k_nearest_neighbors

August 19, 2022

1 K-Nearest Neighbors

You should build a machine learning pipeline using a k-nearest neighbor model. In particular, you should do the following: - Load the `mnist` dataset using Pandas. You can find this dataset in the datasets folder. - Split the dataset into training and test sets using Scikit-Learn. - Train and test a k-nearest neighbor model using Scikit-Learn. - Check the documentation to identify the most important hyperparameters, attributes, and methods of the model. Use them in practice.

1.1 importing libraries

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.model_selection
import sklearn.metrics
```

1.2 loading datasets

[4]:
```
df = pd.read_csv("../../datasets/mnist.csv")
df.head()
```

[4]:

	id	class	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	\
0	31953	5	0	0	0	0	0	0	0	
1	34452	8	0	0	0	0	0	0	0	
2	60897	5	0	0	0	0	0	0	0	
3	36953	0	0	0	0	0	0	0	0	
4	1981	3	0	0	0	0	0	0	0	

	pixel8	…	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	\
0	0	…	0	0	0	0	0	0	
1	0	…	0	0	0	0	0	0	
2	0	…	0	0	0	0	0	0	
3	0	…	0	0	0	0	0	0	
4	0	…	0	0	0	0	0	0	

```
   pixel781  pixel782  pixel783  pixel784
0         0         0         0         0
1         0         0         0         0
2         0         0         0         0
3         0         0         0         0
4         0         0         0         0

[5 rows x 786 columns]
```

[5]:
```
df = df.set_index("id")
```

[6]:
```
df.head()
```

[6]:
```
        class  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  \
id
31953       5       0       0       0       0       0       0       0       0
34452       8       0       0       0       0       0       0       0       0
60897       5       0       0       0       0       0       0       0       0
36953       0       0       0       0       0       0       0       0       0
1981        3       0       0       0       0       0       0       0       0

        pixel9  …  pixel775  pixel776  pixel777  pixel778  pixel779  \
id              …
31953        0  …          0         0         0         0         0
34452        0  …          0         0         0         0         0
60897        0  …          0         0         0         0         0
36953        0  …          0         0         0         0         0
1981         0  …          0         0         0         0         0

        pixel780  pixel781  pixel782  pixel783  pixel784
id
31953          0         0         0         0         0
34452          0         0         0         0         0
60897          0         0         0         0         0
36953          0         0         0         0         0
1981           0         0         0         0         0

[5 rows x 785 columns]
```

1.3 Split Data into Train and Test datasets

[7]:
```
X = df.drop(["class"], axis = 1)
y = df["class"]
```

[8]:
```
print("shape of X = ", X.shape)
print("shape of y = ", y.shape)
```

```
shape of X =  (4000, 784)
shape of y =  (4000,)
```

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,␣
 ↪random_state = 51)
print("shape of X_train = ", X_train.shape)
print("shape of y_train = ", y_train.shape)
print("shape of X_test = ", X_test.shape)
print("shape of X_test = ", y_test.shape)
```

```
shape of X_train =  (3200, 784)
shape of y_train =  (3200,)
shape of X_test =  (800, 784)
shape of X_test =  (800,)
```

[32]:
```
X_train
```

[32]:

id	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	\
42005	0	0	0	0	0	0	0	0	0	
27905	0	0	0	0	0	0	0	0	0	
68331	0	0	0	0	0	0	0	0	0	
61932	0	0	0	0	0	0	0	0	0	
67308	0	0	0	0	0	0	0	0	0	
…	…	…	…	…	…	…	…	…	…	
66843	0	0	0	0	0	0	0	0	0	
36867	0	0	0	0	0	0	0	0	0	
36277	0	0	0	0	0	0	0	0	0	
22120	0	0	0	0	0	0	0	0	0	
66781	0	0	0	0	0	0	0	0	0	

id	pixel10	…	pixel775	pixel776	pixel777	pixel778	pixel779	\
42005	0	…	0	0	0	0	0	
27905	0	…	0	0	0	0	0	
68331	0	…	0	0	0	0	0	
61932	0	…	0	0	0	0	0	
67308	0	…	0	0	0	0	0	
…	…	…	…	…	…	…	…	
66843	0	…	0	0	0	0	0	
36867	0	…	0	0	0	0	0	
36277	0	…	0	0	0	0	0	
22120	0	…	0	0	0	0	0	
66781	0	…	0	0	0	0	0	

```
       pixel780  pixel781  pixel782  pixel783  pixel784
id
42005         0         0         0         0         0
27905         0         0         0         0         0
68331         0         0         0         0         0
61932         0         0         0         0         0
67308         0         0         0         0         0
…           …         …         …         …         …
66843         0         0         0         0         0
36867         0         0         0         0         0
36277         0         0         0         0         0
22120         0         0         0         0         0
66781         0         0         0         0         0

[3200 rows x 784 columns]
```

