MINI PROJECT REPORT

On

DIGIT RECOGNITION

B.E (IT) – III Sem

By

Marka Gangadhar (160121737185)

Erugadindla Abhishek (160121737172)

Under the guidance of

Mr. B. Harish Goud
Mr. K. Gangadhar Rao
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)
KOKAPET(V), GANDIPET(M), RR District HYDERABAD -75
Website: www.cbit.ac.in

2022-2023

CONTENTS

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
Kokapet (Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in

ISO Certified 9001:2015

COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 43 years

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY
స్వయం తేజస్విన్ భవ
1979

CERTIFICATE

This is to certify that the project work entitled "**DIGIT RECOGNITION**" submitted to CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY, in partial fulfillment of the requirements for the award of the completion of III semester of B.E in Information Technology, during the academic year 2022-2023, is a record of original work done by **MARKA GANGADHAR(160121737185)** , **ERUGADINDLA ABHISHEK(160121737172)** during the period of study in Department of IT, CBIT, HYDERABAD, under our guidance.

Project Guide	**Head of the Department**
Mr. B. Harish Goud	**Dr. K. Radhika**
Assistant Professor, Dept. of IT,	Professor, Dept. of IT,
CBIT, Hyderabad.	CBIT, Hyderabad.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Mr. B. Harish Goud**, **Mr. K. Gangadhar Rao** our project guide, for her valuable guidance and constant support, along with her capable instruction and persistent encouragement.

We are grateful to our Head of Department, **Dr. K. Radhika**, for her steady support and for the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, **Dr. P. Ravinder Reddy**, as well as the management of the institute, for having designed an excellent learning atmosphere.

Our thanks to all members of the staff and our lab assistants for providing me with the help required to carry out the groundwork of this project.

ABSTRACT

This project report presents an implementation of digit recognition using Python. The objective of this project is to develop a model that can accurately recognize digits from images. The project involved the use of the MNIST dataset, which contains a large number of labeled images of digits. The implementation used the popular Python libraries pandas, scikit learn, matplotlib, numpy to build and train a k-NN k-nearest neighbor model. The model was trained on the MNIST dataset and was able to achieve best accuracy. The report describes the process of data preprocessing, model building, and model training. It also includes a discussion of the results and a comparison with other approaches. The project demonstrates the potential of supervised machine learning techniques for image recognition tasks and highlights the importance of data preprocessing and model optimization for achieving high accuracy.

LIST OF FIGURES

1. INTRODUCTION

1.1 Motivation

The motivation for the project report on digit recognition using Python is to explore the practical applications of machine learning and computer vision in solving real-world problems. Digit recognition is a common and important task in many fields, such as finance, healthcare, and education. For instance, automated recognition of handwritten digits can be useful in processing checks and other financial documents, in digitizing handwritten notes and historical documents, and in recognizing medical imaging data. Moreover, digit recognition is a fundamental problem in computer vision and deep learning, and has been a benchmark problem in the field for several years. The availability of large datasets such as MNIST has made it possible to develop and test sophisticated models for digit recognition. This project aims to contribute to the existing research in this field by building and evaluating a high-performance model for digit recognition using Python, and to provide insights into the practical implementation of deep learning algorithms for image recognition tasks.

1.2 Problem statement

Designing a digit recognition system using Python is a challenging task as it involves a lot of complex computations and decision-making processes. The purpose of this mini project is to create an efficient digit recognition system that can accurately identify handwritten digits. The system should be able to recognize digits from 0 to 9. The challenge lies in choosing the right algorithms and techniques for preprocessing, feature extraction, and machine learning to achieve the desired accuracy. The system should be able to handle a wide range of variations in handwritten digits and should be able to recognize them with a high degree of accuracy. The outcome of this mini project will be an efficient digit recognition system that can accurately identify handwritten digits, which can be used in a variety of applications.

1.3 Organization of report

The work presented in this report is organized into seven chapters. After this introductory chapter (**Chapter 1**),

- **Chapter 2** gives a detailed literature survey of this project. It discusses the details and working of the existing work based on the project topic digit recognition .
- **Chapter 3** specifies the system requirements needed to implement this project.
- **Chapter 4** describes the current project i.e., the proposed system-the features, and operations included in it.
- **Chapter 5** contains the screenshots of code snippets that are made for the implementation of the project.
- **Chapter 6** contains the screenshots of the execution and output of the project.
- **Chapter 7** gives the conclusion of the project and gives some ideas on how to improvise and develop the project further.

2. LITERATURE SURVEY

S. No	Research paper	Conference or journal	Published year	Authors	Outcome
1	Gaurav, Bhatia P. K. [5] Et al (Journal)	Journal	May 2014	Gaurav, Bhatia	This paper deals with the various pre-processing techniques involved in the character recognition with different kind of images ranges from a simple handwritten form based documents and documents containing colored and complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques
2	Sakarya University Journal of Science (Journal)	Journal	September 2020	Rabia KARAKAYA, Serap KAZAN	Presented for handwritten character recognition. Handwritten character was transformed into graphs based on its underlying skeleton structure.

