umap_largevis_done

April 1, 2023

0.1 Filip Ręka, Daniel Kuc, Kamil Kwarciak

[ ]:
```
!pip install umap-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: umap-learn in /usr/local/lib/python3.9/dist-
packages (0.5.3)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.9/dist-
packages (from umap-learn) (1.10.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-
packages (from umap-learn) (1.22.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages
(from umap-learn) (4.65.0)
Requirement already satisfied: pynndescent>=0.5 in
/usr/local/lib/python3.9/dist-packages (from umap-learn) (0.5.8)
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.9/dist-
packages (from umap-learn) (0.56.4)
Requirement already satisfied: scikit-learn>=0.22 in
/usr/local/lib/python3.9/dist-packages (from umap-learn) (1.2.2)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in
/usr/local/lib/python3.9/dist-packages (from numba>=0.49->umap-learn) (0.39.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-
packages (from numba>=0.49->umap-learn) (67.6.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.9/dist-
packages (from pynndescent>=0.5->umap-learn) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.22->umap-learn)
(3.1.0)
```

[ ]:
```
import umap.umap_ as umap
import sklearn.datasets
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

0.1.1 t-SNE, UMAP and LargeVis

In this and the next notebook we will use manifold learning for data visualization of large data sets (with high dimensionality). In addition to t-SNE, two relatively new methods will be used that are more efficient on large data sets.

- UMAP (Uniform Manifold Approximation and Projection) - Install this Python package: https://umap-learn.readthedocs.io/en/latest/index.html. UMAP package is compatible with scikit-learn, making use of the same API and able to be added to sklearn pipelines. UMAP can work as a drop in replacement for t-SNE and other dimension reduction classes from scikit-learn
- LargeVis (Visualizing Large-scale and High-dimensional Data) - Many techniques (like t-SNE, UMAP and LargeVis) first compute a similarity structure of the data points and then project them into a low-dimensional space with the structure preserved. These two steps suffer from considerable computational costs Comparing to tSNE, LargeVis significantly reduces the computational cost of the graph construction step and employs a principled probabilistic model for the visualization step, the objective of which can be effectively optimized through asynchronous stochastic gradient descent with a linear time complexity. Download this algorithm repository and follow the installation instructions. https://github.com/lferry007/LargeVis

[ ]:
```
from sklearn.manifold import TSNE
```

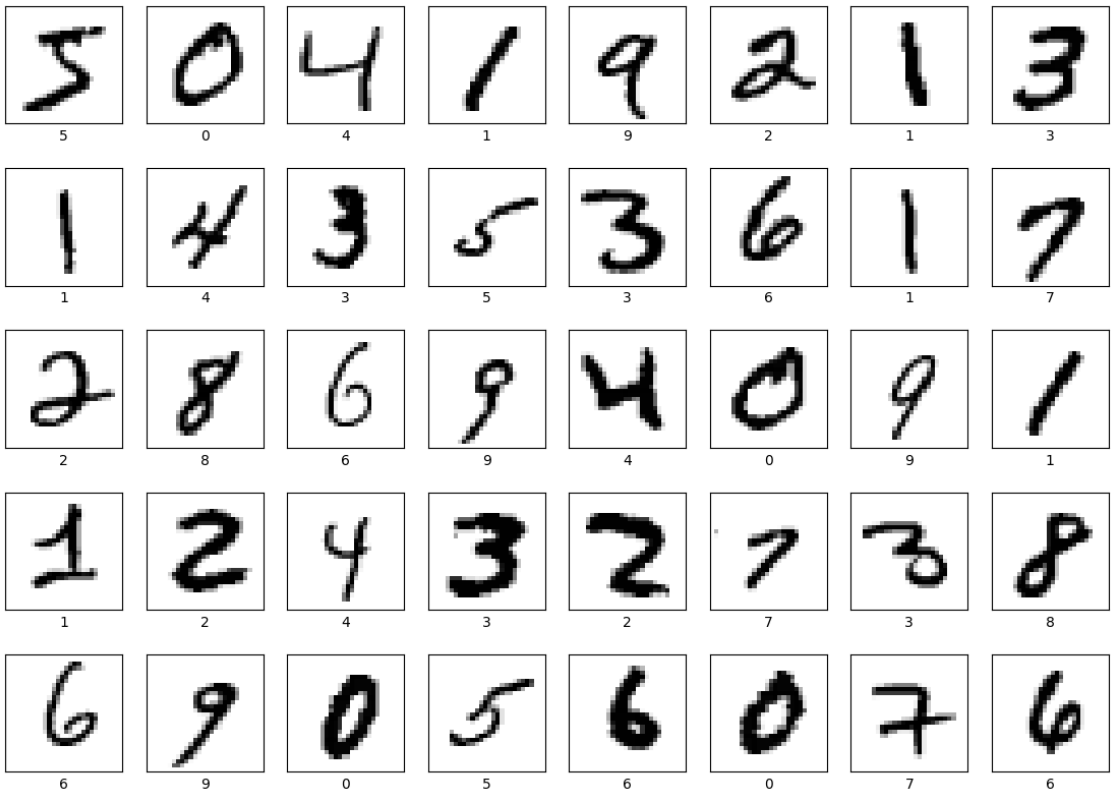
To get data we use the sklearn.datasets.fetch_openml method, which as the name requires, Fetch dataset from openml by name or dataset id. We will use MNIST and Fashion-MNIST(Zalando's article images). Fashion-MNIST is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. Instead of numbers it contains thumbnails of clothes images.

[ ]:
```
mnist = sklearn.datasets.fetch_openml('mnist_784')
fmnist = sklearn.datasets.fetch_openml('Fashion-MNIST')
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/datasets/_openml.py:968:
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
/usr/local/lib/python3.9/dist-packages/sklearn/datasets/_openml.py:968:
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
```

Below are drawings of some samples from mnist and fmnist data sets

[ ]:
```
mnist_names = [i for i in range(10)]