[97]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[98]:
```
model=KNeighborsClassifier(n_neighbors=5,metric='cosine')
```

[99]:
```
model.fit(X,y)
```

[99]:
```
KNeighborsClassifier(metric='cosine')
```

[100]:
```
y_predicted=model.predict(X_test)
```

[101]:
```
print(y_predicted)
```

```
[9 6 6 9 3 7 8 0 4 2 3 0 4 0 0 1 1 3 6 0 1 6 2 9 8 4 9 9 4 4 8 0 6 6 8 9 4
 4 5 7 6 9 0 7 6 0 3 1 7 9 0 8 7 0 3 9 8 3 4 3 9 3 4 2 7 8 5 4 1 6 2 1 5 4
 3 5 2 6 8 2 3 3 0 8 3 7 8 0 6 0 5 9 1 1 1 0 3 8 6 0 9 8 7 1 1 8 9 5 7 5 1
 3 3 0 1 8 9 3 6 5 5 7 1 4 6 4 6 1 1 1 4 9 7 8 0 0 6 4 1 8 9 2 5 7 1 2 6 0
 1 5 4 3 6 0 9 8 2 3 8 0 8 7 1 6 0 4 9 8 3 6 9 0 7 8 6 8 1 9 0 8 3 5 5 0 7
 2 8 4 9 4 7 6 4 2 7 1 7 3 3 1 6 8 7 8 6 8 6 0 5 8 5 7 6 0 6 4 4 6 3 1 4 8
 7 0 9 7 2 6 9 8 9 6 1 6 9 5 3 3 7 7 9 5 2 1 4 0 4 1 9 9 4 3 1 5 5 1 4 0 5
 0 4 0 1 2 1 9 5 4 5 8 1 9 7 1 3 6 4 2 7 2 3 3 7 8 3 3 4 3 9 2 1 9 8 6 0 7
 7 3 2 7 3 3 3 1 6 4 2 6 8 4 8 0 4 1 8 1 6 1 4 2 5 6 1 2 5 1 0 6 4 3 2 9 9
 1 2 8 7 8 1 3 5 0 0 4 6 8 3 3 2 2 9 4 4 1 3 2 8 0 0 6 8 6 6 9 9 9 3 4 6 5
 6 9 3 5 1 5 0 7 4 0 7 8 8 3 0 4 3 5 7 5 8 3 6 5 2 5 1 0 6 3 5 1 5 0 0 2 6
 8 5 3 3 7 7 0 4 0 0 8 3 9 8 1 8 6 1 3 1 1 7 1 4 3 5 6 1 7 1 8 3 8 8 9 3 0
 1 7 6 2 0 8 7 2 0 3 8 7 1 5 6 5 9 0 1 3 2 7 6 2 8 3 0 1 2 1 6 5 2 0 6 6 5
 0 0 9 6 2 7 5 4 1 1 9 5 6 2 3 8 1 6 9 3 5 1 0 5 0 5 8 8 0 5 3 7 0 6 1 0 1
 8 4 5 6 5 1 9 8 6 2 3 0 0 3 0 6 9 5 3 1 2 1 0 0 1 3 6 7 3 6 6 4 7 0 6 5 2
 9 8 6 5 9 4 1 1 4 3 2 8 8 1 9 3 1 3 0 3 1 1 2 3 7 1 6 8 8 2 4 8 1 8 5 7 7
 0 4 2 7 6 0 2 6 9 2 8 8 5 3 3 8 7 1 4 7 6 6 8 3 4 7 1 1 2 4 9 3 3 9 5 9 7
 1 6 7 4 8 1 0 6 4 9 1 2 5 1 9 1 2 9 1 6 3 9 1 6 7 3 6 8 0 0 8 1 0 2 0 9 8
 3 3 1 7 6 5 1 1 9 1 4 7 1 1 4 1 7 1 7 0 4 2 1 3 5 4 2 7 0 5 9 9 8 7 7 1 3
```

```
 7 5 1 2 3 1 3 4 7 9 4 6 2 0 0 1 1 2 3 0 8 3 0 3 7 3 5 3 0 5 0 3 0 3 8 6 2
 8 5 1 2 4 2 7 0 2 8 8 1 8 4 5 4 9 8 3 1 7 9 2 9 7 8 5 4 8 9 6 3 8 0 7 1 7
 8 1 0 0 6 1 1 3 1 0 5 3 3 7 9 1 4 8 8 8 0 4 2]
```

[102]:
```
accuracy = sklearn.metrics.accuracy_score(y_test,y_predicted)
```

[103]:
```
accuracy
```

[103]: 0.9675

1.4 Observations

- As we saw this data set is a classifier .I have use KNeighborsClassifier
- in the data set the ”ID” was useless so i put it as index (i can either drop it
- In this model shows 96% Accuracy with hyperparametre(cosine) rathen then others.
- others are below 96% accuracy showing

[ ]:
```

```