DIGIT RECOGNITION TOOL

P SAIKARTHIK

VU21CSEN0100398

1. Abstract

Handwritten digit classification aims to correctly identify digits from images, despite the differences in writing styles. The complexity arises from factors such as variations in slant, thickness, and curvature of handwritten digits.

2. Introduction

Image classification is a critical task in computer vision, where the goal is to categorize images into predefined classes. In this project, a Random Forest Classifier, an ensemble learning method, is used to perform image classification. The dataset consists of grayscale images, each labeled with a class.

3. Methodology

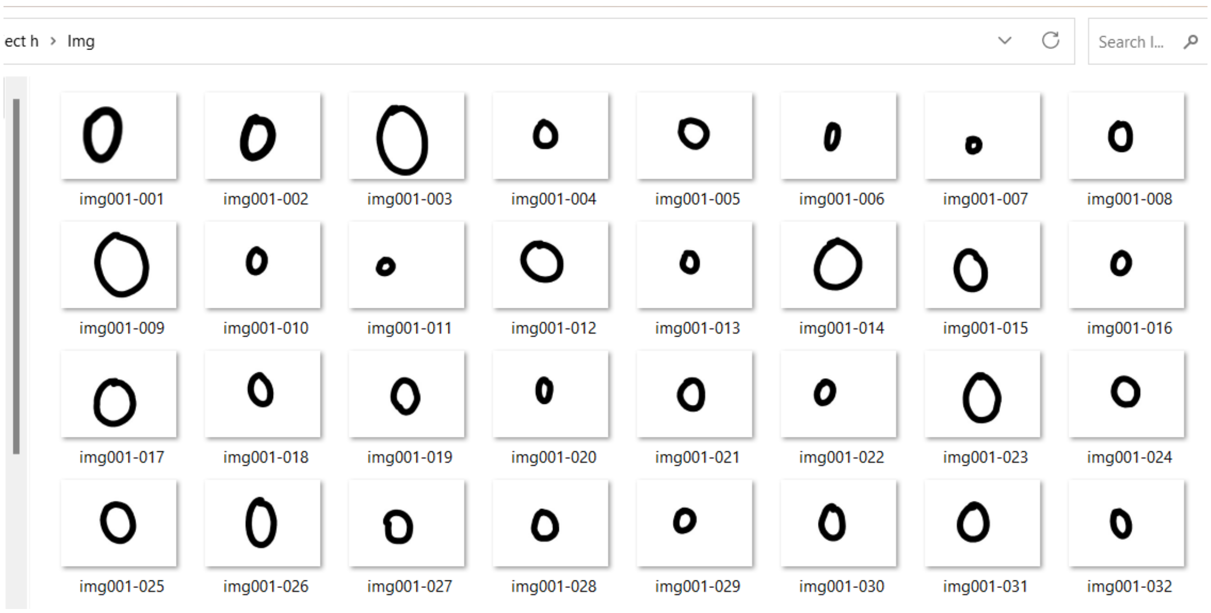
Data Preprocessing: The images are first read in grayscale mode using OpenCV, resized to 200x200 pixels, normalized by dividing by 255, and flattened into a one-dimensional array.