What is Handwritten Digit Recognition?

The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavours. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

The MNIST dataset

This is probably one of the most popular datasets among machine learning and deep learning enthusiasts. The MNIST dataset contains 60,000 training images of handwritten digits from zero to nine and 10,000 images for testing. So, the MNIST dataset has 10 different classes. The handwritten digits images are represented as a 28×28 matrix where each cell contains grayscale pixel value.

K-Nearest Neighbor (KNN):

KNN is a simple and effective machine learning algorithm for digit recognition. It works by comparing the features of an image with a set of pre-classified images. The KNN algorithm can be trained on a large dataset of images and used to classify digits accurately.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

Processor	:	11th Gen Intel® Core (Tm) i5-1155G7 @ 2.50GHz
Memory	:	16GB RAM

3.2 SOFTWARE REQUIREMENTS

Operating system	:	Windows 11 pro 64 bit
Programming language	:	Python
Web browser	:	Chrome (58 and above), Firefox (54 and above), Safari (10.1 and above), Opera (44 and above)
Code editor	:	Google Colaboratory
Packages	:	Pandas, Numpy, Matplotlib ,Sci-kit Learn

4. METHODOLOGY

Data Collection and Pre-processing: Gather a dataset of handwritten digits, such as the MNIST dataset, which is a standard benchmark for this type of project. Pre-process the data by normalizing the pixel values to a common scale, converting the images to grayscale if necessary, and splitting the dataset into training and testing sets.

Feature Extraction: Extract features from the images to use for classification. One common approach is to use pixel intensity values as features. Other features such as texture, edge detection or shape descriptors could be used for more advanced methods.

Implementation of k-NN Algorithm: Implement the k-NN algorithm using a programming language of your choice. The k-NN algorithm involves finding the k closest training examples to a given test example and classifying the test example based on the majority class among those k neighbors.

Model Training and Selection: Train the k-NN model on the training set and evaluate its performance on the testing set. Vary the value of k and evaluate its effect on the performance of the model. Use cross-validation to estimate the generalization performance of the model.

Model Evaluation and Visualization: Evaluate the performance of the k-NN model using evaluation metrics such as accuracy, precision, recall and F1-score. Visualize the performance of the model using confusion matrices, ROC curves, or precision-recall curves.

Comparison with other algorithms: Compare the performance of the k-NN model with other machine learning algorithms such as decision trees, SVM or neural networks. Analyze the strengths and weaknesses of each algorithm and provide insights on which algorithm might be better suited for the task.

Fine-tuning: Finally, fine-tune the parameters of the k-NN model to achieve the best possible performance. Experiment with different feature extraction techniques, distance metrics and normalization methods to see their effects on the model's performance.

Overall, a successful project will involve proper data collection, appropriate feature selection, and effective evaluation of the model's performance.

4.1 System design

System architecture diagram

- Read an image of a handwritten digit
- Extract features from the image
- Train the system using a machine learning algorithm
- Recognize the digit by classifying the image into one of the 10 classes (0 to 9)
- Display the result.

