

Digit Recognizer

การจดจำเลขดิจิตัลที่เขียนด้วยมือ

จัดทำโดย

บพด. เกียรติศรี 6404062610286
พชร. เต็มใจ 6404062610138

ที่มาและความสำคัญของโครงงาน

มีความสำคัญเนื่องจากเป็นโครงการเริ่มต้นที่เหมาะสมสำหรับผู้ที่สนใจการเรียนรู้เกี่ยวกับการเรียนรู้ของ Machine โดยชุดข้อมูล MNIST เป็นชุดข้อมูลขนาดเล็กและเข้าใจง่าย ทำให้ผู้เริ่มต้นสามารถเรียนรู้เกี่ยวกับขั้นตอนต่างๆ ในการจำแนกภาพอย่างมืออาชีพได้

ประโยชน์ของโครงการและการนำไปใช้

สามารถเรียนรู้เกี่ยวกับขั้นตอนต่างๆ
ในการจำแนกภาระมือเขียนตัวเลขได้ด้วยความถูกต้อง
และแม่นยำ

ข้อมูลที่นำมาใช้ทำโครงสร้าง



1

ข้อมูลจาก kaggle

<https://www.kaggle.com/competitions/digit-recognizer>

The screenshot shows the Kaggle interface for the 'Digit Recognizer' competition. At the top, there's a navigation bar with 'Overview', 'Data' (which is currently selected), 'Code', 'Models', 'Discussion', 'Leaderboard', 'Rules', 'Team', 'Submissions', 'Submit Predictions', and a '...' button. Below the navigation bar, there's a large preview area showing a grid of handwritten digits from the MNIST dataset. To the left of the preview, there's some descriptive text: 'Getting Started Prediction Competition', 'Learn computer vision fundamentals with the famous MNIST data', and 'Kaggle 1,480 teams Ongoing'. The bottom part of the screenshot shows a table with the first few rows of the 'train.csv' file.

#	label	# pixel0
1	0	0
0	0	0
1	0	0
4	0	0

2

Attributes នូវ features

- 1.label
- 2.pixel

[train.csv \(76.78 MB\)](#)

	Detail	Compact	Column
#	label	# pixel0	
1	0	0	
0	0	0	
1	0	0	
4	0	0	

4

ข้อมูลที่นำมาใช้ทำโครงงาน



3

เอาท์พุตหรือลາเบลที่ต้องการ

1. ImageId (1,2,3...)
2. Label (0-9)

[sample_submission.csv \(240.91 kB\)](#)

Detail	Compact	Column
ImageId	# Label	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	

Platform ที่ใช้ในการทำโครงสร้าง



Kaggle Kernel

Model Digit reg.

File Edit View Run Add-ons Help

+ X D Run All Code Draft Session off (run a cell to start) ...

```
import pandas as pd
import numpy as np
data=pd.read_csv("/kaggle/input/digit-recognizer/train.csv")
X=data.drop(columns='label')
Y=data['label']
test_data=pd.read_csv("/kaggle/input/digit-recognizer/test.csv")
test_data
```

[33]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777
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
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
3	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
...
27995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
27996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
27997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
27998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
27999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0

28000 rows × 784 columns

Getting Started Prediction Competition

Digit Recognizer

Learn computer vision fundamentals with the famous MNIST data

Kaggle 1,481 teams Ongoing

Overview Data Code Models Discussion Leaderboard Rules Team Submissions Submit Predictions ...

Dataset Description

The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as $x = i * 28 + j$, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28×28 matrix, (indexing by zero).

For example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.

Visually, if we omit the "pixel" prefix, the pixels make up the image like this:

Files
3 files

Size
128.13 MB

Type
CSV

License
CC BY-SA 3.0

ผลลัพธ์ที่คาดหวัง

สามารถที่จะคำนยตัวเลขที่เขียน
ด้วยมือได้อย่างแม่นยำและ
ถูกต้อง

ขั้นตอนการดำเนินงาน

1

1. ศึกษา data
input output
ที่ต้องการ

2

2. เปรียบเทียบ
model ที่จะใช้

3

3. เลือก model
ที่ต้องการใช้

4

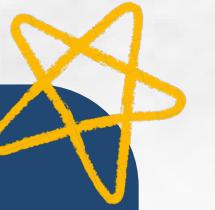
4. Train model

5

5. ประเมิน model
และปรับปรุง
parameter

1

ศึกษา data input output ที่ต้องการ



ศึกษา data input output ที่ต้องการ

ข้อมูลที่ใช้ในการ train model
เป็นไฟล์ภาพที่แปลงเป็น csv
ซึ่งบอก ค่าสีในแต่ละ pixel (0-255)
และ label ที่เป็นบวกว่าเป็นตัวเลขใด

Input

- digit-recognizer
 - sample_submission.csv
 - test.csv
 - train.csv

train.csv (76.78 MB)