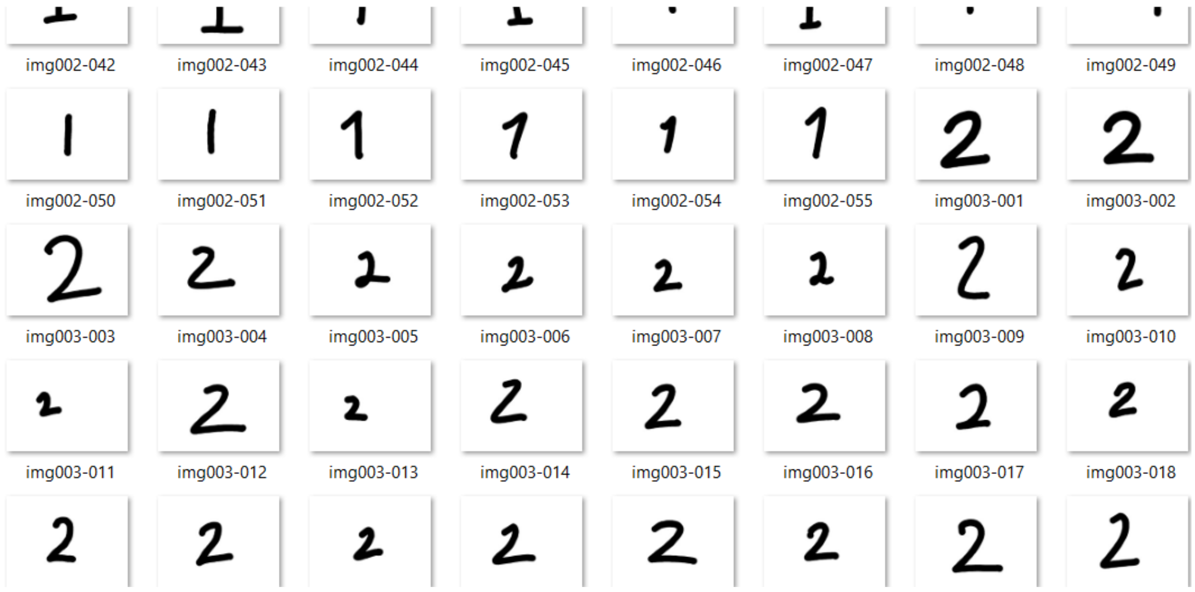
Label Encoding: The image labels are encoded using LabelEncoder to transform categorical labels into numeric form.

Train-Test Split: The dataset is split into training and testing sets using an 80-20 split.

Model Training: A RandomForestClassifier with 500 estimators and a random state of 40 is trained on the training data.

Prediction: The model is used to predict the labels of 10 test images, which are then displayed alongside their predicted labels.