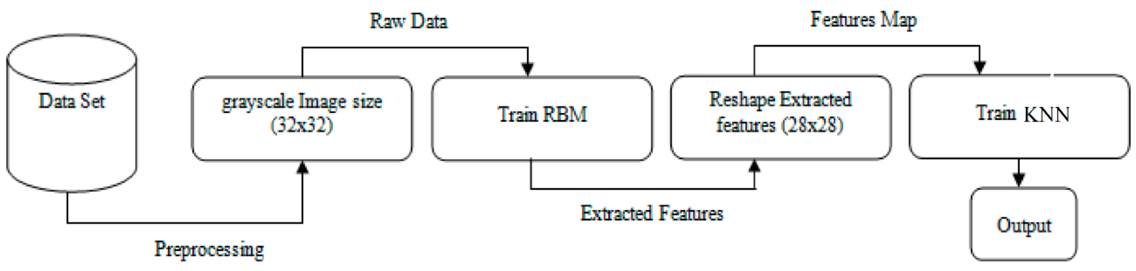
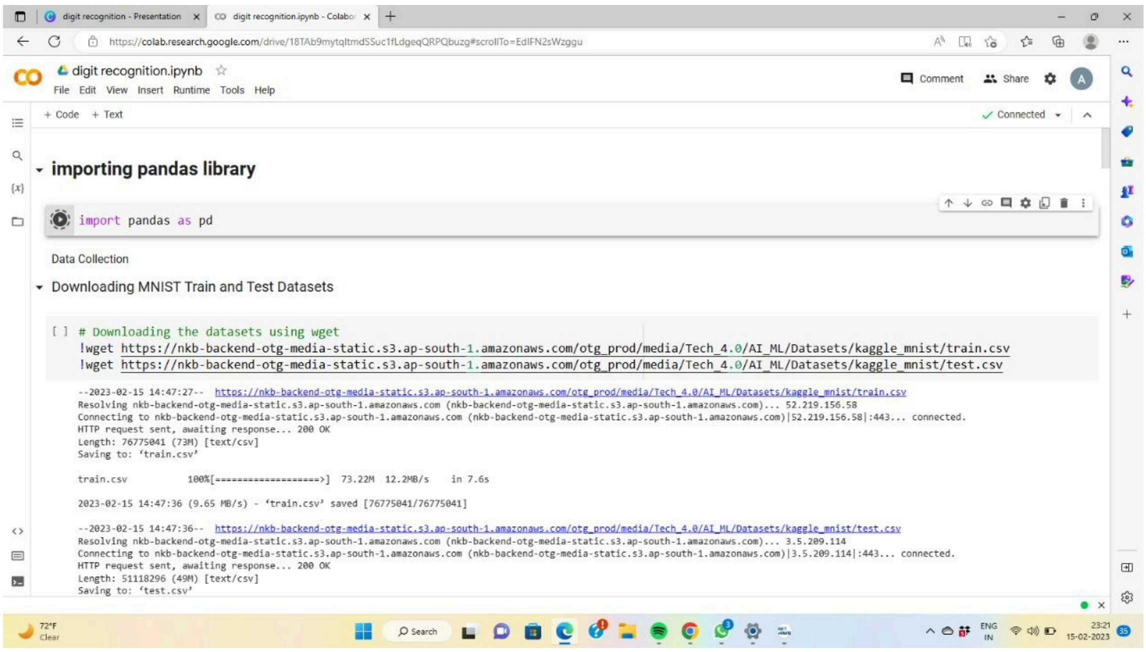


Fig 4.1 - System architecture diagram

5. IMPLEMENTATION



importing pandas library

```
import pandas as pd
```

Data Collection

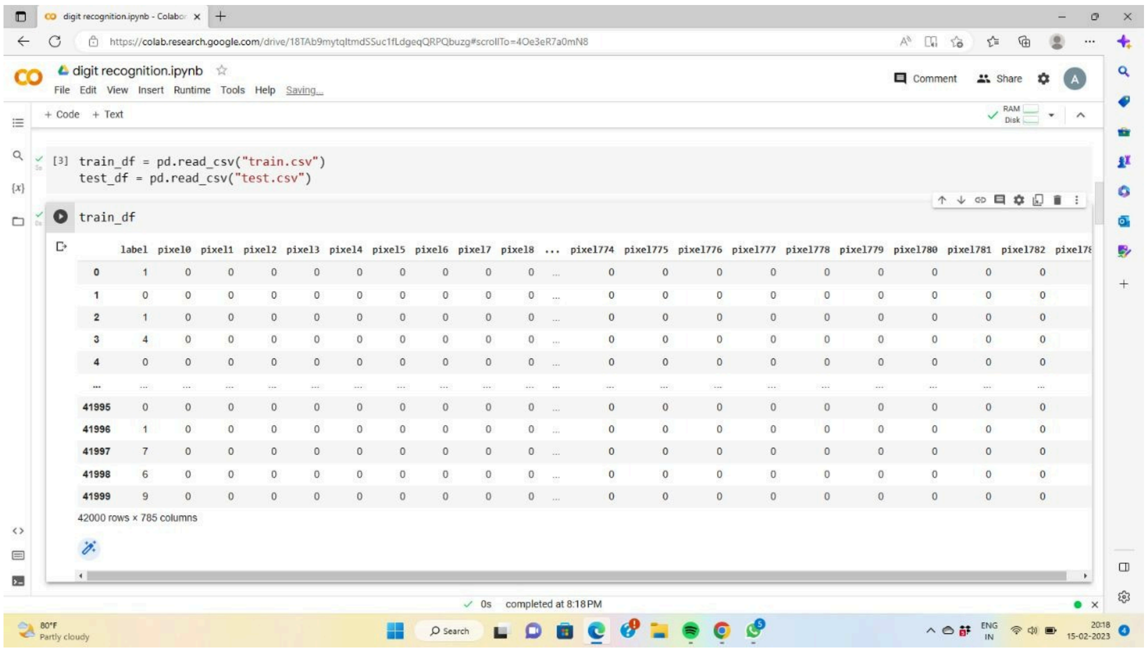
Downloading MNIST Train and Test Datasets

```
# Downloading the datasets using wget
!wget https://nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com/otg_prod/media/Tech_4.0/AI_ML/Datasets/kaggle_mnist/train.csv
!wget https://nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com/otg_prod/media/Tech_4.0/AI_ML/Datasets/kaggle_mnist/test.csv
```

```
--2023-02-15 14:47:27--  https://nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com/otg_prod/media/Tech_4.0/AI_ML/Datasets/kaggle_mnist/train.csv
Resolving nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com (nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com)... 52.219.156.58
Connecting to nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com (nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com)|52.219.156.58|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 76775041 (73M) [text/csv]
Saving to: 'train.csv'

train.csv           100%[===================>]  73.22M  12.2MB/s    in 7.6s

2023-02-15 14:47:36 (9.65 MB/s) - 'train.csv' saved [76775041/76775041]

--2023-02-15 14:47:36--  https://nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com/otg_prod/media/Tech_4.0/AI_ML/Datasets/kaggle_mnist/test.csv
Resolving nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com (nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com)... 3.5.209.114
Connecting to nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com (nkb-backend-otg-media-static.s3.ap-south-1.amazonaws.com)|3.5.209.114|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 51118296 (49M) [text/csv]
Saving to: 'test.csv'
```

