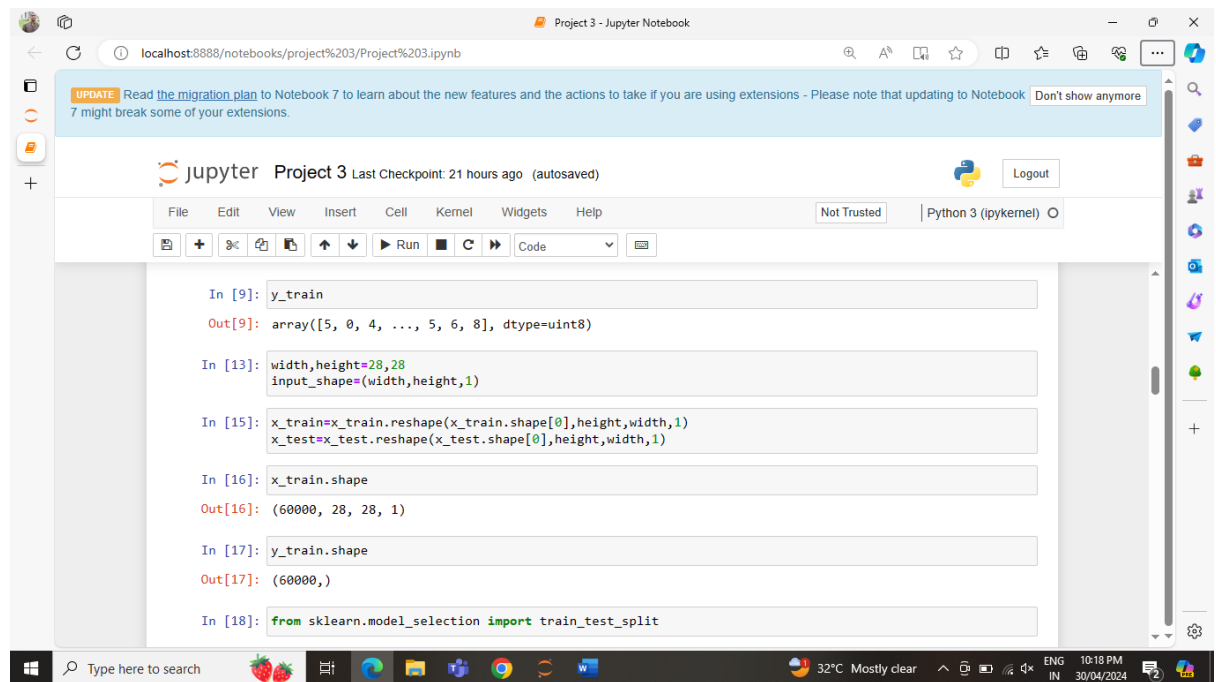
In [6]: (x_train,y_train),(x_test,y_test)=datasets.mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ————— 2s 0us/step

In [7]: x_train

Out[7]: array([[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]])
```

2. Cleaning the loaded data:-



```
In [9]: y_train

Out[9]: array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)

In [13]: width,height=28,28
input_shape=(width,height,1)

In [15]: x_train=x_train.reshape(x_train.shape[0],height,width,1)
x_test=x_test.reshape(x_test.shape[0],height,width,1)