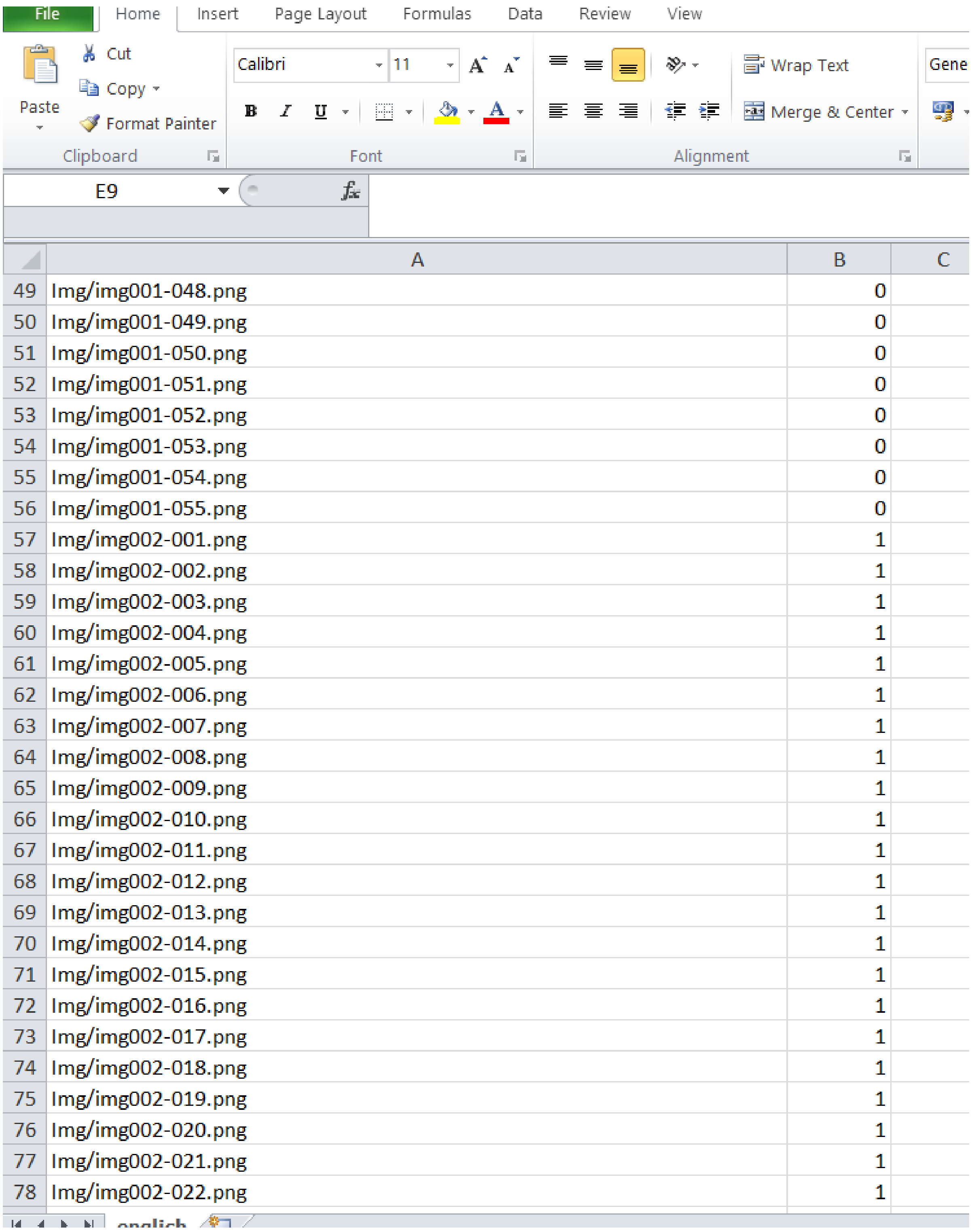


4. Code Explanation

Data Loading: The code begins by reading the dataset english.csv and initializing an empty list l to store image data and labels.

Image Processing Function (fun): This function reads, resizes, normalizes, and flattens each image, appending the processed image and its label to list l.

Feature and Label Extraction: The processed images (x) and labels (y) are extracted and prepared for model training.



	A	B	C
49	Img/img001-048.png	0	
50	Img/img001-049.png	0	
51	Img/img001-050.png	0	
52	Img/img001-051.png	0	
53	Img/img001-052.png	0	
54	Img/img001-053.png	0	
55	Img/img001-054.png	0	
56	Img/img001-055.png	0	
57	Img/img002-001.png	1	
58	Img/img002-002.png	1	
59	Img/img002-003.png	1	
60	Img/img002-004.png	1	
61	Img/img002-005.png	1	
62	Img/img002-006.png	1	
63	Img/img002-007.png	1	
64	Img/img002-008.png	1	
65	Img/img002-009.png	1	
66	Img/img002-010.png	1	
67	Img/img002-011.png	1	
68	Img/img002-012.png	1	
69	Img/img002-013.png	1	
70	Img/img002-014.png	1	
71	Img/img002-015.png	1	
72	Img/img002-016.png	1	
73	Img/img002-017.png	1	
74	Img/img002-018.png	1	
75	Img/img002-019.png	1	
76	Img/img002-020.png	1	
77	Img/img002-021.png	1	
78	Img/img002-022.png	1	

Model Training: The RandomForestClassifier is trained on the training set and then used to predict the labels of test images.

Visualization: The predicted labels are printed, and corresponding images are displayed using Matplotlib.

IMPLEMENTATION

```
[31]: import pandas as pd
      import cv2
      import numpy as np
      from sklearn.preprocessing import LabelEncoder
      from sklearn.model_selection import train_test_split
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.metrics import accuracy_score
      import matplotlib.pyplot as plt
```

```
•[2]: df = pd.read_csv('english.csv')
      df.head(1)
```