plt.figure(figsize=(14,10))
for i in range(40):
    plt.subplot(5, 8, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(np.array(mnist.data.iloc[i, :]).reshape((28, 28)), cmap=plt.cm.
 ↪binary)
    plt.xlabel(mnist_names[int(mnist.target[i])])
plt.show()
```



[ ]:
```
fmnist_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
                'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

plt.figure(figsize=(14,10))
for i in range(40):
    plt.subplot(5, 8, i+1)
    plt.xticks([])
    plt.yticks([])
```

```
    plt.grid(False)
    plt.imshow(np.array(fmnist.data.iloc[i, :]).reshape((28, 28)), cmap=plt.cm.
  ↪binary)
    plt.xlabel(fmnist_names[int(fmnist.target[i])])
plt.show()
```

Use t-SNE, UMAP and LargeVis to project mnist and fmnist data sets into a 2-dimensional space. For LargeVis, you need to create a function that saves the data to the required by LargeVis txt file format, and a function that loads the resulting file. Draw charts for all visualizations.

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

X_mnist, _, y_mnist, _ = train_test_split(mnist.data, mnist.target,␣
  ↪train_size=10000, stratify=mnist.target, random_state=42)
X_fmnist, _, y_fmnist, _ = train_test_split(fmnist.data, fmnist.target,␣
  ↪train_size=10000, stratify=fmnist.target, random_state=42)

print(f'MNIST: {X_mnist.shape}, F-MNIST: {X_fmnist.shape}')
```

```
MNIST: (10000, 784), F-MNIST: (10000, 784)
```

[ ]:
```
X_fmnist
```

[ ]:
```
       pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  pixel9  \
24557     0.0     0.0     0.0     0.0     0.0     0.0     0.0     1.0     1.0
55234     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
58643     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
61570     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
35134     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
…         …       …       …       …       …       …       …       …
67898     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
57387     0.0     0.0     0.0     0.0     1.0     0.0     0.0     0.0   122.0
53791     0.0     0.0     1.0     0.0     0.0     0.0     0.0     0.0     0.0
420       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
64153     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

       pixel10  …  pixel775  pixel776  pixel777  pixel778  pixel779  \
24557      0.0  …       0.0       1.0       0.0       0.0       0.0
55234      0.0  …     156.0     148.0      55.0       0.0       1.0
58643      0.0  …       0.0       0.0       0.0       0.0       0.0
61570      0.0  …      90.0       0.0       0.0       0.0       0.0
35134      0.0  …       0.0       0.0       0.0      30.0      55.0
…          …    …         …         …         …         …
67898      0.0  …       0.0       0.0       0.0       0.0       0.0
57387    104.0  …     138.0     120.0     137.0      89.0       0.0
53791     16.0  …      19.0      19.0       4.0       0.0       0.0
420        0.0  …       0.0       0.0       0.0       0.0       0.0
64153     21.0  …      19.0      32.0      46.0      69.0      46.0

       pixel780  pixel781  pixel782  pixel783  pixel784
24557       0.0       0.0       0.0       0.0       0.0
55234       0.0       0.0       0.0       0.0       0.0
58643       0.0       0.0       0.0       0.0       0.0
61570       0.0       0.0       0.0       0.0       0.0
35134      30.0       0.0       0.0       0.0       0.0
…          …         …         …         …         …
67898       0.0       0.0       0.0       0.0       0.0
57387       0.0       0.0       0.0       0.0       0.0
53791       0.0       0.0       1.0       0.0       0.0
420         0.0       0.0       0.0       0.0       0.0
64153       0.0       0.0       0.0       0.0       0.0

[10000 rows x 784 columns]
```