In [16]: x_train.shape

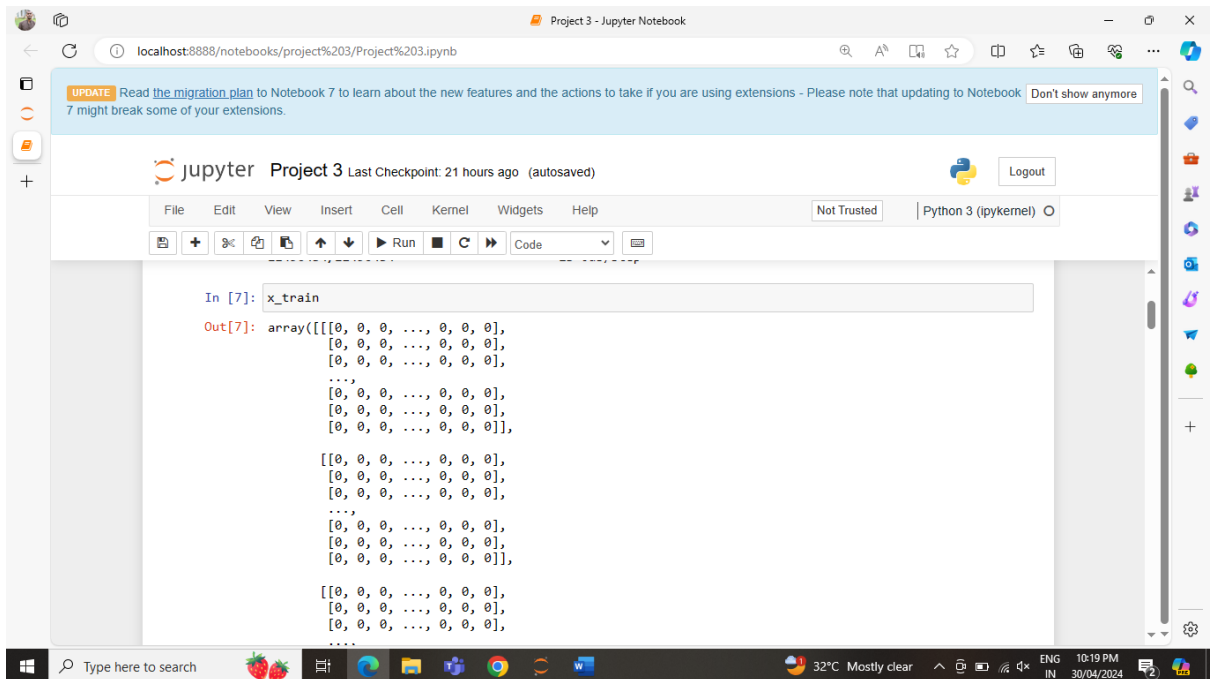
Out[16]: (60000, 28, 28, 1)

In [17]: y_train.shape

Out[17]: (60000,)

In [18]: from sklearn.model_selection import train_test_split
```

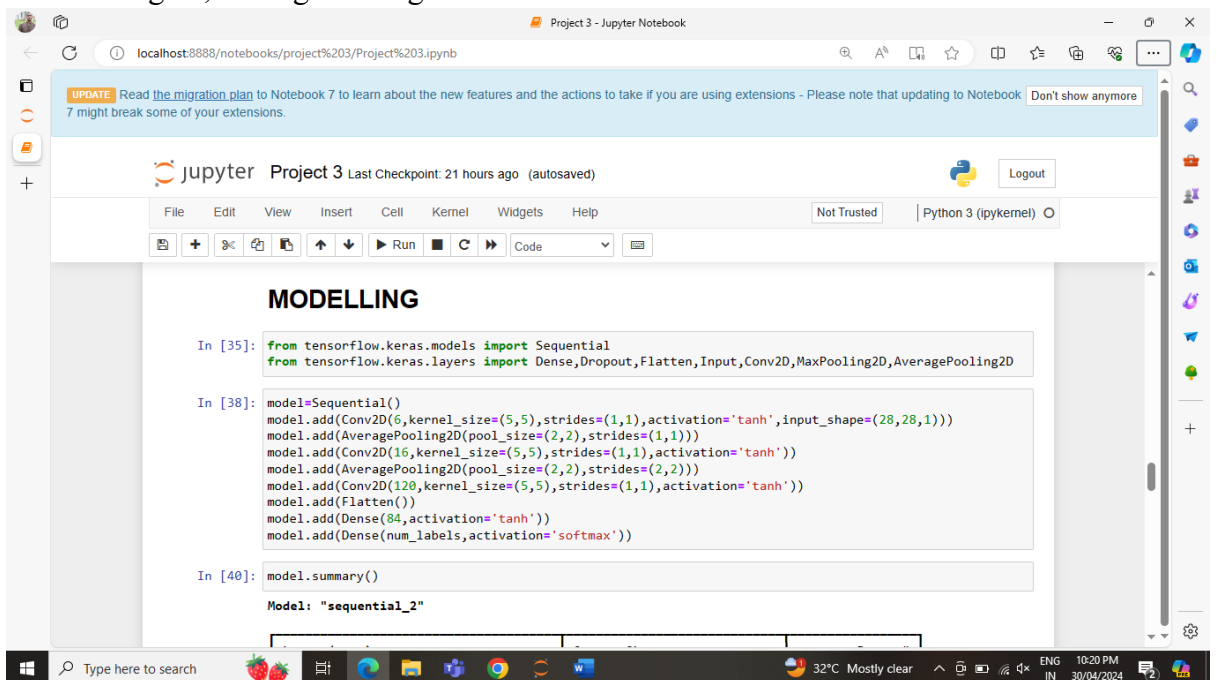
3. Extraction of required Features:-