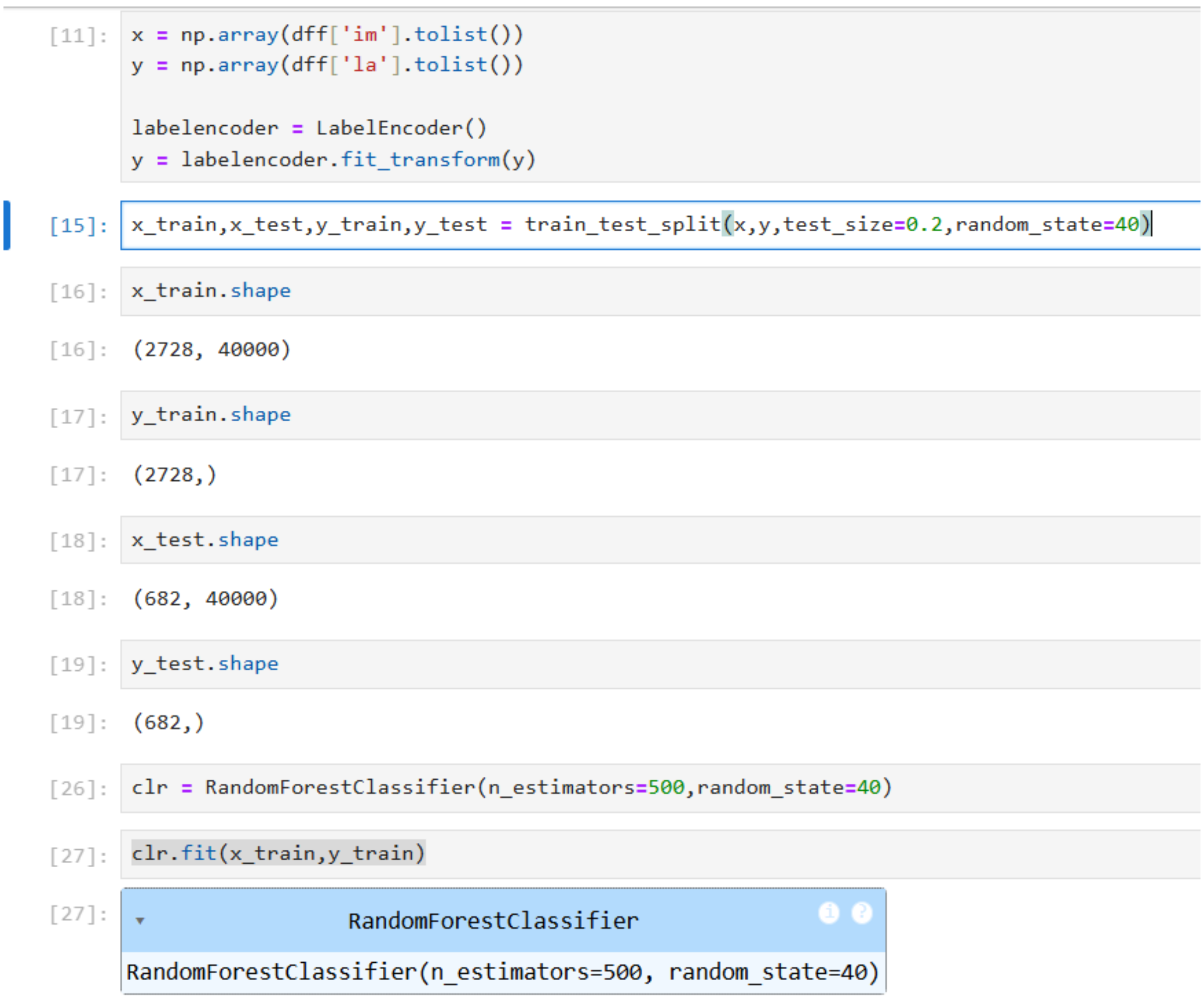
```
[2]:                  image   label
     0   Img/img001-001.png       0
```

```
[4]: l=[]
     def fun(file,lab):
         img = cv2.imread(file,cv2.IMREAD_GRAYSCALE)
         img = cv2.resize(img,(200,200))
         img = img/255
         img = img.flatten()
         l.append([img,lab])
```