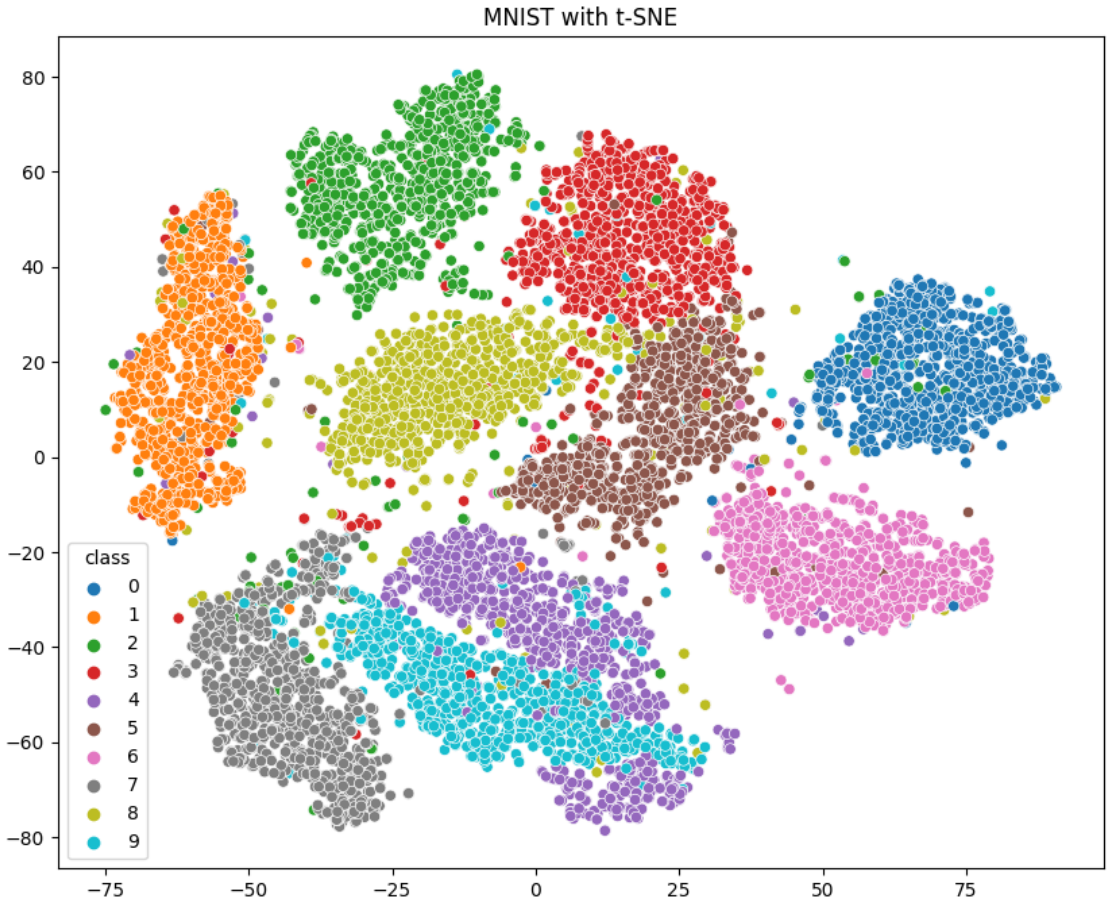
[ ]:
```
y_fmnist.values
```

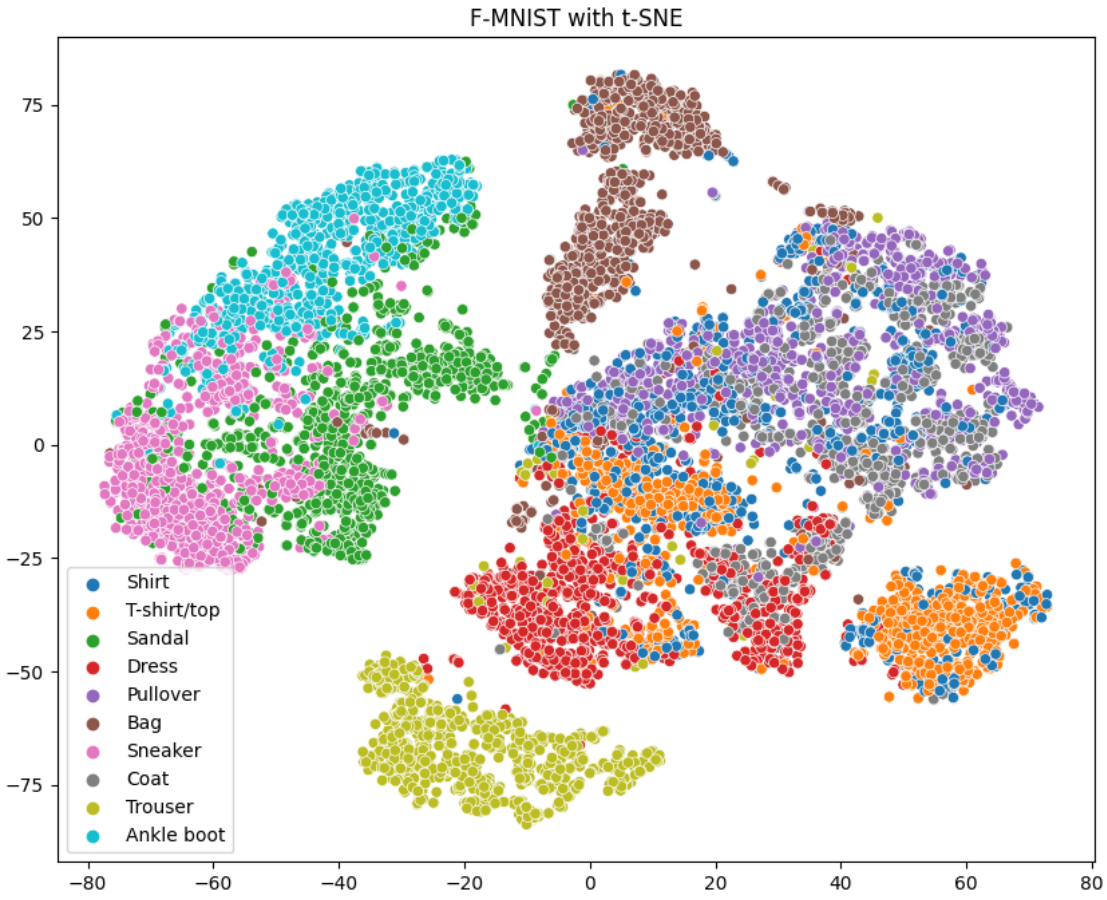
[ ]:
```
['6', '0', '5', '3', '2', …, '8', '0', '0', '8', '6']
Length: 10000
Categories (10, object): ['0', '1', '2', '3', …, '6', '7', '8', '9']
```

[ ]:
```
tsne = TSNE(n_components=2, random_state=0)

mnist_tsne = tsne.fit_transform(X_mnist)