Detail Compact Column

# label	# pixel0	# pixel1
0	9	0
3	0	0
8	0	0
9	0	0
1	0	0
3	0	0
3	0	0
1	0	0

ศึกษา data input output ที่ต้องการ

Output ที่ต้องการเป็นไฟล์ csv ที่
ประกอบไปด้วย ImageID และ Label
ดังนั้นเมื่อสร้าง model เสร็จแล้วต้อง
สร้างไฟล์ csv ให้เป็นอย่างที่ Kaggle
ต้องการ

[sample_submission.csv \(240.91 kB\)](#)

[Detail](#) [Compact](#) [Column](#)

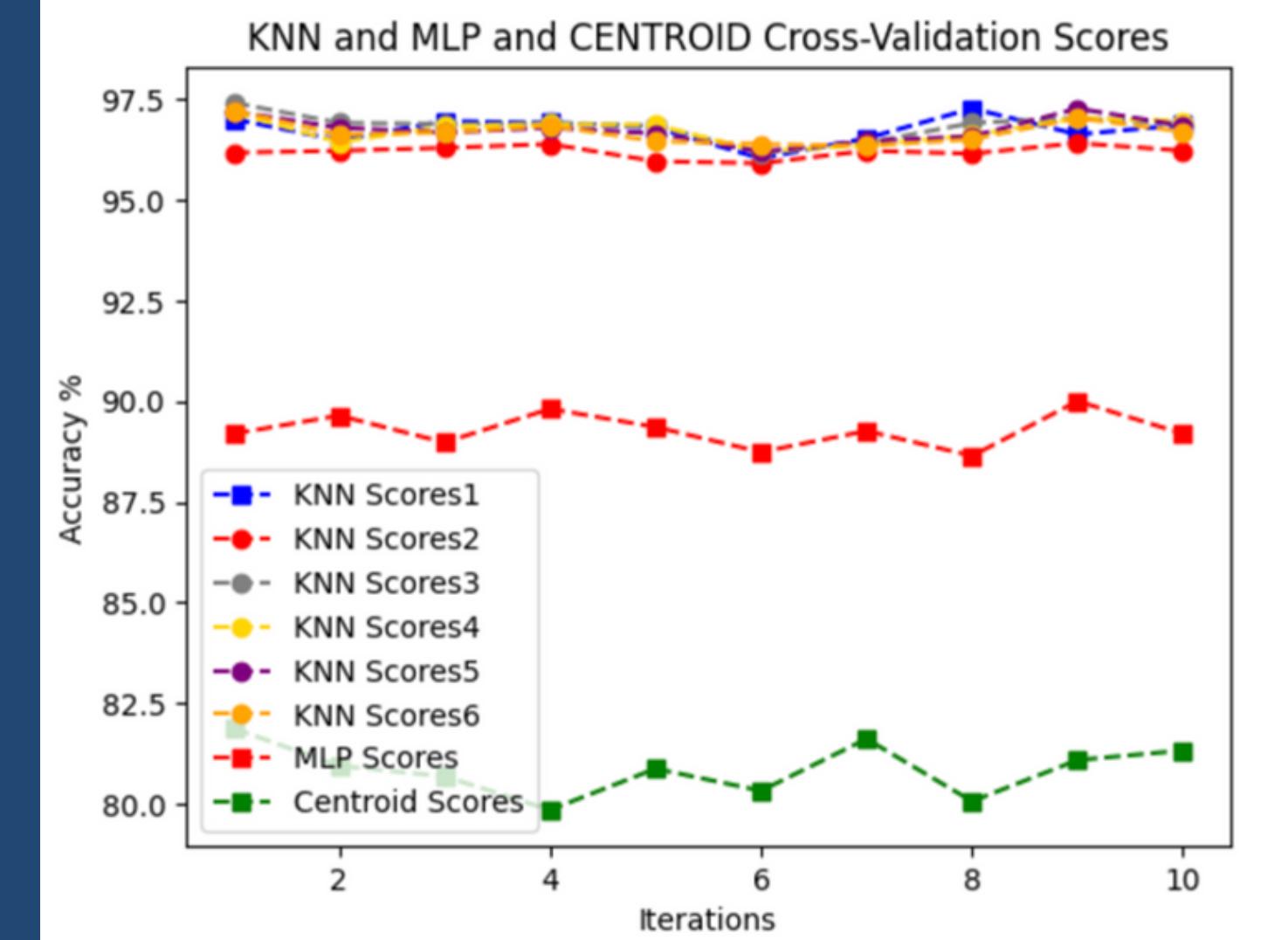
ImageId	# Label
1	28.0k
1	0
2	0
3	0
4	0
5	0

2

เปรียบเทียบ model ที่จะใช้

เปรียบเทียบ model ที่จะใช้

ความแม่นยำ MLP_Model: 89.28095238095237 %
ความแม่นยำ KNN_Model1: 96.74047619047619 %
ความแม่นยำ KNN_Model2: 96.18809523809524 %
ความแม่นยำ KNN_Model3: 96.8309523809524 %
ความแม่นยำ KNN_Model4: 96.72380952380955 %
ความแม่นยำ KNN_Model5: 96.73809523809526 %
ความแม่นยำ KNN_Model6: 96.67142857142856 %
ความแม่นยำ Centroid_Model: 80.87142857142857 %