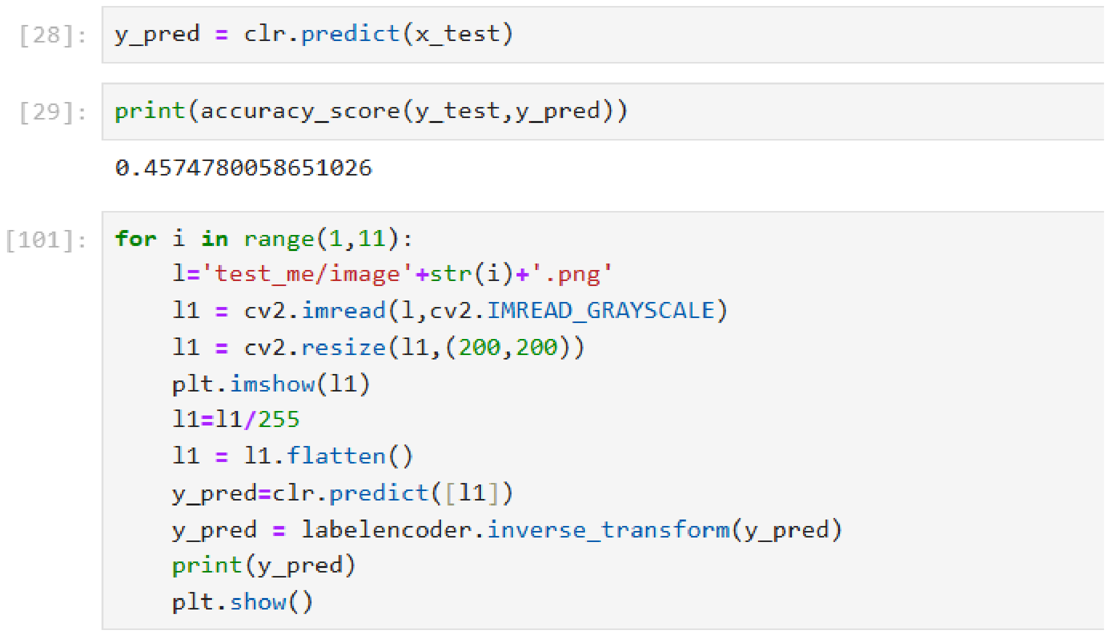
```
[5]: for i in range(len(df)):
         fun(df['image'][i],df['label'][i])
```

```
[6]: dff = pd.DataFrame(l,columns=['im','la'])
```

```
[11]: x = np.array(dff['im'].tolist())
```



```
[11]: x = np.array(dff['im'].tolist())
      y = np.array(dff['la'].tolist())

      labelencoder = LabelEncoder()
      y = labelencoder.fit_transform(y)

[15]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)

[16]: x_train.shape

[16]: (2728, 40000)

[17]: y_train.shape

[17]: (2728,)

[18]: x_test.shape

[18]: (682, 40000)

[19]: y_test.shape

[19]: (682,)

[26]: clr = RandomForestClassifier(n_estimators=500,random_state=40)

[27]: clr.fit(x_train,y_train)

[27]:                RandomForestClassifier
      RandomForestClassifier(n_estimators=500, random_state=40)
```

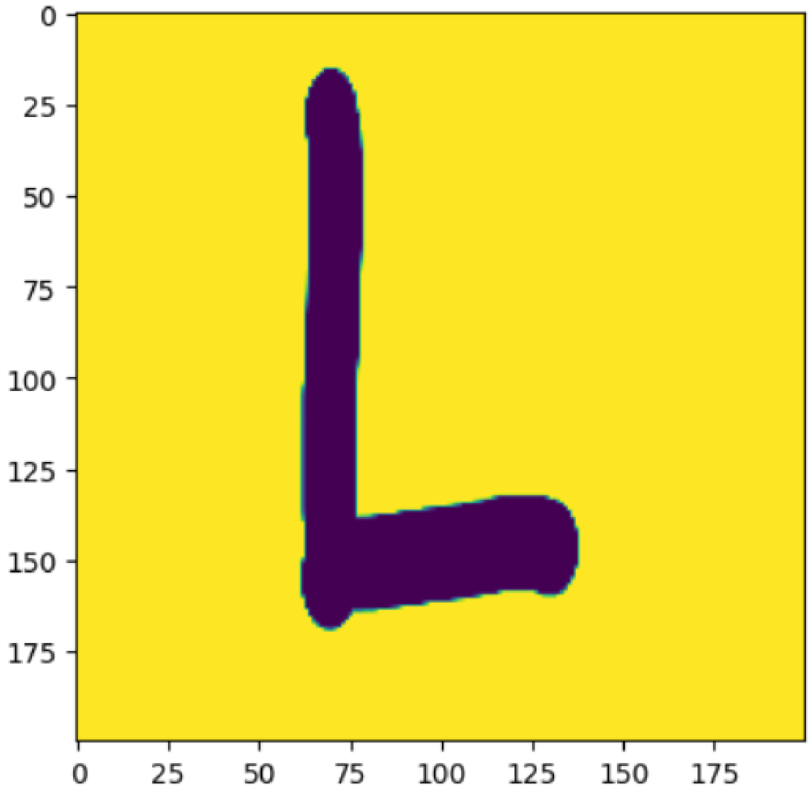


```
[28]: y_pred = clr.predict(x_test)

[29]: print(accuracy_score(y_test,y_pred))

      0.4574780058651026

[101]: for i in range(1,11):
           l='test_me/image'+str(i)+'.png'
           l1 = cv2.imread(l,cv2.IMREAD_GRAYSCALE)
           l1 = cv2.resize(l1,(200,200))
           plt.imshow(l1)
           l1=l1/255
           l1 = l1.flatten()
           y_pred=clr.predict([l1])
           y_pred = labelencoder.inverse_transform(y_pred)
           print(y_pred)
           plt.show()
```