fmnist_tsne = tsne.fit_transform(X_fmnist)
```

[ ]:
```
y_fmnist_names = []

for index in y_fmnist:
    y_fmnist_names.append(fmnist_names[int(index)])
```

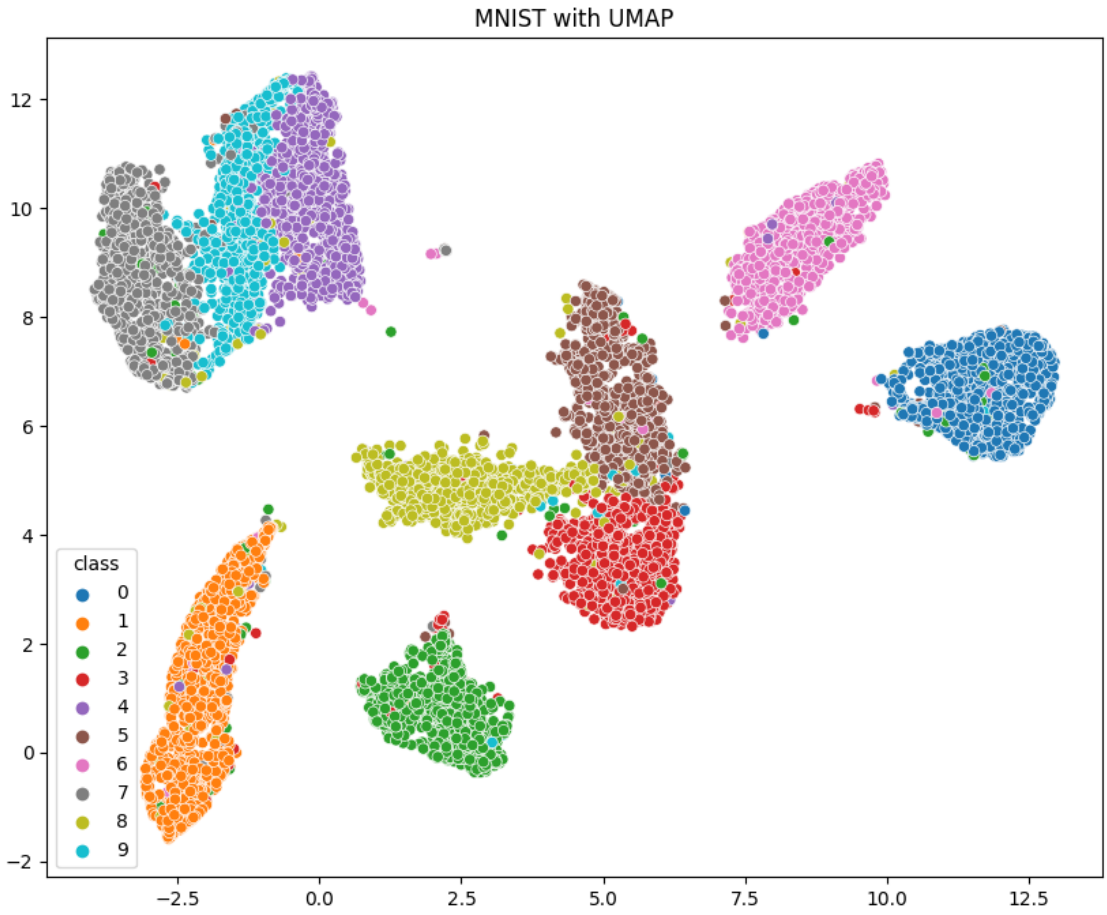
[ ]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(x=mnist_tsne[:, 0], y=mnist_tsne[:, 1], hue=y_mnist)
plt.title('MNIST with t-SNE')
plt.show()