3

เลือก model ที่ต้องการใช้

เลือก model ที่ต้องการใช้

จากที่เห็นข้อสรุปจาก ข้อ 2 จะเห็นได้ว่า KNN
มีความแม่นยำค่อนข้างสูง และ เป็นอัลกอริทึมที่เหมาะสมสำหรับ¹
งานการจำแนกที่เข้าใจง่ายและเรียนรู้จากข้อมูลโดยตรง โดยไม่
ต้องการการปรับค่าหลาย ๆ ตัวแปร



4

Train model

Train model

เริ่มจากขั้นตอนแรก
import data ทั้ง **train** และ **test**
จากนั้นทำการแบ่งข้อมูล **train**
เป็น **X**, **Y**

```
from sklearn.neighbors import KNeighborsClassifier  
model = KNeighborsClassifier(n_neighbors=5)  
model.fit(X, Y)
```

```
import pandas as pd  
import numpy as np  
data=pd.read_csv("/kaggle/input/digit-recognizer/train.csv")  
X=data.drop(columns='label')  
Y=data['label']  
test_data=pd.read_csv("/kaggle/input/digit-recognizer/test.csv")  
test_data
```

สร้างและฝึกโมเดล KNN
ด้วยเพื่อนบ้าน = 5
โดยที่ให้ **X** เป็นข้อมูลฝึก
และ **Y** เป็นค่าเป้าหมาย

Train model

```
label = model.predict(test_data)
import matplotlib.pyplot as plt, matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
from sklearn import svm
%matplotlib inline

for i in range(len(test_data)):      #length of test_data
    plt.title(label[i])
    img = test_data.iloc[i].values
    img = img.reshape((28, 28))
    plt.imshow(img, cmap='gray')
    plt.show()
```

ตัวอย่างการแสดงผล
การคำนایของ model



5

ประเมณ model และ ปรับ parameter

ปรับแต่ง model และ ปรับ parameter



```
from sklearn.model_selection import cross_val_score
model_knn = KNeighborsClassifier(n_neighbors=5)
knn_scores = cross_val_score(model_knn, X, Y, cv=10, scoring='accuracy')
print("KNN scores:", knn_scores)
```

```
KNN scores: [0.97190476 0.96785714 0.96666667 0.96785714 0.96642857 0.96190476
 0.96452381 0.96571429 0.97261905 0.96833333]
```

ใช้ crossvalidation
โดยกำหนดให้ $cv=10$

```
print("Accuracy of KNN_Model:", (sum(knn_scores)/len(knn_scores))*100, "%")
```

```
Accuracy of KNN_Model: 96.73809523809526 %
```

จะเห็นได้ว่าความแม่นยำ
= 96.7 %



ประเมิน model และ ปรับ parameter

```
print("Accuracy of KNN_Model 5:",(sum(knn_scores5)/len(knn_scores5))*100,"%")
print("Accuracy of KNN_Model 4:",(sum(knn_scores4)/len(knn_scores4))*100,"%")
print("Accuracy of KNN_Model 3:",(sum(knn_scores3)/len(knn_scores3))*100,"%")
print("Accuracy of KNN_Model 2:",(sum(knn_scores2)/len(knn_scores2))*100,"%")
print("Accuracy of KNN_Model 1:",(sum(knn_scores1)/len(knn_scores1))*100,"%")
print("Accuracy of KNN_Model 10:",(sum(knn_scores10)/len(knn_scores10))*100,"%")
```



```
Accuracy of KNN_Model 5: 96.73809523809526 %
Accuracy of KNN_Model 4: 96.72380952380955 %
Accuracy of KNN_Model 3: 96.8309523809524 %
Accuracy of KNN_Model 2: 96.18809523809524 %
Accuracy of KNN_Model 1: 96.74047619047619 %
Accuracy of KNN_Model 10: 96.44285714285714 %
```

จากการปรับ
parameter
จะเห็นได้ว่า^{ใช้ K = 3}
มีความแม่นยำสูง
ที่สุด

Submission

Digit Recognizer
Learn computer vision fundamentals with the famous MNIST data

Kaggle 1,489 teams Ongoing

Overview Data Code Models Discussion Leaderboard Rules Team Submissions Submit Predictions ...

Submissions

All Successful Errors

Recent ▾

Submission and Description

Public Score ⓘ

 submission_version_2.csv 0.96803

Complete · now

 submission_version_1.csv 0.967

Complete · 3h ago

 data2.csv 0.967

Complete · 1d ago

เริ่มแรกทำการหาโมเดลที่ดีที่สุดโดยการทำ K-cross validation
และนำค่า Accuracy มาเปรียบเทียบแล้ว
ได้ Model ที่ดีที่สุดคือ K-NN โดยตอนแรกใช้ค่า $K=5$ ก่อนในตอน
แรก แล้วมาปรับเป็น $K=3$ เพราะว่ามีความแม่นยำมากกว่า