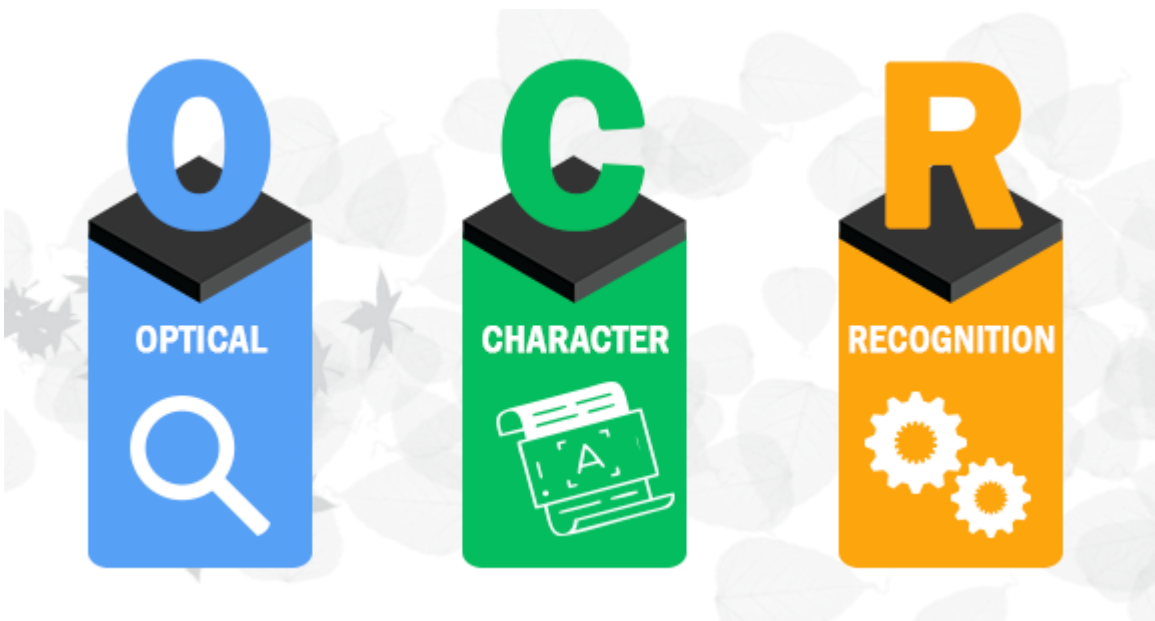

Artificial Intelligence Documentation



Code is Available on :

1. [GitHub \(without dataset\)](#)
2. [Google Drive \(with dataset\)](#)

Written By:

NAME	Department	ID
Mohamed Ashraf Abu Al-Maati	CS	20210742
Mahmoud Nasr Abdelaziz	CS	20210883
Mostafa Waleed Said	IS	20210948
Andrew Atef Shakrallah	CS	20210191
Anas Saleh Mousa	CS	20210192
Mohamed Ehab Mohamed	CS	20210753
Omar Magdy Abdelaziz	CS	20210615

Idea

- Optical Character Recognition (OCR)

Main functionalities

- Image source
- Pre-processing
- segmentation
- Feature extraction
- Classification

Applications

- Google Translate
- Cambridge Assessment
- OCR Applications in Banking
- OCR Use Cases in Healthcare OCR machine
- Stadtwerke München

Papers

- First paper
- Secon paper
- Third paper
- Fourth paper
- Fifth paper

Dataset

- Train
- Test
- Mapping

Details of the algorithm

- what are decision tree ?
- what are random forest ?
- how do decision tree ?
- how do random forest ?

DIAGRAMS

- Activity Diagram
- Flowchart
- Use case

Model

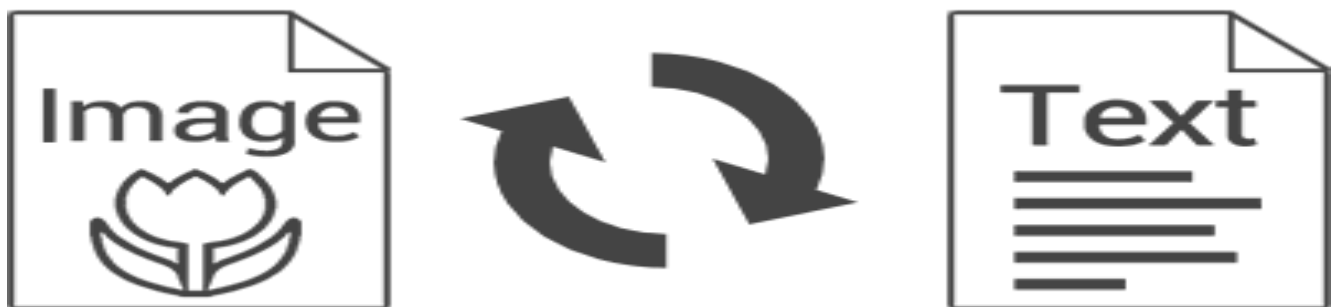
- Preparing the data
- Building the Model
- Training the Model
- Save & Load
- Testing the Model

Development platform

Optical Character Recognition "OCR"

What is OCR??

OCR (optical character recognition) is the use of technology to distinguish printed or handwritten text characters inside digital images of physical documents, such as a scanned paper document. The basic process of OCR involves examining the text of a document and translating the characters into code that can be used for data processing. OCR is sometimes also referred to as text recognition. Hardware, such as an optical scanner or specialized circuit board, is used to copy or read text while software typically handles the advanced processing. Software can also take advantage of artificial intelligence (AI) to implement more advanced methods of intelligent character recognition (ICR), like identifying languages or styles of handwriting.



What is Scene Text Recognition (STR)?

In computer vision, machines can read text in natural scenes by first detecting text regions, cropping those regions, and subsequently recognizing text in those regions. The vision task of recognizing text from the cropped regions is called Scene Text Recognition (STR). STR makes it possible to read road signs, billboards, logos, and printed objects such as text on shirts, paper bills, etc. STR applications include practical use cases such as self-driving cars, augmented reality, retail analysis, education, devices for the visually impaired, and others.