plt.figure(figsize=(10, 8))
sns.scatterplot(x=fmnist_tsne[:, 0], y=fmnist_tsne[:, 1], hue=y_fmnist_names)
plt.title('F-MNIST with t-SNE')
plt.show()
```

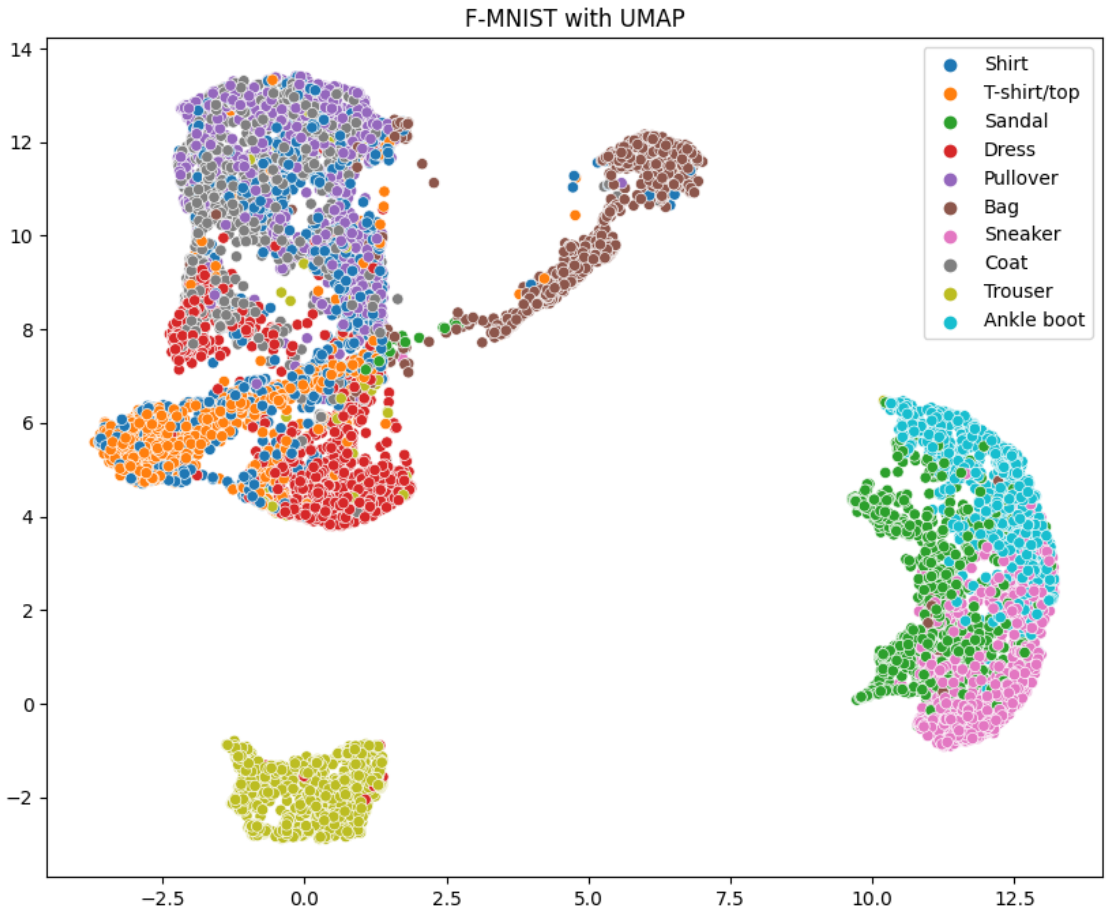




[ ]:
```
reducer = umap.UMAP()
mnist_umap = reducer.fit_transform(X_mnist)