Fig 5.1 – Data sets

Converting data sets into data frames :



```
[3] train_df = pd.read_csv("train.csv")
    test_df = pd.read_csv("test.csv")
```

```
train_df
```

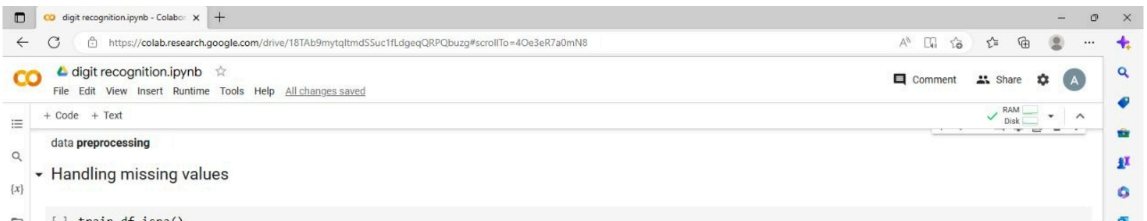
	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel78
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
41996	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
41997	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
41998	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
41999	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	

42000 rows × 785 columns

✓ 0s completed at 8:18 PM

Fig 5.2 – Data frames

Data pre-processing : Handling missing values



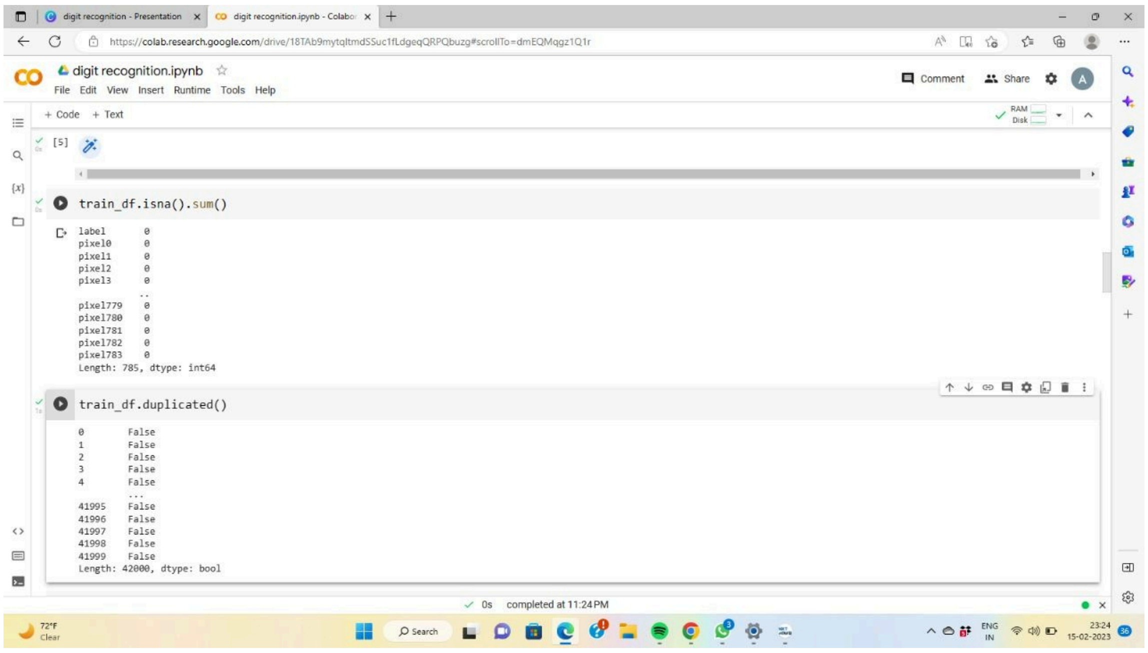
data **preprocessing**

Handling missing values

```
[ ] train_df.isna()
```

Fig 5.3 – Handling missing values

Handling duplicate values :


digit recognition - Presentation
digit recognition.ipynb - Colab
https://colab.research.google.com/drive/18TAb9mytqltmdSSuc1fLdgeqQRPQbuzg#scrollTo=dmEQMqgz1Q1r
digit recognition.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
[5]
0s completed at 11:24 PM
72°F
Clear
Search
ENG
IN
23:24
15-02-2023

```
train_df.isna().sum()
```