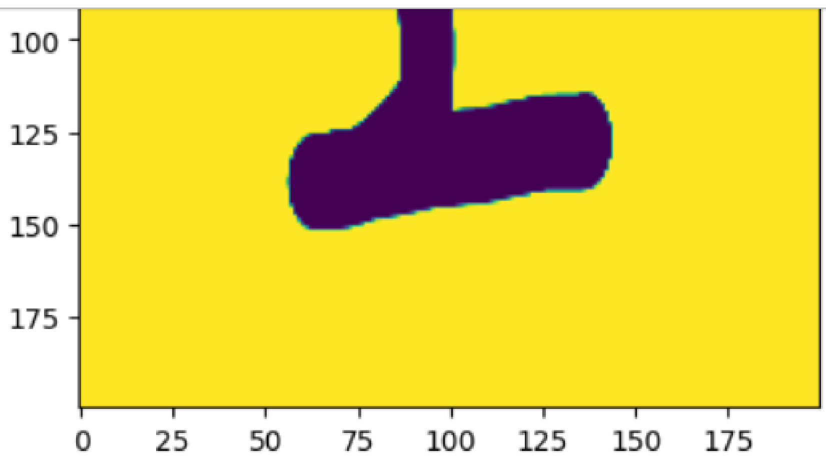
5. Results

The Random Forest model successfully predicted the labels for the test images. The results were displayed for visual confirmation.

```
['L']
```