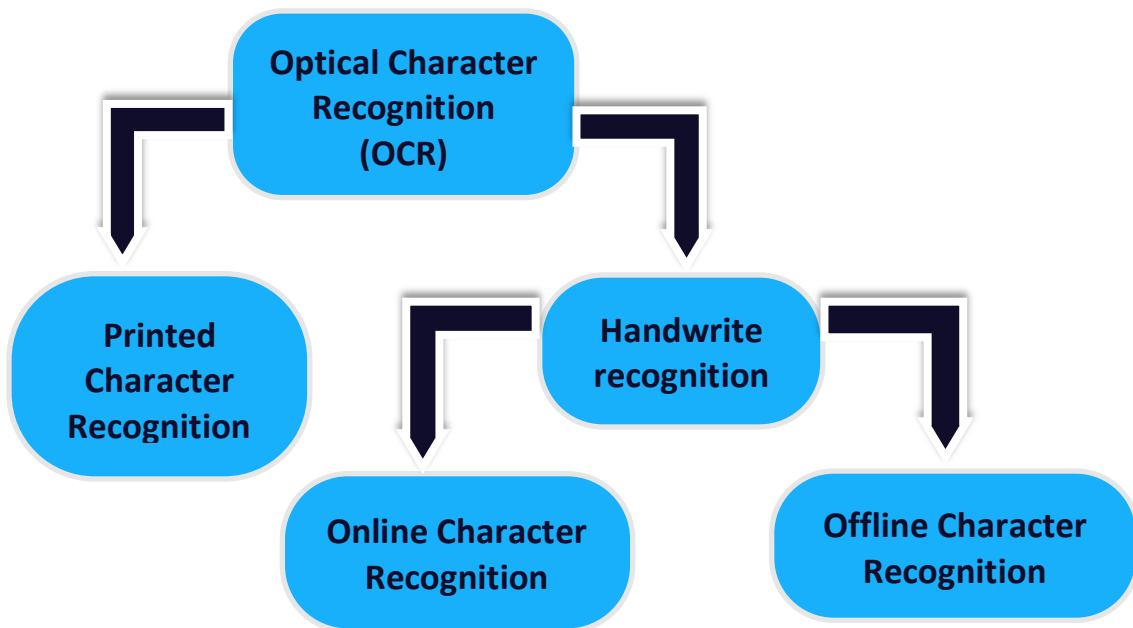


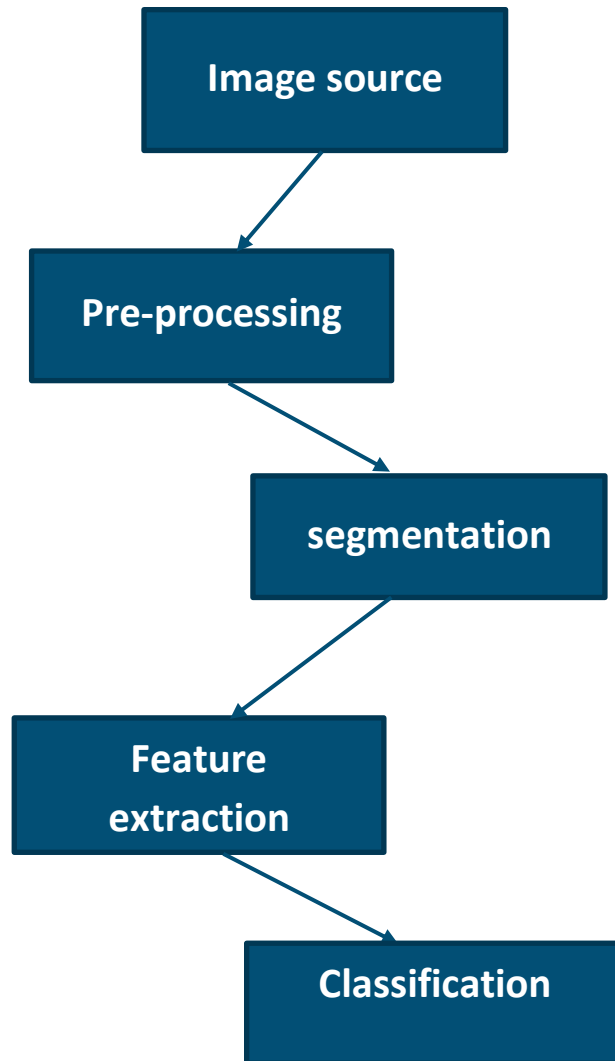
How Artificial Intelligence Gives OCR a Boost?

Artificial intelligence is transforming the capabilities of optical character recognition (OCR) tools. An area of computer vision, OCR processes images of text and converts that text into machine-readable forms. In other words, it takes handwritten or typed text within physical documents and converts them into digital formats.

Process and function of Optical Character Recognition (OCR)

Handwrite recognition is one of the types of Optical Character Recognition (OCR). OCR is identification of text, which may be printed or hand-written. In OCR, the document is captured via camera as image and can be converted to desired formats like PDFs. Then the file is fed to the algorithm for character recognition





1. Image source

This phase comes in offline hand-written character recognition. Image source can be from any digitized tool.

2. Pre-processing:

that improves the quality of image and hence increases the accuracy of image. For the hand-written character recognition process, the following pre-processing techniques are followed.

3. Segmentation:

Segmentation is a mechanism that extracts individual characters in the image. There exist two types of segmentation. They are Implicit segmentation and Explicit segmentation. In implicit segmentation the words are recognized without segmentation process. But in explicit segmentation, words are predicted by extracting individual character.

4. Feature-extraction

This is the important phase in the recognition process, and the algorithm of recognition starts from here. Each character contains its own features. It contains group of rules where each rule explains feature of a character. Extraction of such features is done in this phase. 5. Classification By this time, the training would have completed, and the testing of input data starts. The testing data would pass all the above process and the varying probabilities are assigned to the matching rules. The rule with highest probability is selected and the corresponding class-label is made recognized character.

5. Classification

By this time, the training would have completed, and the testing of input data starts. The testing data would pass all the above process and the varying probabilities are assigned to the matching rules. The rule with highest probability is selected and the corresponding class-label is made recognized character

Applications of Optical Character Recognition in real life:

Google Translate:

- One of the most widely used OCR applications stems from Google Translate's OCR addition, which revolutionized the way we communicate in different languages. Instead of typing text into the app like before, you can simply point your mobile phone camera at text, and it will be scanned into Google Translate via OCR.

- This is extremely useful, and it can sometimes be the only viable method of translation. If you're in a foreign country and you see a signboard, even with a normal translation app, you won't necessarily know what it means if you can't speak the language. Some



languages don't use the same script style as English, so you likely won't even be able to input the correct characters into the translation app. Bringing OCR to the translation game changes everything — you don't need to know the language to understand it. Google's advanced OCR can identify and parse the language into its system, providing instant identification of what has been written. For millions of travelers worldwide, this has been a game-changer, especially in countries with vastly differing language scripts (China, Japan, Russia, India, etc.).

Cambridge Assessment:

- is another company successfully using OCR to automate complex tasks. Cambridge Assessments runs some of the world's biggest examinations, including GCSEs, IGCSEs, and Cambridge A Levels. They process over one million physical papers per year. Examination papers are handwritten by students, and Cambridge examiners work remotely from all over the world, making it infeasible to send physical copies of answer scripts. To solve their problem, Cambridge Assessments turned to technology.
- Simple scanning and replicating won't really do the job: although it transmits the papers electronically to the examiners, the process makes marking much harder. Examiners need to manually go through each answer script and have to sift out the relevant information.



Instead, Cambridge Assessments uses OCR

technology to scan each paper that passes through their system. This allows for proper documentation and sorting of answers. For example, with OCR technology, papers can be scanned into the system, and computers can group together sets of papers based on age range, location, examination type, examination level, and more. Humans don't need to physically sit at sorting centers to group papers together—OCR makes it possible via computers. Additionally, OCR allows answers to be separated by question. For example, if an examiner wants to mark “question 2” from a given paper, OCR technology can scan and separate out all of the “question 2s” from all the papers so that the examiner can mark them all at once. This is far more efficient and faster than having an examiner manually open each document, scroll to question 2, and then mark all of them one by one.

Stadtwerke München:

- a communal company owned by the city of Munich, was one of the first major government-led organizations to adopt a novel form of OCR—license plate scanning. The company offers public services such as public transport and parking management, and as such, parking inspection became an important part of their work. However, manual parking inspection can be quite a chore—to ensure accuracy, parking inspectors were made to input the license plate number into the system twice, manually. Not only was this cumbersome, it was also a waste of resources that could have easily been deployed elsewhere.
- The city of Munich found their solution in Any line, an OCR app that allowed parking inspectors to accurately input license plate numbers into their system in under one second. This allows them to save a lot of time and effort during parking management. A spokesperson for Stadtwerke München said OCR has “made it much faster for our parking attendants to collect license plate data. This efficiency improvement at the base of our parking control operations will help us to improve parking control within Munich.”



OCR Applications in Banking:

- The banking industry is deemed one of the largest consumers of OCR recognition apps as it helps enhance security, improve data management, optimize risk management, and enhance customer experience. Before applying OCR technology, most banking documents were physical, including customer records, checks, bank statements, and others. Using an OCR recognition solution, it becomes possible to digitize and store even older documents in databases. OCR technology has also completely revolutionized the banking industry by:
 - Providing easy verification: OCR allows a real-time verification of money deposit checks and a signature by scanning them using an OCR-based application. An example of this can be seen in mobile banking apps, where checks can be deposited digitally and processed within days through OCR-based check depositing features.
 - Enhancing security: The electronic deposition of checks through OCR technology results in fraud prevention and increasingly secure transactions, fostering a better user experience. OCR can use the character reader count and machine learning methods to detect forged documents.