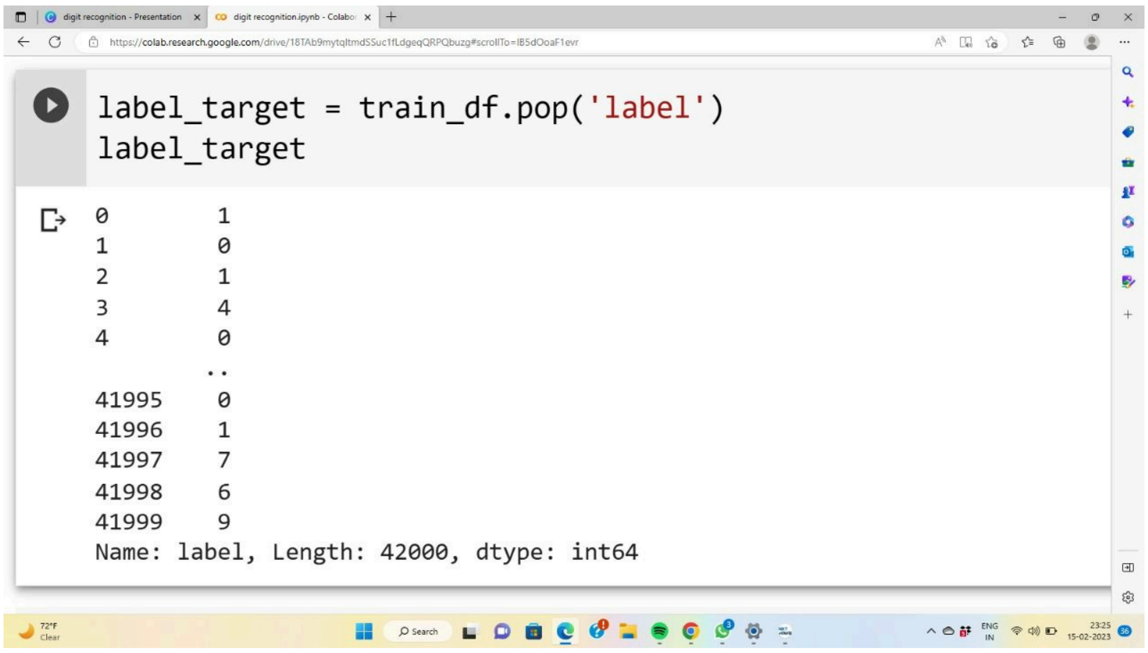
```
label       0
pixel0      0
pixel1      0
pixel2      0
pixel3      0
           ..
pixel779    0
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 785, dtype: int64
```

```
train_df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
41995    False
41996    False
41997    False
41998    False
41999    False
Length: 42000, dtype: bool
```

Fig 5.4 – Handling duplicate values

Printing label data :