reducer_krowa = umap.UMAP()
fmnist_umap = reducer_krowa.fit_transform(X_fmnist)
```

[ ]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(x=mnist_umap[:, 0], y=mnist_umap[:, 1], hue=y_mnist)
plt.title('MNIST with UMAP')
plt.show()

plt.figure(figsize=(10, 8))
sns.scatterplot(x=fmnist_umap[:, 0], y=fmnist_umap[:, 1], hue=y_fmnist_names)
plt.title('F-MNIST with UMAP')
plt.show()
```


MNIST with UMAP
class
0
1
2
3
4
5
6
7
8
9
12
10
8
6
4
2
0
−2
−2.5
0.0
2.5
5.0
7.5
10.0
12.5



[ ]:
```
!pip install trimap
```

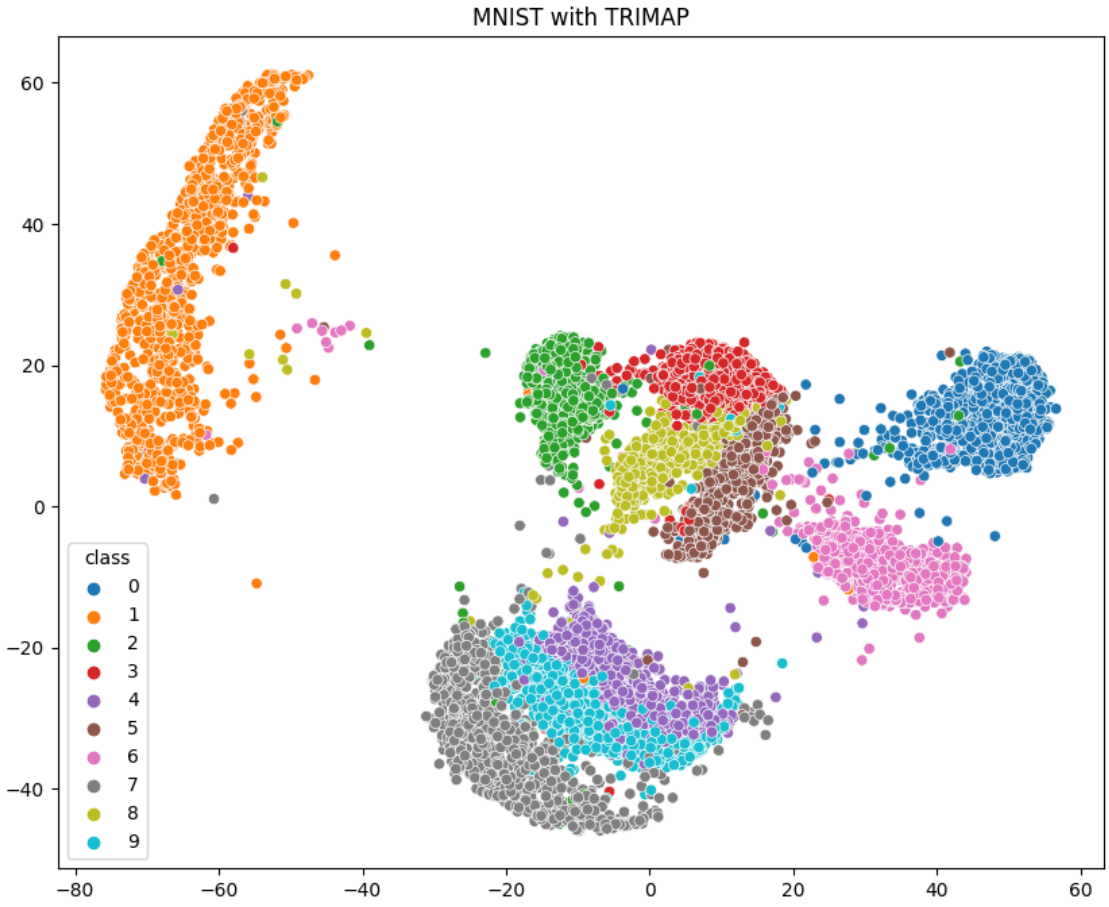
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: trimap in /usr/local/lib/python3.9/dist-packages
(1.1.4)
Requirement already satisfied: numba>=0.34 in /usr/local/lib/python3.9/dist-
packages (from trimap) (0.56.4)
Requirement already satisfied: scikit-learn>=0.16 in
/usr/local/lib/python3.9/dist-packages (from trimap) (1.2.2)
Requirement already satisfied: annoy>=1.11 in /usr/local/lib/python3.9/dist-
packages (from trimap) (1.17.1)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in
/usr/local/lib/python3.9/dist-packages (from numba>=0.34->trimap) (0.39.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-