ACE BANK
ACE BANK, P.O. Box 659754
San Jose, CA 95165-9754

CUSTOMER SERVICE INFORMATION
Web site: acebank.com
Service Phone: 1-800-814- 9025 Deaf and Hard of Hearing: 1-800-240-1997
International Calls: 1-877-111-4132
Para Espanol: 1-711-131-6277
Primary Account: 00000086532944

STEVEN PRI NCENTER
2481 SHELDEN ST APT 305 PARIS TX
71261-0140

Statement period : January 27, 2017 through February 24, 2017

SUMMARY	
Beginning Balance	\$2,791.56
Deposits and Additions	28,037.41
Checks Paid	-2,080.00
Electronic Withdrawals	-12,367.35
Ending Balance	\$16,401.56

Annual Percentage Yield Earned This Period: 0.07%
Interest Paid This Period: \$0.06
Interest Paid Year-to-Date: \$0.09

Transaction Summary

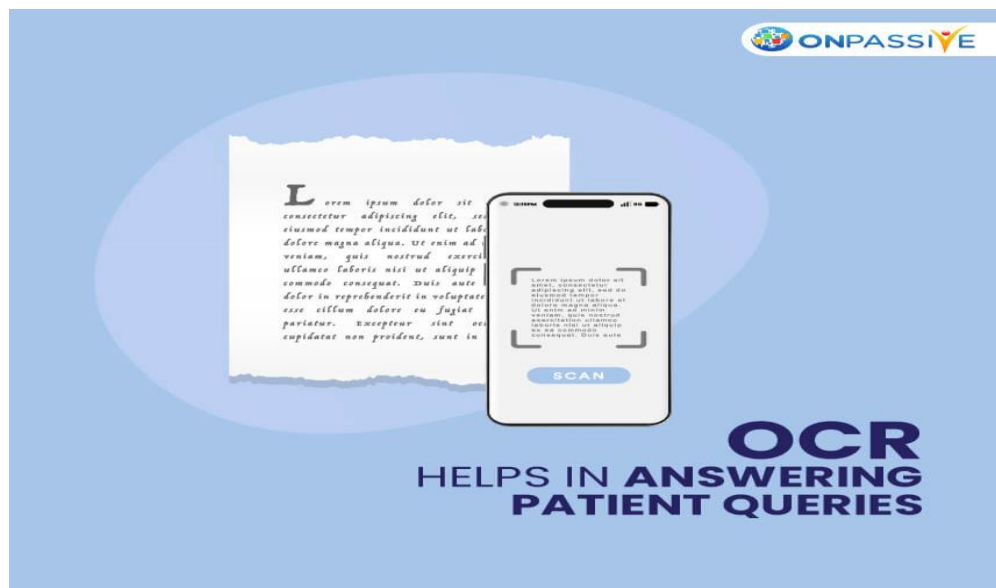
Ref. No.	Transaction Date	Transaction Description	Transaction Amount
38405	01/27/2017	Van Transfer van jose	-55,800.00
8349	01/29/2017	AutoDeposit	\$20,000.00
1494	02/01/2017	Cash Deposit	\$8,037.51
135424	02/08/2017	Check -89	-2,080.00
3435	02/15/2017	ZenWithdrawal	-85,567.28
0000	02/24/2017	Ending Balance	\$16,401.56

API response

```
{
  "Basic Information": {
    "Bank Name": "ACE BANK",
    "PO Box": "659754",
    "City": "",
    "State": "",
    "User Name": "Steven Pri Ncenter",
    "Address": "2481 Shelden St. Apt. 305 Paris",
    "State": "",
    "City": "",
    "Period": "27/01/2017 - 24/02/2017",
    "Zip": ""
  },
  "Account Information": {
    "Account": "000000865332944",
    "Opening Balance": "$2,791.56",
    "Closing Balance": "$16,401.56"
  },
  "Line Items": [
    {
      "Ref. No.": "38405",
      "Date": "27/01/2017",
      "Description": "Van Transfer-san Jose",
      "Amount": "$16,401.56",
      "Ref. No.": "8349",
      "Date": "29/01/2017",
      "Description": "Auto Deposit"
    }
  ]
}
```


• OCR Use Cases in Healthcare OCR machine:

- learning has proved to be beneficial for the healthcare industry. In the healthcare sector, OCR technology allows patient medical histories to be accessed digitally by patients and doctors alike. In addition, patient records, including their X-rays, treatments, tests, hospital records, and insurance payments, can easily be scanned, searched, and stored using OCR full form methods to digitize records and read labels with cameras. Thus, optical character recognition helps streamline the workflow and reduce manual work at hospitals while keeping the records up to date.



Academic publications (papers):

- **First paper**

Image:



Figure 2 Basic Operation of digital conversion

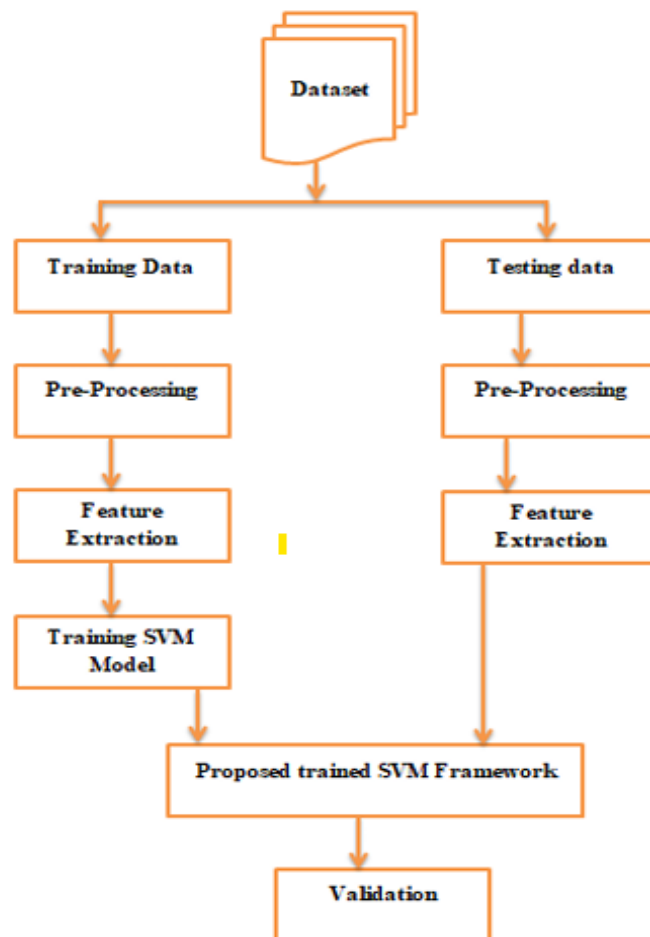


Figure 5 Classification process of proposed SVM framework

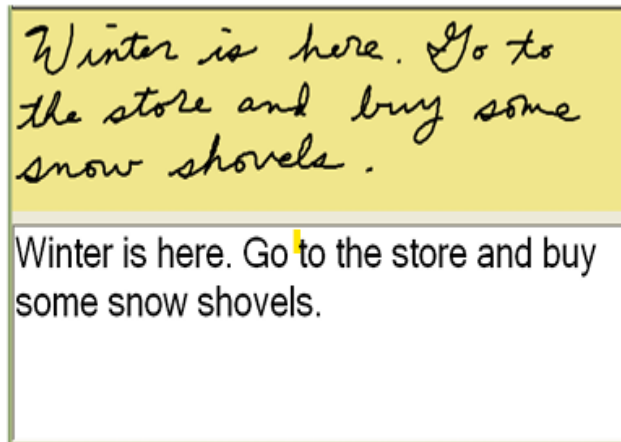


Figure 8 Output results of stylish text characters recognition after examination

Description:

the research paper is comparing statistical approach support vector machine (SVM) classifiers network method with statistical, template matching, structural pattern recognition, and graphical methods. It has proved Statistical SVM for OCR system performance that is providing a good result that is configured with machine learning approach.