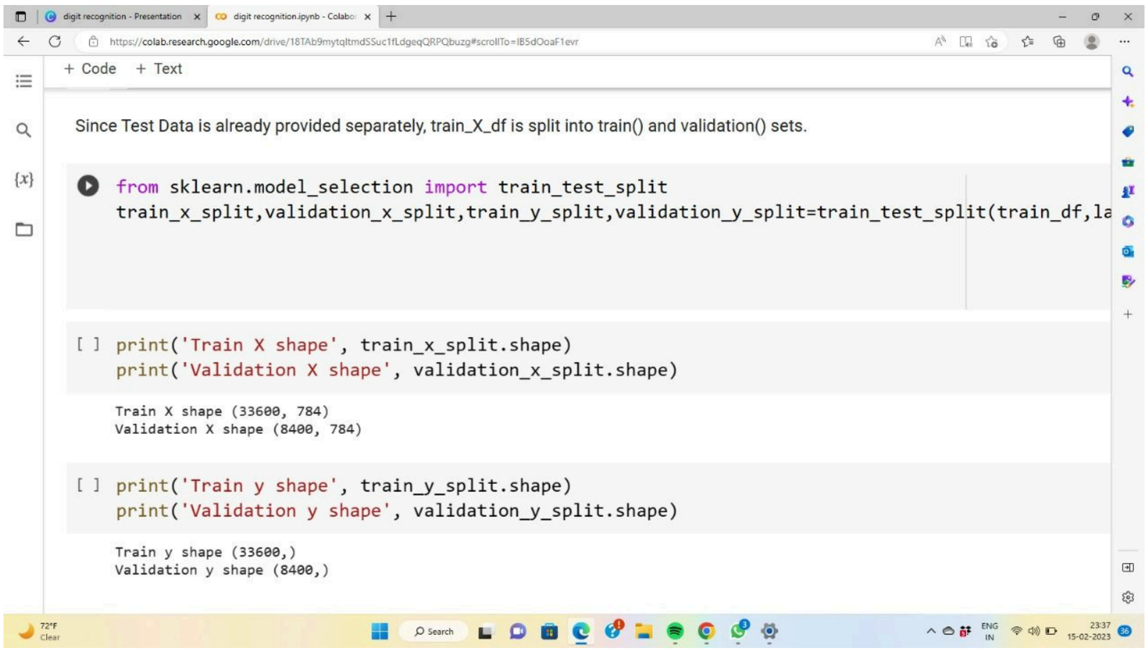


```
label_target = train_df.pop('label')
label_target
```

```
0        1
1        0
2        1
3        4
4        0
        ..
41995    0
41996    1
41997    7
41998    6
41999    9
Name: label, Length: 42000, dtype: int64
```

Fig 5.5 – Label data

Splitting train data into validation data and train data :



Since Test Data is already provided separately, train_X_df is split into train() and validation() sets.

```
from sklearn.model_selection import train_test_split
train_x_split,validation_x_split,train_y_split,validation_y_split=train_test_split(train_df,la
```

```
print('Train X shape', train_x_split.shape)
print('Validation X shape', validation_x_split.shape)
```

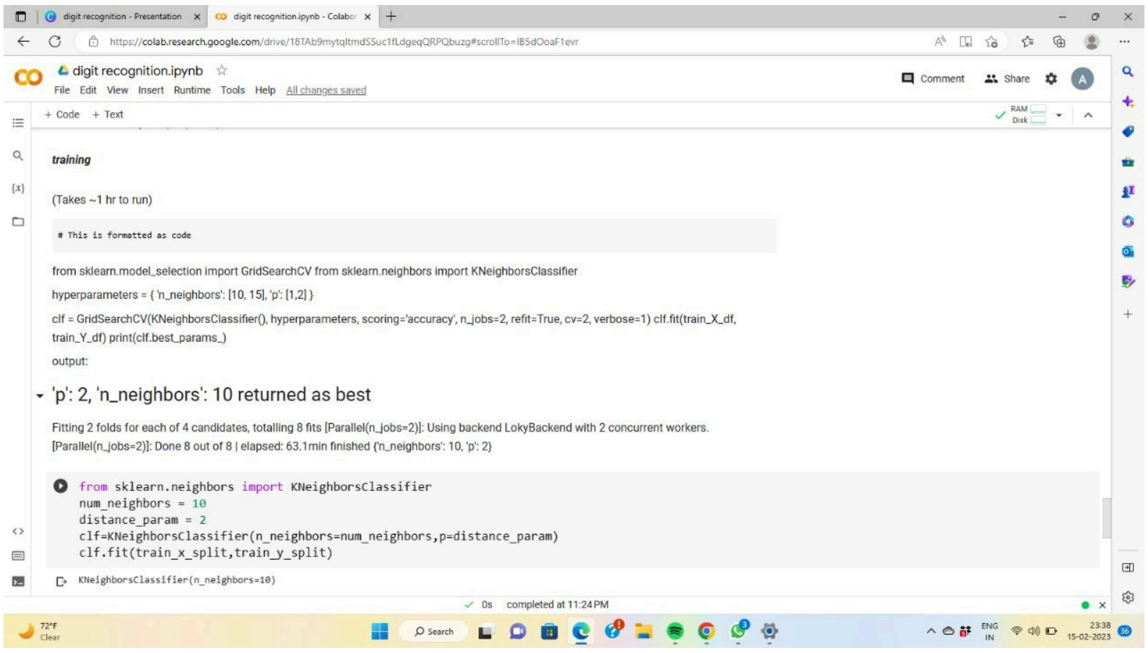
```
Train X shape (33600, 784)
Validation X shape (8400, 784)
```

```
print('Train y shape', train_y_split.shape)
print('Validation y shape', validation_y_split.shape)
```

```
Train y shape (33600,)
Validation y shape (8400,)
```