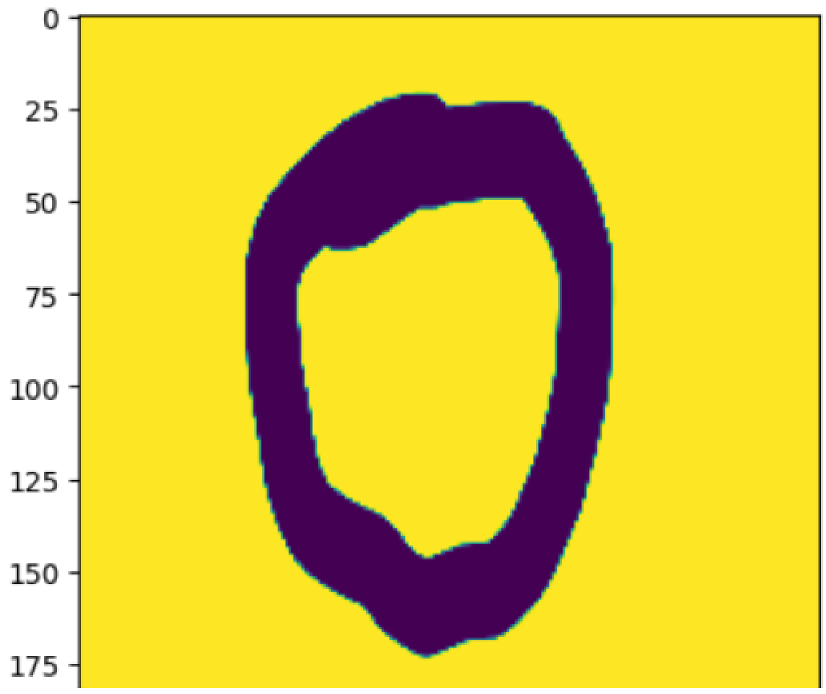


```
['2']
```

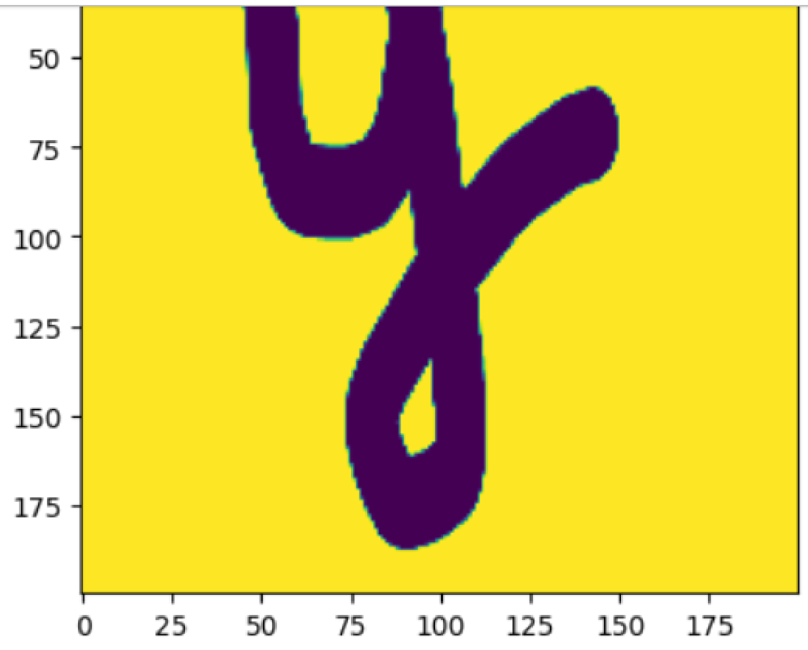


100
125
150
175
0
25
50
75
100
125
150
175

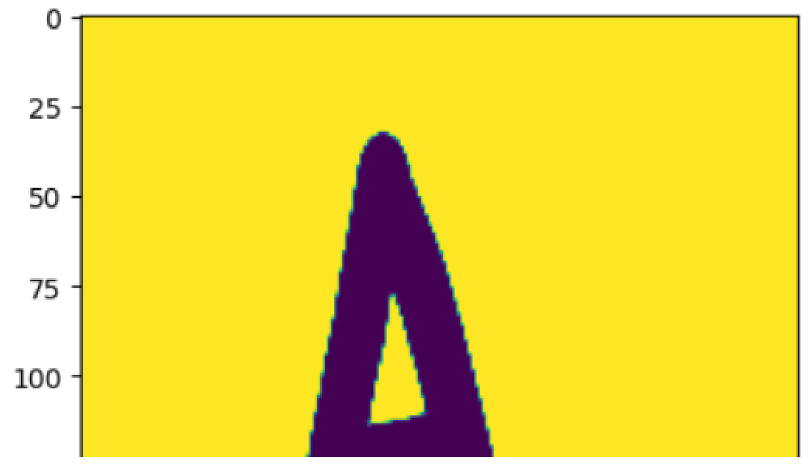
```
['ㅇ']
```



0
25
50
75
100
125
150
175



```
['A']
```



6. Discussion

The model demonstrates a straightforward application of Random Forests for image classification. The preprocessing steps, such as resizing and normalization, are crucial for ensuring that the model performs well. Random Forest, being an ensemble method, helps in achieving stable and accurate predictions by aggregating the results of multiple decision trees.

7. Conclusion

The implemented Random Forest model effectively classifies grayscale images, demonstrating its capability in image recognition tasks. Future improvements could include experimenting with different image sizes, enhancing preprocessing techniques, and testing other classifiers for better performance.

8. References

Kaggle: Image classification using Sklearn (RandomForest) [Kaggle].

GeeksforGeeks: Random Forest Classifier using Scikit-learn [GeeksforGeeks].

🌐 Sources

kaggle.com - Image classification using Sklearn (RandomForest)

geeksforgeeks.org - Random Forest Classifier using Scikit-learn