• Second paper

Image:



Figure 8: Examples where model makes mistakes

Description:

In this paper, we propose a CNN based architecture that bridges this gap. Thier work, Easter2.0, is composed of multiple layers of 1D Convolution, Batch Normalization, ReLU, Dropout, Dense Residual connection, Squeeze-and-Excitation module and make use of Connectionist Temporal Classification (CTC) loss. In addition to the Easter2.0 architecture, we propose a simple and effective data augmentation technique 'Tiling and Corruption (TACo)' relevant for the task of HTR/OCR. Our work achieves state-of-the-art results on IAM handwriting database when trained using only publicly available training data.

- **Third paper:**

Image:

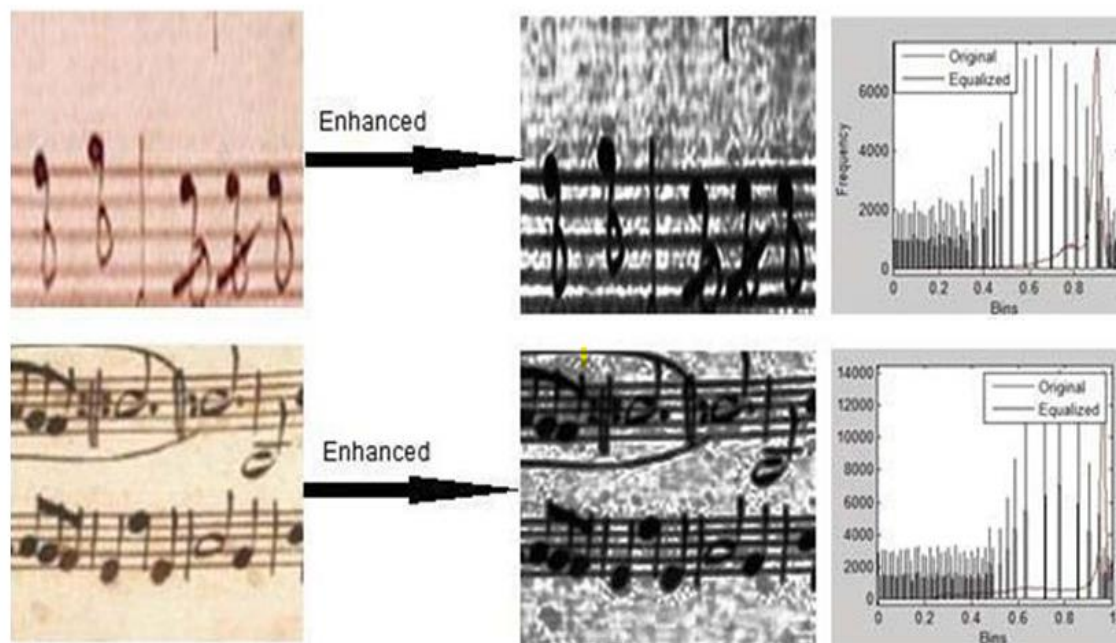


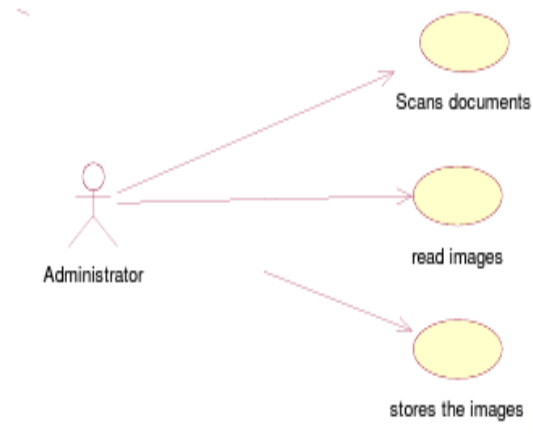
Fig. 2. Segmentation results for optical Music recognition

Description:

The paper presents a survey of applications of OCR in different fields and further presents the experimentation for three important applications such as Captcha, Institutional Repository and Optical Music Character Recognition. We make use of an enhanced image segmentation algorithm based on histogram equalization using genetic algorithms for optical character recognition. The paper will act as a good literature survey for researchers starting to work in the field of optical character recognition.

- **Fourth paper:**

Image:



V. Results And Discussion

The main purpose of Optical Character Recognition (OCR) system based on a grid infrastructure is to perform Document Image Analysis, document processing of electronic document formats converted from paper formats more effectively and efficiently. This improves the accuracy of recognizing the characters during document processing compared to various existing available character recognition methods. Here OCR technique derives the meaning of the characters, their font properties from their bit-mapped images.

Description:

Optical character recognition refers to the branch of computer science that involves reading text from paper and translating the images into a form that the computer can manipulate (for example, into ASCII codes). An OCR system enables you to take a book or a magazine article, feed it directly into an electronic computer file, and then edit the file using a word processor. All OCR systems include an optical scanner for reading text, and sophisticated software for analyzing images. Most OCR systems use a combination of hardware (specialized circuit boards) and

software to recognize characters, although some inexpensive systems do it entirely through software. Advanced OCR systems can read text in large variety of fonts, but they still have difficulty with handwritten text. It is the mechanical or electronic translation of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used to convert books and documents into electronic files, to computerize a record-keeping system in an office, or to publish the text on a website.

- **Fifth paper:**

Image:

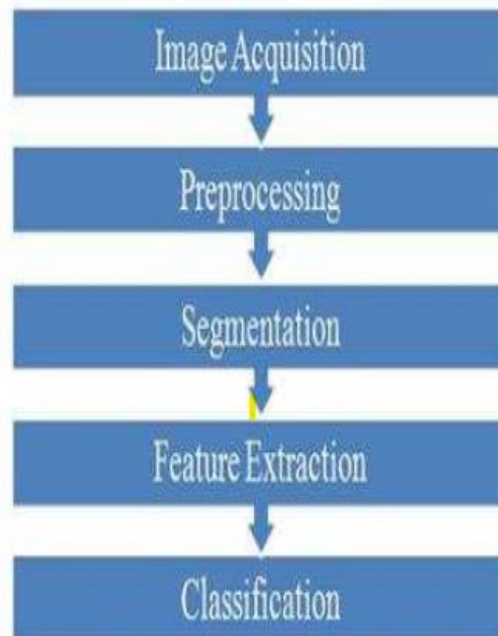


Fig 2: Offline Handwritten Character Recognition Architecture

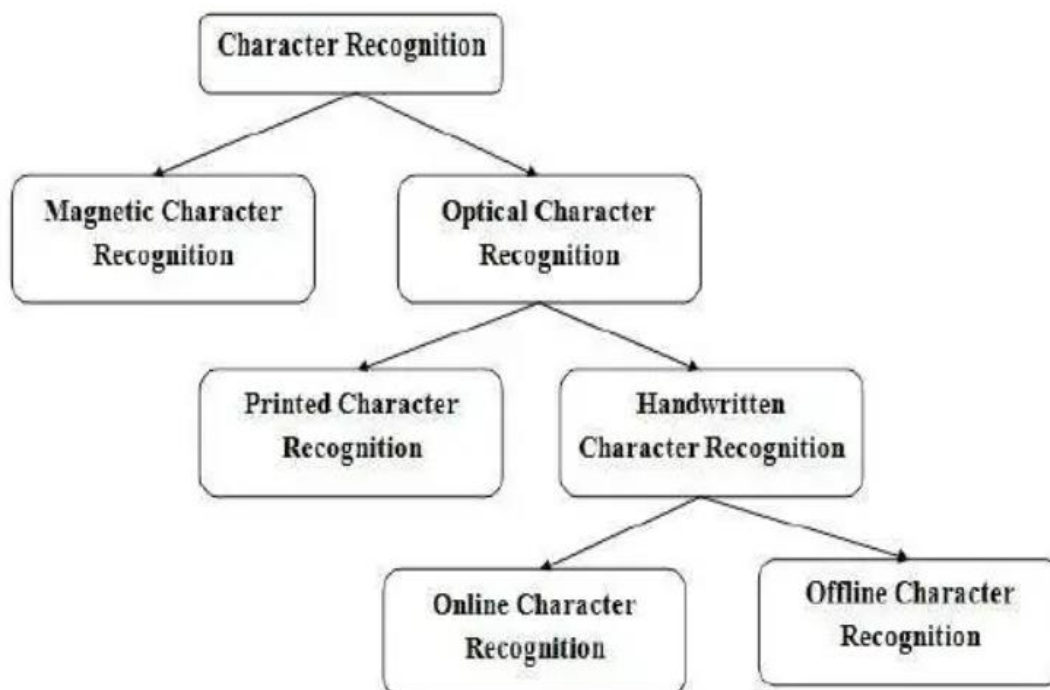


Fig 1: Classification of Character Recognition

Description:

This paper presents a detailed review of Offline Handwritten Character Recognition. HCR is an optical character recognition, which convert the human readable character to machine readable format. In HCR, to attain 99% accuracy is very difficult.