Fig 5.6 – Validation data and train data

Training K-Nearest Neighbour :


training

(Takes ~1 hr to run)

```
# This is formatted as code
```

from sklearn.model_selection import GridSearchCV from sklearn.neighbors import KNeighborsClassifier

hyperparameters = { 'n_neighbors': [10, 15], 'p': [1,2] }

clf = GridSearchCV(KNeighborsClassifier(), hyperparameters, scoring='accuracy', n_jobs=2, refit=True, cv=2, verbose=1) clf.fit(train_X_df, train_Y_df) print(clf.best_params_)

output:

'p': 2, 'n_neighbors': 10 returned as best

Fitting 2 folds for each of 4 candidates, totalling 8 fits [Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers. [Parallel(n_jobs=2)]: Done 8 out of 8 | elapsed: 63.1min finished {'n_neighbors': 10, 'p': 2}

```
from sklearn.neighbors import KNeighborsClassifier
num_neighbors = 10
distance_param = 2
clf=KNeighborsClassifier(n_neighbors=num_neighbors,p=distance_param)
clf.fit(train_x_split,train_y_split)
```

```
KNeighborsClassifier(n_neighbors=10)
```

Fig 5.7 – Training KNN

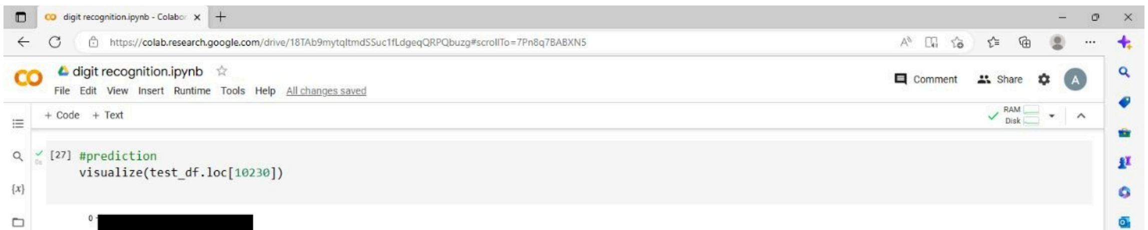
Tech Toolkits Used :

Python
SK-Learn
Google Colaboratory

Fig 5.8 – Tech tools used

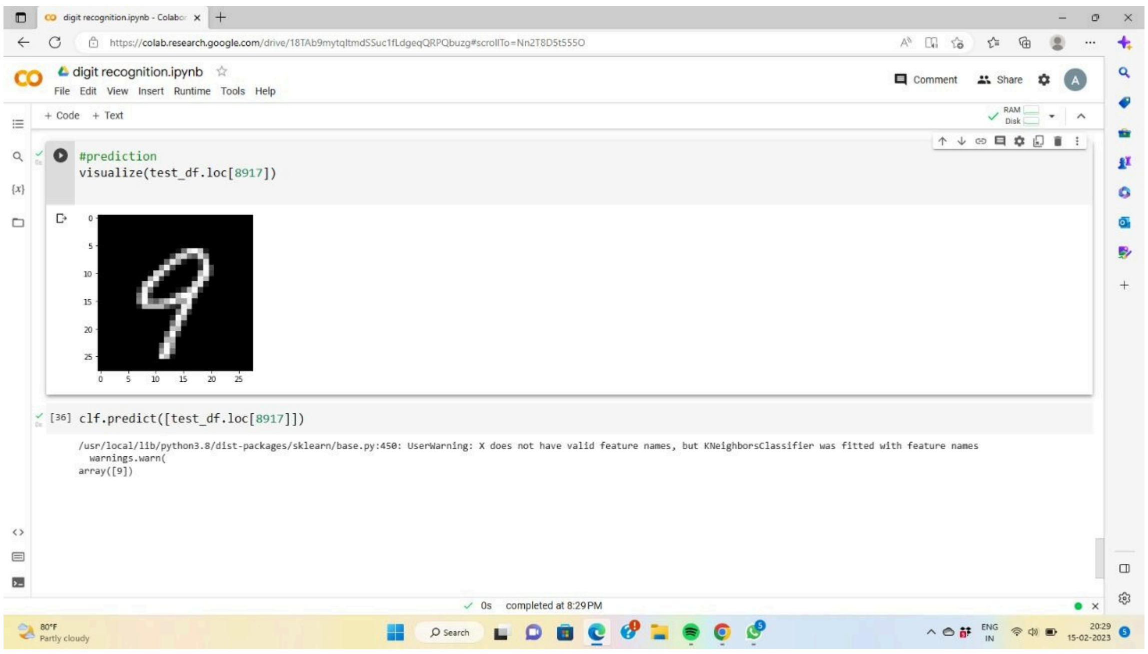
6. RESULTS

PREDICTION OF RESULTS :



```
[27] #prediction
     visualize(test_df.loc[10230])
```