packages (from numba>=0.34->trimap) (67.6.1)
Requirement already satisfied: numpy<1.24,>=1.18 in
/usr/local/lib/python3.9/dist-packages (from numba>=0.34->trimap) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-
```

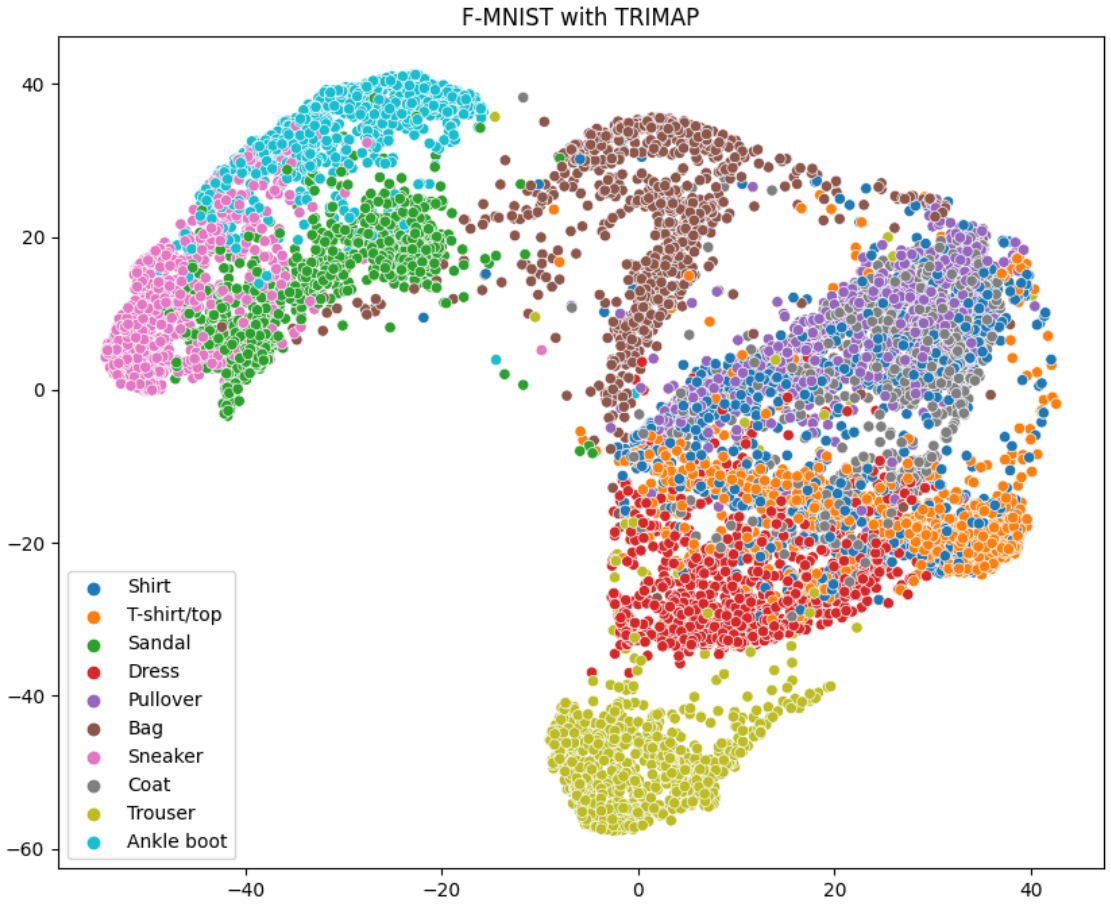
```
packages (from scikit-learn>=0.16->trimap) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-
packages (from scikit-learn>=0.16->trimap) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.16->trimap) (3.1.0)
```

[ ]:
```
import trimap
```

[ ]:
```
reducer = trimap.TRIMAP()
mnist_trimap = reducer.fit_transform(X_mnist)

reducer_krowa = trimap.TRIMAP()
fmnist_trimap = reducer_krowa.fit_transform(X_fmnist)
```

[ ]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(x=mnist_trimap[:, 0], y=mnist_trimap[:, 1], hue=y_mnist)
plt.title('MNIST with TRIMAP')
plt.show()

plt.figure(figsize=(10, 8))
sns.scatterplot(x=fmnist_trimap[:, 0], y=fmnist_trimap[:, 1],␣
 ↪hue=y_fmnist_names)
plt.title('F-MNIST with TRIMAP')
plt.show()
```


MNIST with TRIMAP
60
40
20
0
−20
−40
−80
−60
−40
−20
0
20
40
60
class
0
1
2
3
4
5
6
7
8
9