Dataset

Train:

<https://www.kaggle.com/datasets/crawford/emnist?select=emnist-balanced-train.csv>

Test:

<https://www.kaggle.com/datasets/crawford/emnist?select=emnist-balanced-test.csv>

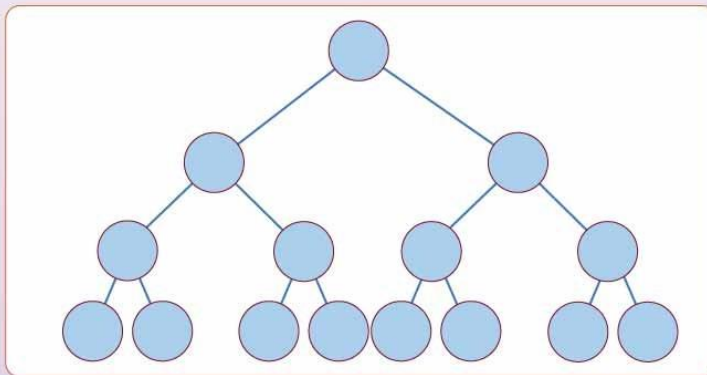
Mapping:

<https://www.kaggle.com/datasets/crawford/emnist?select=emnist-balanced-mapping.txt>

What is decision tree.?!

A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are used for classification and regression tasks, providing easy-to-understand models.

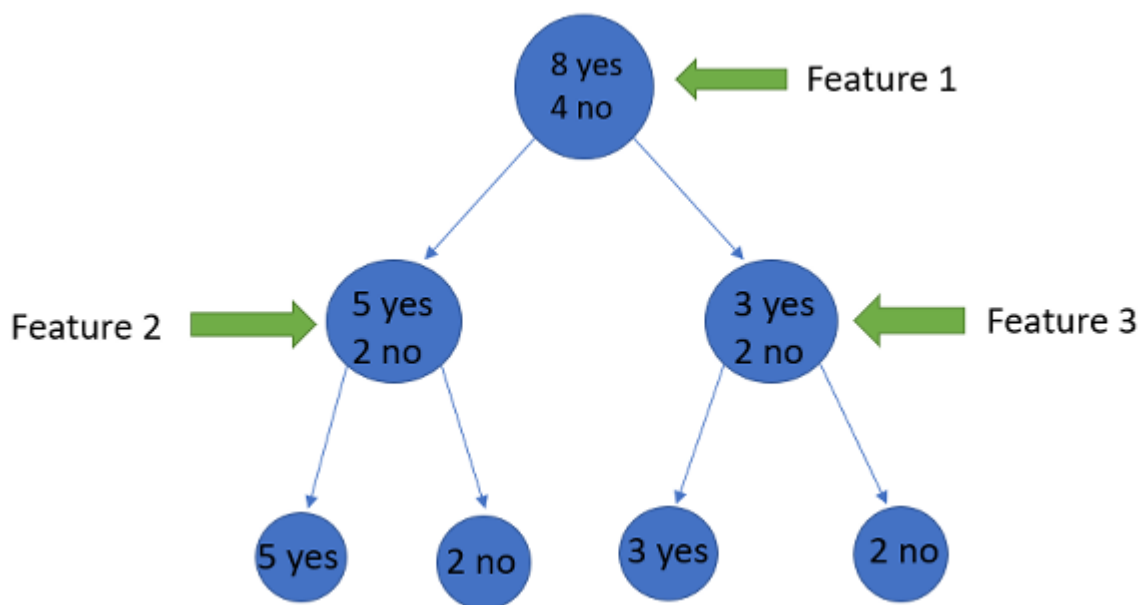
Understanding Decision Tree Algorithm In Machine Learning



A decision tree is a hierarchical model used in decision support that depicts decisions and their potential outcomes, incorporating chance events, resource exp

How does decision tree work.?!

Decision trees is a type of supervised machine learning algorithm that is used by the Train Using AutoML tool and classifies or regresses the data using true or false answers to certain questions. The resulting structure, when visualized, is in the form of a tree with different types of nodes—root, internal, and leaf. The root node is the starting place for the decision tree, which then branches to internal nodes and leaf nodes. The leaf nodes are the final classification categories or real values. Decision trees are easy to understand and are explainable.



Decision trees use a recursive partitioning process, where each node is divided into child nodes, and this process continues until a stopping criterion is met. This assumes that data can be effectively subdivided into smaller, more manageable subsets.

Root Node: The initial node at the beginning of a decision tree, where the entire population or dataset starts dividing based on various features or conditions.

Decision Nodes: Nodes resulting from the splitting of root nodes are known as decision nodes. These nodes represent intermediate decisions or conditions within the tree.

Leaf Nodes: Nodes where further splitting is not possible, often indicating the final classification or outcome. Leaf nodes are also referred to as terminal nodes.

Sub-Tree: Similar to a subsection of a graph being called a sub-graph, a subsection of a decision tree is referred to as a sub-tree. It represents a specific portion of the decision tree.

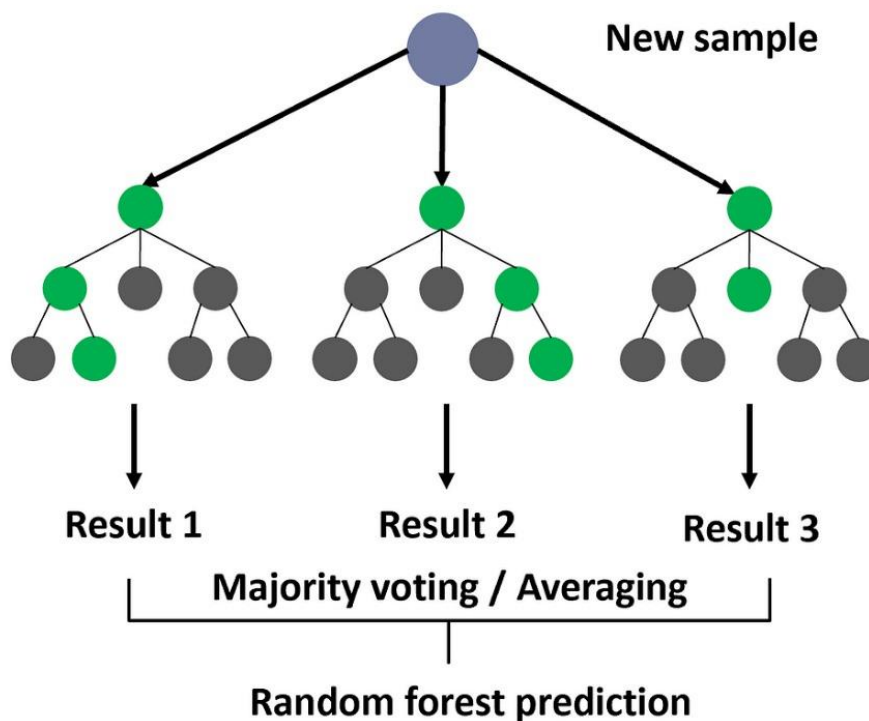
Pruning: The process of removing or cutting down specific nodes in a decision tree to prevent overfitting and simplify the model.