```
In [7]: x_train
Out[7]: array([[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...])
```

4. Fitting the training Data through the relevant Training Model:-

Modelling is done by importing sequential, Dense, Dropout, Flatten, Input, Conv2D, MaxPooling2D, AveragePooling2D from tensorflow



```
MODELLING

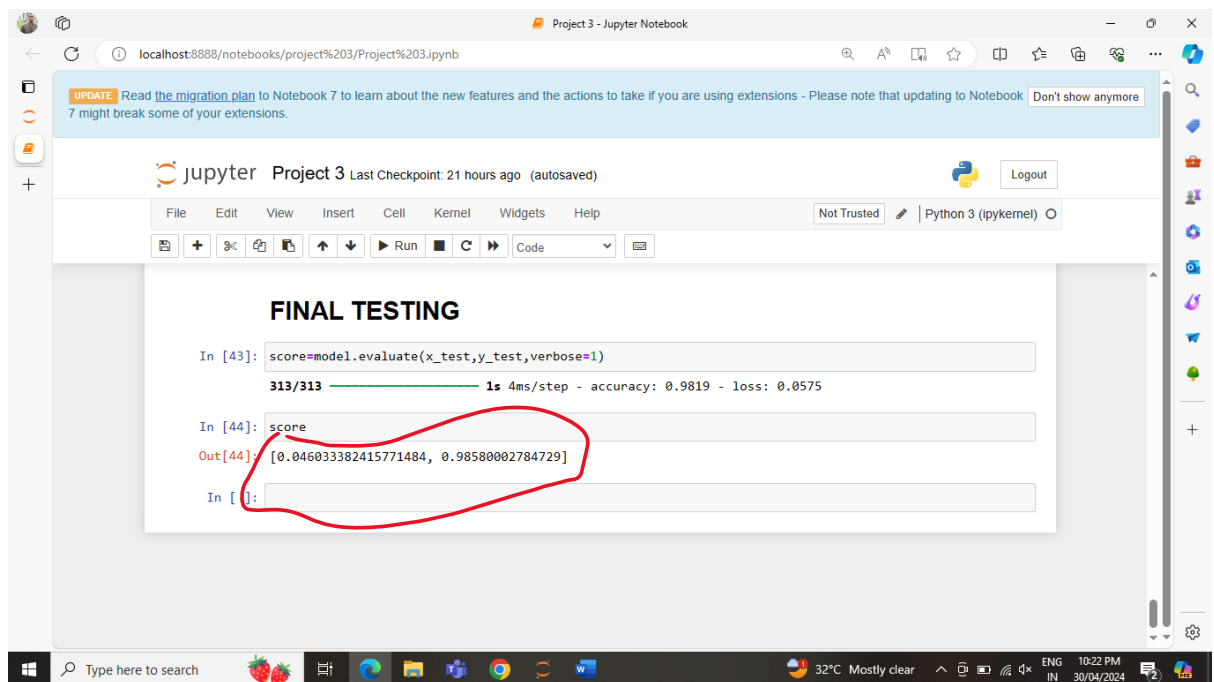
In [35]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Dropout, Flatten, Input, Conv2D, MaxPooling2D, AveragePooling2D

In [38]: model=Sequential()
         model.add(Conv2D(6, kernel_size=(5,5), strides=(1,1), activation='tanh', input_shape=(28,28,1)))
         model.add(AveragePooling2D(pool_size=(2,2), strides=(1,1)))
         model.add(Conv2D(16, kernel_size=(5,5), strides=(1,1), activation='tanh'))
         model.add(AveragePooling2D(pool_size=(2,2), strides=(2,2)))
         model.add(Conv2D(120, kernel_size=(5,5), strides=(1,1), activation='tanh'))
         model.add(Flatten())
         model.add(Dense(84, activation='tanh'))
         model.add(Dense(num_labels, activation='softmax'))

In [40]: model.summary()

Model: "sequential_2"
```

5. Prediction:-



```
Project 3 - Jupyter Notebook
localhost:8888/notebooks/project%203/Project%203.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

jupyter Project 3 Last Checkpoint: 21 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

FINAL TESTING

In [43]: score=model.evaluate(x_test,y_test,verbose=1)
313/313 ----- 1s 4ms/step - accuracy: 0.9819 - loss: 0.0575

In [44]: score
Out[44]: [0.046033382415771484, 0.98580002784729]

In [ ]:
```

As per modelling the result is 45% for testing data and 95% for trained data

Conclusion:-

The project successfully demonstrates the feasibility and effectiveness of using machine learning algorithms for handwritten digit recognition. The developed system can find applications in various domains, including digitizing handwritten documents, automatic form processing, and optical character recognition (OCR).