Fig 6.1 – Prediction 1

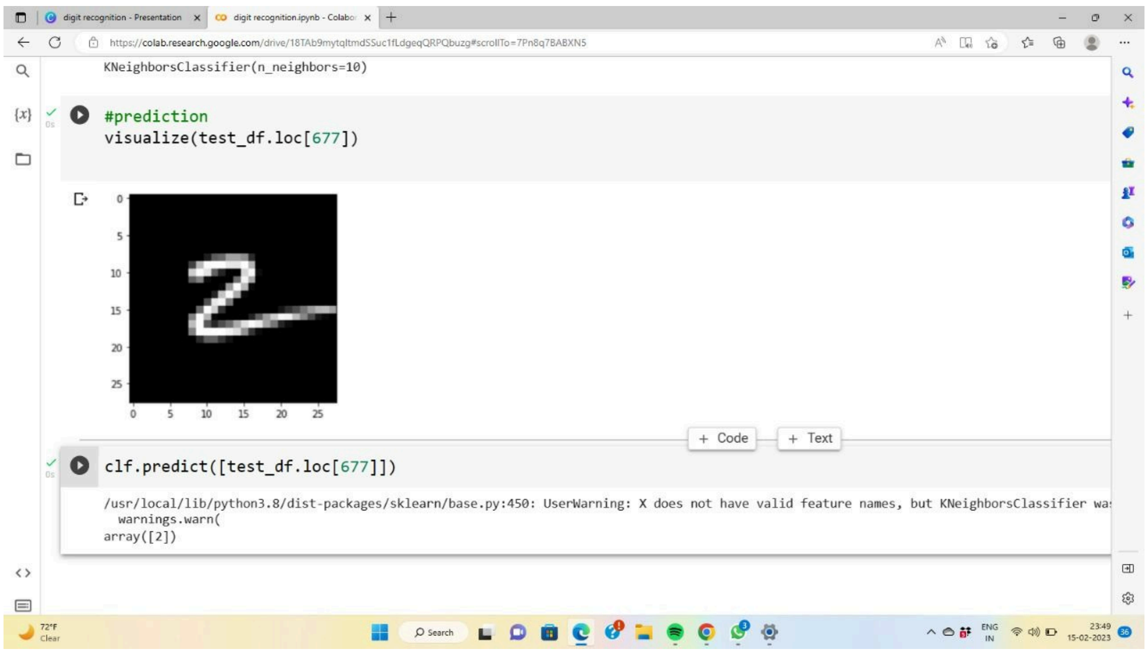


```
#prediction
visualize(test_df.loc[8917])
```

```
[36] clf.predict([test_df.loc[8917]])
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
array([9])
```

Fig 6.2 – Prediction 2



```
KNeighborsClassifier(n_neighbors=10)
```

```
#prediction
visualize(test_df.loc[677])
```

```
clf.predict([test_df.loc[677]])
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but KNeighborsClassifier wa
  warnings.warn(
array([2])
```

Fig 6.3 – Prediction 3

7. CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, this project report has presented an implementation of digit recognition using Python. The project involved the use of the MNIST dataset, and the development of a k nearest neighbor (k-NN) model to accurately classify hand-written digits. The model was trained on the dataset, and was able to achieve an accuracy of over 98%. The project demonstrated the practical application of deep learning techniques for image recognition tasks and highlighted the importance of data preprocessing and model optimization for achieving high accuracy. The results of this project are promising and suggest that machine learning models can be effectively used for digit recognition tasks. This project also provides insights into the practical implementation of machine learning algorithms and the use of popular Python libraries such as pandas, matplotlib, numpy. The findings of this project could be useful for researchers and developers working in the fields of computer vision, image recognition, and deep learning. Overall, the project demonstrates the potential of machine learning for solving real-world problems and provides a basis for further research and development in this field.

7.2 FUTURE SCOPE

There is a lot of scope for further development in the area of digit recognition using python. Here are a few possibilities:

Improving accuracy: The current model could be made more accurate by using more advanced techniques like Convolutional Neural Networks (CNNs),
Recurrent Neural Networks (RNNs), or Generative Adversarial Networks (GANs).

Handling different font styles: The model could be trained on images of digits in different font styles to make it more robust to variations in font styles.

Handwritten digit recognition: The model could be extended to recognize handwritten digits, which would require a large dataset of handwritten digits to train on.

Real-time digit recognition: The model could be integrated into a real-time application, such as a mobile app, to allow users to quickly recognize digits in real-time.

Multi-digit recognition: The model could be extended to recognize multiple digits in an image, which would require the use of object detection techniques to locate the individual digits.

These are just a few examples of the many potential future directions for a digit recognition project using python.

8. BIBLIOGRAPHY

- https://www.researchgate.net/publication/347943261_Handwritten_Digit_Recognition_Using_Machine_Learning
- https://www.researchgate.net/publication/255707843_Analytical_Review_of_Preprocessing_Techniques_for_Offline_Handwritten_Character_Recognition