Branch / Sub-Tree: A subsection of the entire decision tree is referred to as a branch or sub-tree. It represents a specific path of decisions and outcomes within the tree.

Parent and Child Node: In a decision tree, a node that is divided into sub-nodes is known as a parent node, and the sub-nodes emerging from it are referred to as child nodes. The parent node represents a decision or condition, while the child nodes represent the potential outcomes or further decisions based on that condition.

What is random forest.?!

Random Forest is a widely-used machine learning algorithm developed by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems. In this article, we will understand how random forest algorithm works, how it differs from other algorithms and how to use it



Random Forest Algorithm widespread popularity stems from its user-friendly nature and adaptability, enabling it to tackle both classification and regression problems effectively. The algorithm's strength lies in its ability to handle complex datasets and mitigate overfitting, making it a valuable tool for various predictive tasks in machine learning.

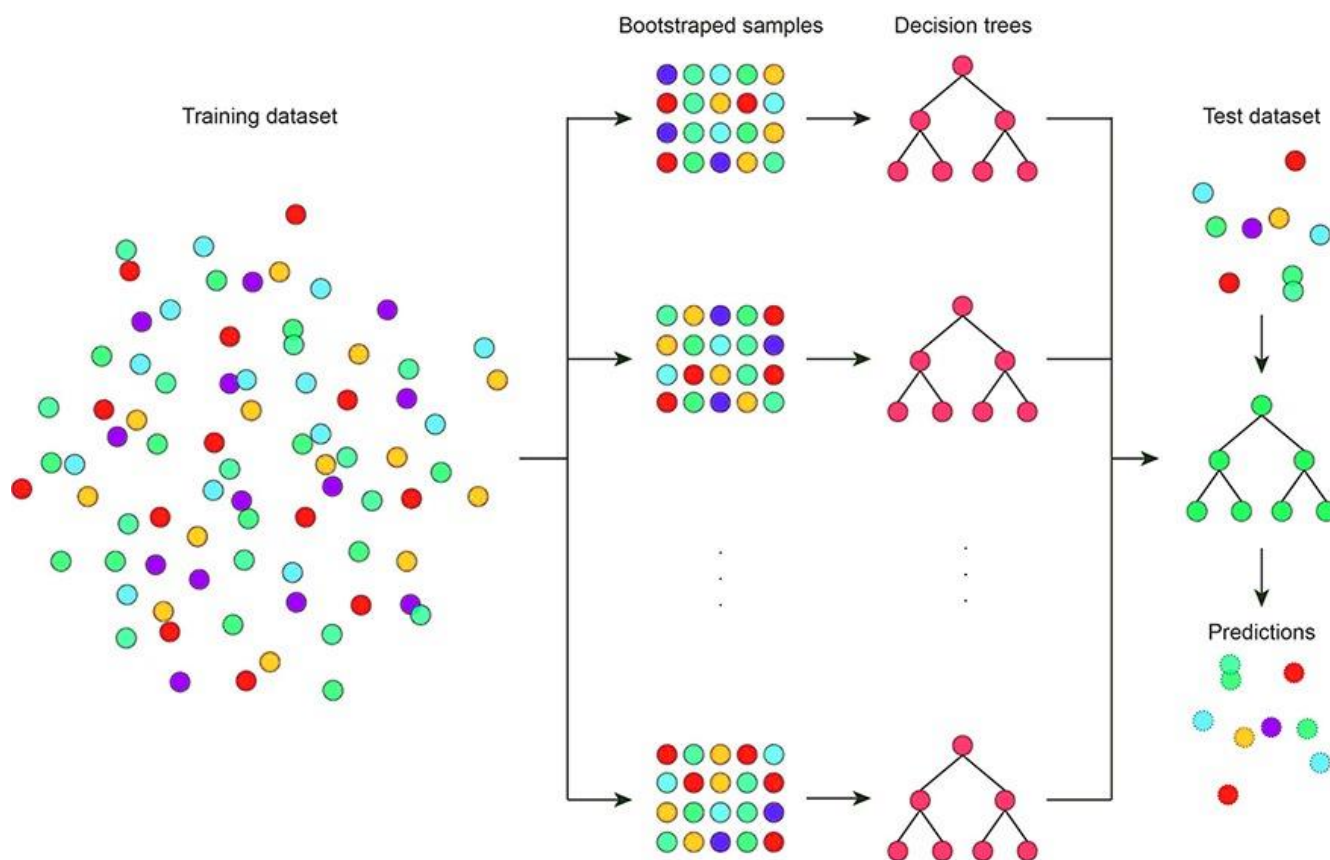
One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for

classification and regression tasks. In this tutorial, we will understand the working of random forest and implement random forest on a classification task.

How does random forest work.?!

Random Forest grows multiple decision trees which are merged together for a more accurate prediction. The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone.

The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees. The outcome chosen by most decision trees will be the final



The following steps can be used to demonstrate the working process: Step 1: Pick M data points at random from the training set. Step 2: Create decision trees for your chosen data points (Subsets). Step 3: Each decision tree will produce a result.

DIAGRAMS

1- Activity Diagram

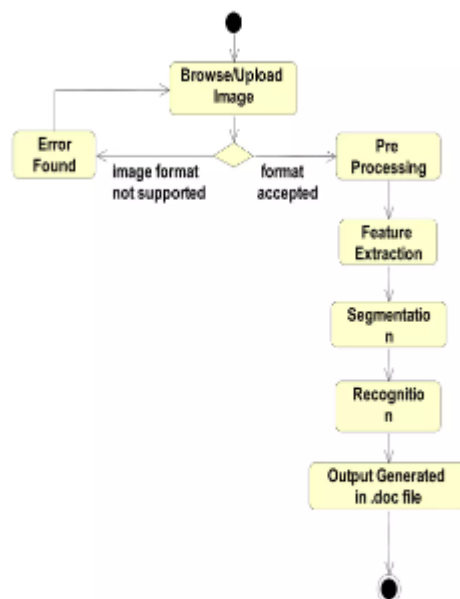
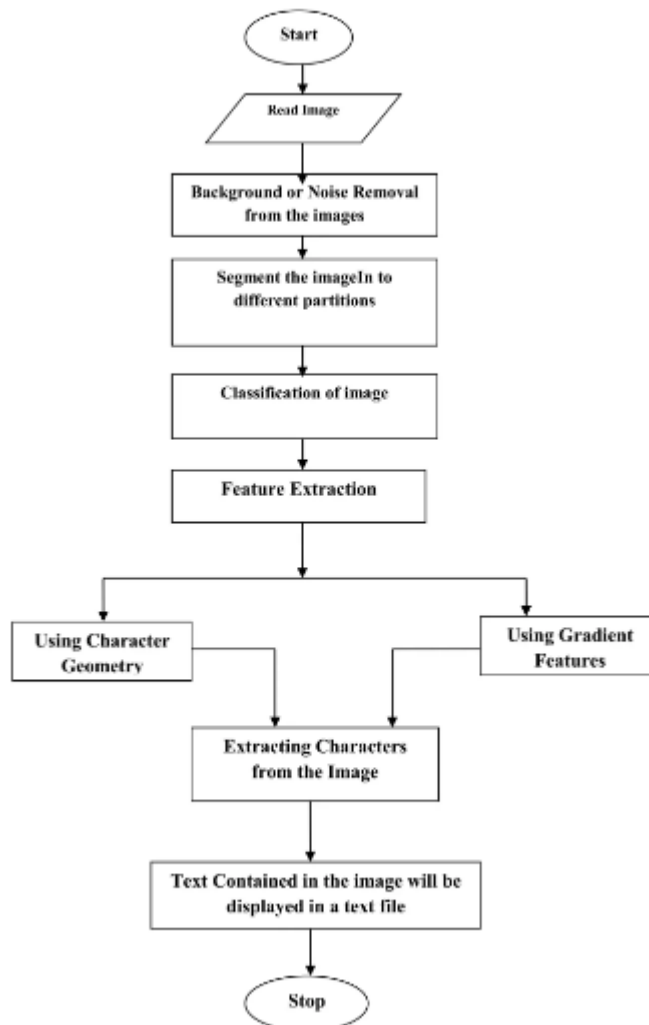


Figure 5.4.3: Activity Diagram

2- Flowchart



3- Use case

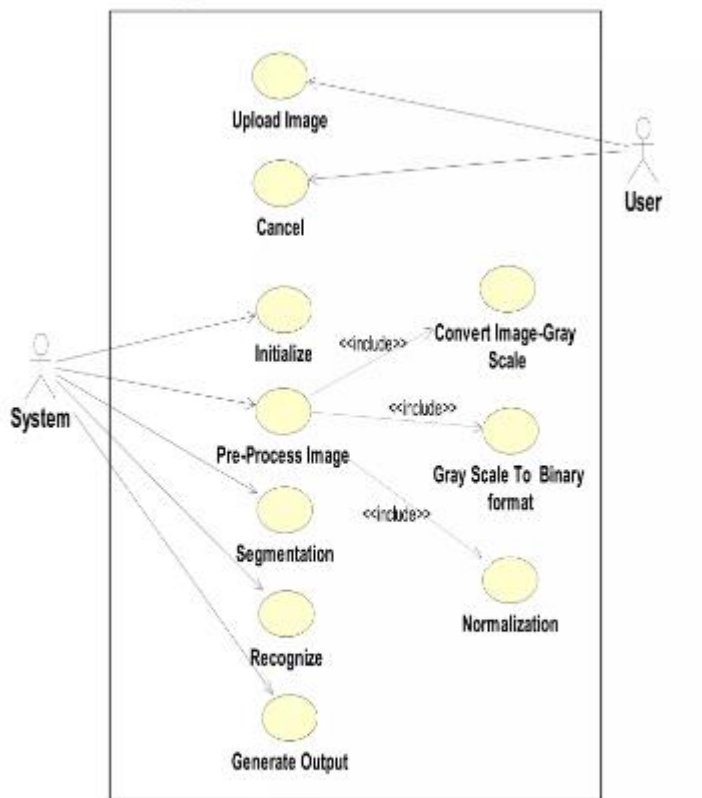


Figure 5.4.1: Use Case Diagram

model

Handwritten alphabet recognition model:

Let's go through the steps of creating, training, and testing the model.

We used all the letters to train our model with 112800 images, we split the images of each letter.

into training and testing. We use almost 85% of the images to train and 15% to test.

The First Step Preparing the data :

We define A variable train and test to load the dataset then we split the data into features and target using .iloc so we can put the features in X and the target in y.

Read Datasets

```
In [2]: df = pd.read_csv('C:\\Users\\mohamed\\Downloads\\project\\RF_Model\\emnist-balanced-train.csv', header=None)
char_map = pd.read_csv('C:\\Users\\mohamed\\Downloads\\project\\RF_Model\\emnist-balanced-mapping.txt', delimiter=' ', header=None)

# df
char_map

Out[2]:
1
.
```

```
In [5]: X_df = df.iloc[:,1:]
y_df = df.iloc[:, 0]
print(X_df)
```

Then we wanted to rotate image and reshape it (28 * 28) and then showing it after rotate and reshape.

Rotate Images

```
In [6]: def rotate(image):
        image = image.reshape([28, 28])
        image = np.fliplr(image)
        image = np.rot90(image)
        return image

image = X_df.iloc[1258].values
print("features before reshape: "+str(image.shape)+"\n")
image = image.reshape((28, 28))
print("features after reshape: "+str(image.shape))
plt.imshow(image)
plt.show()

plt.imshow(rotate(image))
plt.show()

features before reshape: (784,)
features after reshape: (28, 28)
```

Images after rotate

```
B]: index = 3333

if y_df[index] in mapping:
    print("Target is:", mapping[y_df[index-1]])
else:
    print("Target is not found in the mapping dictionary.")

plt.imshow(X_df[index-1])

Target is: F

B]: <matplotlib.image.AxesImage at 0x20f020d66d0>
```

After that for training and testing our model and see how it performs we needed to split the data into training and testing with `train_test_split()`.

Split to Fit

```
In [11]: X_train, X_test, y_train, y_test = train_test_split(X_df, y_df, test_size= 0.15, random_state=22)
```

Second Step Building the model:

After preparing the data we used decision tree to predict and show the accuracy (60%).

Decision Tree

```
In [16]: DT_model_e = DecisionTreeClassifier(criterion="entropy",random_state=100,max_depth=35 , min_samples_split=2)
DT_model_g = DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=35 , min_samples_split=2)

In [17]: DT_model_e.fit(X_train , y_train)
DT_model_g.fit(X_train , y_train)

Out[17]:
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=35, random_state=100)

In [18]: y_pred_DT_e = DT_model_e.predict(X_test)
y_pred_DT_g = DT_model_g.predict(X_test)
```

Accuracy

```
In [19]: f1_score(y_pred_DT_e,y_test , average='weighted')
# f1_score(y_pred_DT_g,y_test , average='weighted')

Out[19]: 0.6043208335205105

In [20]: accuracy_score(y_pred_DT_e,y_test)
# accuracy_score(y_pred_DT_g,y_test)

Out[20]: 0.6048463356973995
```


Then going to second algorithm, we used random forest to predict and show the accuracy, too. The accuracy is 76%

Random Forest

```
In [21]: RF_modelg = RandomForestClassifier(n_estimators=350 , criterion='gini' , max_depth=30, min_samples_split=2 , min_samples_leaf=24,
      RF_modelle = RandomForestClassifier(n_estimators=350 , criterion='entropy' , max_depth=30, min_samples_split=2 , min_samples_leaf=24)

In [22]: RF_modelg.fit(X_train , y_train)
      RF_modelle.fit(X_train , y_train)

Out[22]:
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=30, min_samples_leaf=24,
                       n_estimators=350, n_jobs=-1)

In [23]: y_pred_RFg = RF_modelg.predict(X_test)
      y_pred_RFe = RF_modelle.predict(X_test)
```

Accuracy

```
In [24]: # f1_score(y_pred_RFg,y_test , average='weighted')
      f1_score(y_pred_RFe,y_test , average='weighted')

Out[24]: 0.7659335558417635

In [25]: # accuracy_score(y_pred_RFg,y_test)
      accuracy_score(y_pred_RFe,y_test)

Out[25]: 0.7617612293144208
```

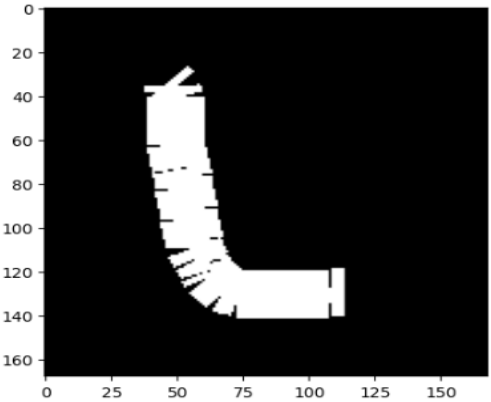
Third Step: Testing.

We used external sample by loading a image and make the model predict the answer.

Test external sample

```
In [27]: img = cv2.imread('C:\\Users\\DELL\\Downloads\\ai project\\image.png')
      plt.imshow(img)

Out[27]: <matplotlib.image.AxesImage at 0x1db4acf50>
```



```
In [28]: BW = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)
      resized_img = cv2.resize(BW , (28,28) , interpolation = cv2.INTER_AREA)
      final_img = resized_img.reshape(-1)
      y_pred_samp = RF_modelle.predict([final_img])

In [29]: if y_pred_samp[0] in mapping:
      print("Predicted " + mapping[ ( y_pred_samp[0] ) ] )
      else:
      print("Key not found in mapping")

Predicted L
```

Forth step: Save & load model:

We save the model after generating the weights and checkpoints, to use it at any time without training from scratch.

Saving The Model

```
In [42]: jb.dump(RF_model, 'C:\\Users\\DELL\\Downloads\\ai project\\final_ai.sav')
Out[42]: ['C:\\Users\\DELL\\Downloads\\ai project\\final_ai.sav']

In [214]: last = jb.load('C:\\Users\\mohamed\\Downloads\\project\\RF_Model\\final_ai_RF_model.sav')
```

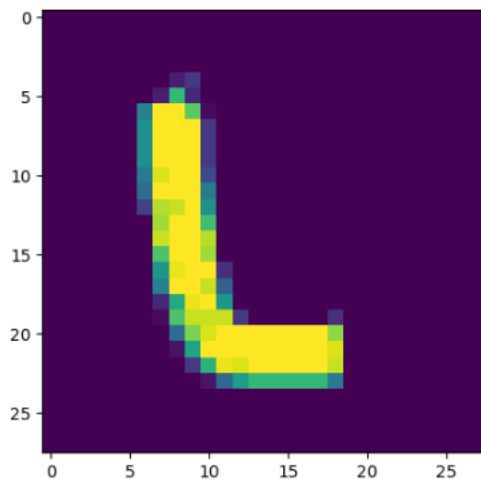
Test the saved Model

```
In [256]: predicted_key = last.predict([final_img])[0]
if predicted_key in mapping:
    print("Predicted " + mapping[predicted_key])
else:
    print("Key not found in mapping")

im = final_img
im = rotate(im)
im = rotate(im)
plt.imshow(im)
```

Predicted L

Out[256]: <matplotlib.image.AxesImage at 0x1929ca72f50>



Fifth step: Testing the model using GUI:

After that we test the model with GUI window the predict the letter.

But before that let's deep dive into the code and the function we used to run the GUI below that line.

Def Helping Functions

```
In [25]: def resize_img(cimage):
        BW = cv2.cvtColor(cimage , cv2.COLOR_BGR2GRAY)
        resized_img = cv2.resize(BW , (28,28) , interpolation = cv2.INTER_AREA)
        resized_img=resized_img.reshape(-1)
        return resized_img

In [26]: def activate_paint(e):
        global lastx, lasty
        cv.bind('<B1-Motion>', paint)
        lastx, lasty = e.x, e.y

        def paint(e):
            global lastx, lasty
            x, y = e.x, e.y
            cv.create_line((lastx, lasty, x, y), width=22,fill = 'white')
            # --- PIL
            draw.line((lastx, lasty, x, y), fill='white', width=22)
            lastx, lasty = x, y

        def clear():
            global cv,image1,draw

            image1 = PIL.Image.new('RGB', (168, 168), 'black')
            draw = ImageDraw.Draw(image1)
            cv.delete("all")

In [27]: def recoginze():
        global lb, pred_text, cv, image1, draw

        image1.save('R.png')

        # read
        img = cv2.imread('R.png')
        final = resize_img(img)
        final = final.reshape(-1)
        y_pred = model.predict([final])

        # Check if the predicted label is in the mapping dictionary
        if y_pred[0] in mapping:
            lb.config(text="Letter: " + mapping[y_pred[0]])
        else:
            lb.config(text="Letter: Not in mapping") # Handle the case when the key is not found
```

Main Window

```
In [28]: root = Tk()

        lastx, lasty = None, None
        image_number = 0

        cv = Canvas(root, width=168, height=168, bg='black')
        # --- PIL
        image1 = PIL.Image.new('RGB', (168, 168), 'black')
        draw = ImageDraw.Draw(image1)

        cv.bind('<1>', activate_paint)
        cv.pack(expand=YES, fill=BOTH)

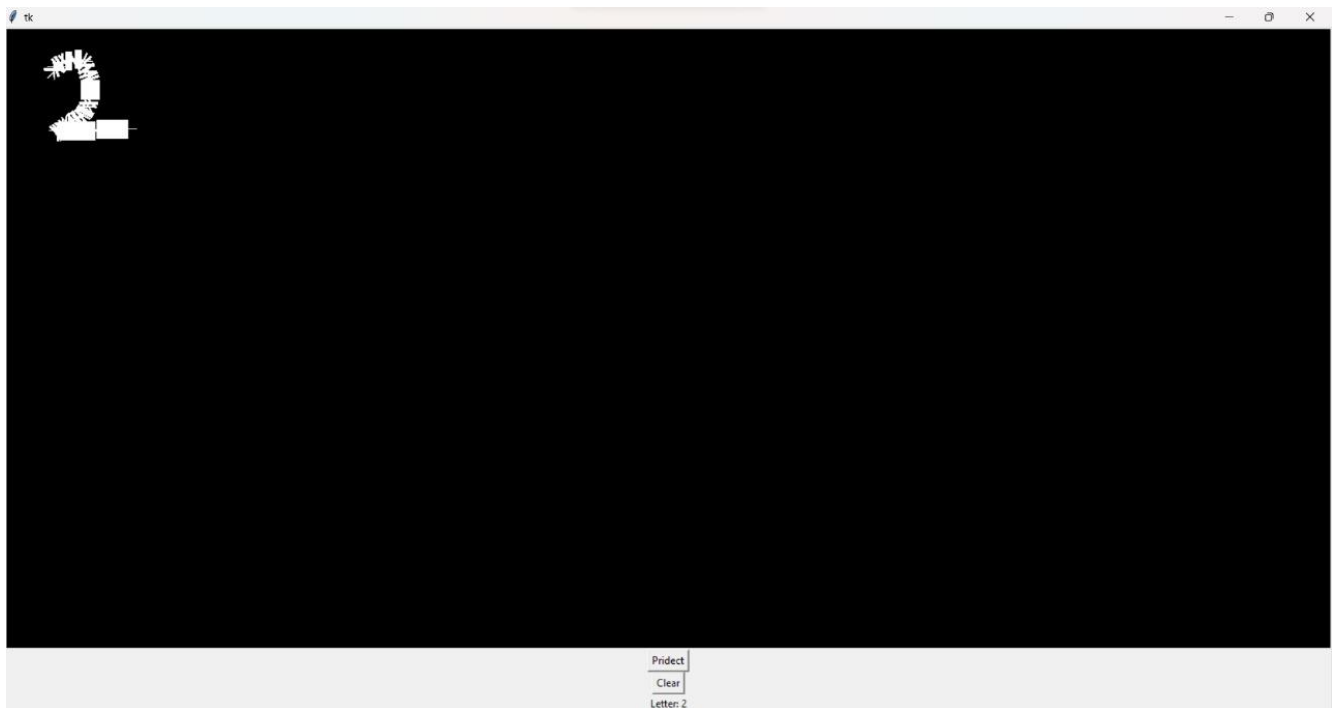
        btn_save = Button(text="Pridect", command=recoginze)
        btn_save.pack()

        btn_clr = Button(text="Clear", command=clear)
        btn_clr.pack()

        lb = Label(text = "Letter: ")
        lb.pack()

        root.mainloop()
```

And there the GUI window and the prediction.



Development platforms:

We used google Kaggle and Jupyter notebook to write and execute arbitrary python code.

- **Numpy:** is used to convert the image into an array and to deal with it.
- **Sklearn:** we used it for splitting the dataset..
- **Matplotlib:** Matplotlib is a comprehensive library for creating static and interactive visualizations in Python to virtualize data.

-
- **Pandas:** we used this library to read the csv file.
 - **Jolib:** is a Python library that provides tools for parallel processing and efficient data caching.

```
In [5]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, f1_score, accuracy_score
import cv2
import joblib as jb

# pd.set_option('display.max_rows', 5000)
# pd.set_option('display.max_columns', 5000)
# pd.set_option('display.width', 10000)
# warnings.filterwarnings('ignore')
```