In order to compare the results of these three methods, calculate for each case the average distance between two points belonging to the same class divided by the average distance between points belonging to 2 different classes

[ ]:
```
from sklearn.metrics import silhouette_score
```

[ ]:
```
mnist_silhouette = silhouette_score(X_mnist, y_mnist)
mnist_tsne_silhouette = silhouette_score(mnist_tsne, y_mnist)
mnist_umap_silhouette = silhouette_score(mnist_umap, y_mnist)
mnist_trimap_silhouette = silhouette_score(mnist_trimap, y_mnist)
```

[ ]:
```
fmnist_silhouette = silhouette_score(X_fmnist, y_mnist)
fmnist_tsne_silhouette = silhouette_score(fmnist_tsne, y_mnist)
fmnist_umap_silhouette = silhouette_score(fmnist_umap, y_mnist)
fmnist_trimap_silhouette = silhouette_score(fmnist_trimap, y_mnist)
```

[ ]:
```
methods = ['T-SNE', 'UMAP', 'TRIEMAP']*2
before_embedding = [mnist_silhouette]*3 + [fmnist_silhouette]*3
after_embedding = [mnist_tsne_silhouette,
```

```
                  mnist_umap_silhouette,
                  mnist_trimap_silhouette,
                  fmnist_tsne_silhouette,
                  fmnist_umap_silhouette,
                  fmnist_trimap_silhouette]
dataset = ['MNIST']*3 + ['FMNIST']*3
```

[ ]:
```
results_silhouette = pd.DataFrame({'Metoda': methods,
                       'Przed zanurzeniem': before_embedding,
                       'Po zanurzeniu': after_embedding,
                       'Zbiór danych': dataset})
```

[ ]:
```
results_silhouette['Różnica'] = abs(results_silhouette['Przed zanurzeniem'] -␣
↪results_silhouette['Po zanurzeniu'])
```

[ ]:
```
results_silhouette
```

[ ]:

	Metoda	Przed zanurzeniem	Po zanurzeniu	Zbiór danych	Różnica
0	T-SNE	0.043155	0.338948	MNIST	0.295792
1	UMAP	0.043155	0.424967	MNIST	0.381812
2	TRIEMAP	0.043155	0.371517	MNIST	0.328362
3	T-SNE	-0.008280	-0.020407	FMNIST	0.012128
4	UMAP	-0.008280	-0.023492	FMNIST	0.015212
5	TRIEMAP	-0.008280	-0.021526	FMNIST	0.013247

[ ]:
```
from sklearn.manifold import trustworthiness
```

[ ]:
```
mnist_tsne_trustworthiness = trustworthiness(mnist_tsne, X_mnist)
mnist_umap_trustworthiness = trustworthiness(mnist_umap, X_mnist)
mnist_trimap_trustworthiness = trustworthiness(mnist_trimap, X_mnist)
```

[ ]:
```
fmnist_tsne_trustworthiness = trustworthiness(fmnist_tsne, X_fmnist)
fmnist_umap_trustworthiness = trustworthiness(fmnist_umap, X_fmnist)
fmnist_trimap_trustworthiness = trustworthiness(fmnist_trimap, X_fmnist)
```

[ ]:
```
measures = [mnist_tsne_trustworthiness,
            mnist_umap_trustworthiness,
            mnist_trimap_trustworthiness,
            fmnist_tsne_trustworthiness,
            fmnist_umap_trustworthiness,
            fmnist_trimap_trustworthiness]
```

[ ]:
```
results_trustworthiness = pd.DataFrame({'Metoda': methods,
                                        'Miara': measures,
                                        'Zbiór danych': dataset})
```

[ ]:
```
results_trustworthiness
```

[ ]:

	Metoda	Miara	Zbiór danych
0	T-SNE	0.979384	MNIST
1	UMAP	0.981357	MNIST
2	TRIEMAP	0.975682	MNIST
3	T-SNE	0.988719	FMNIST
4	UMAP	0.990267	FMNIST
5	TRIEMAP	0.990227	FMNIST

[ ]:
```
from local_score import LocalMetric
```

[ ]:
```
local_metrics_mnist = LocalMetric()
local_metrics_fmnist = LocalMetric()
```

[ ]:
```
local_metrics_mnist.calculate_knn_gain_and_dr_quality(
    X_lds=mnist_tsne,
    X_hds=X_mnist,
    labels=np.array(y_mnist.astype(str).astype(int)),
    method_name='T-SNE')
```

```
Calculating d_hd
T-SNE
```

[ ]:
```
local_metrics_mnist.calculate_knn_gain_and_dr_quality(
    X_lds=mnist_umap,
    X_hds=X_mnist,
    labels=np.array(y_mnist.astype(str).astype(int)),
    method_name='UMAP')
```

```
Calculating d_hd
UMAP
```

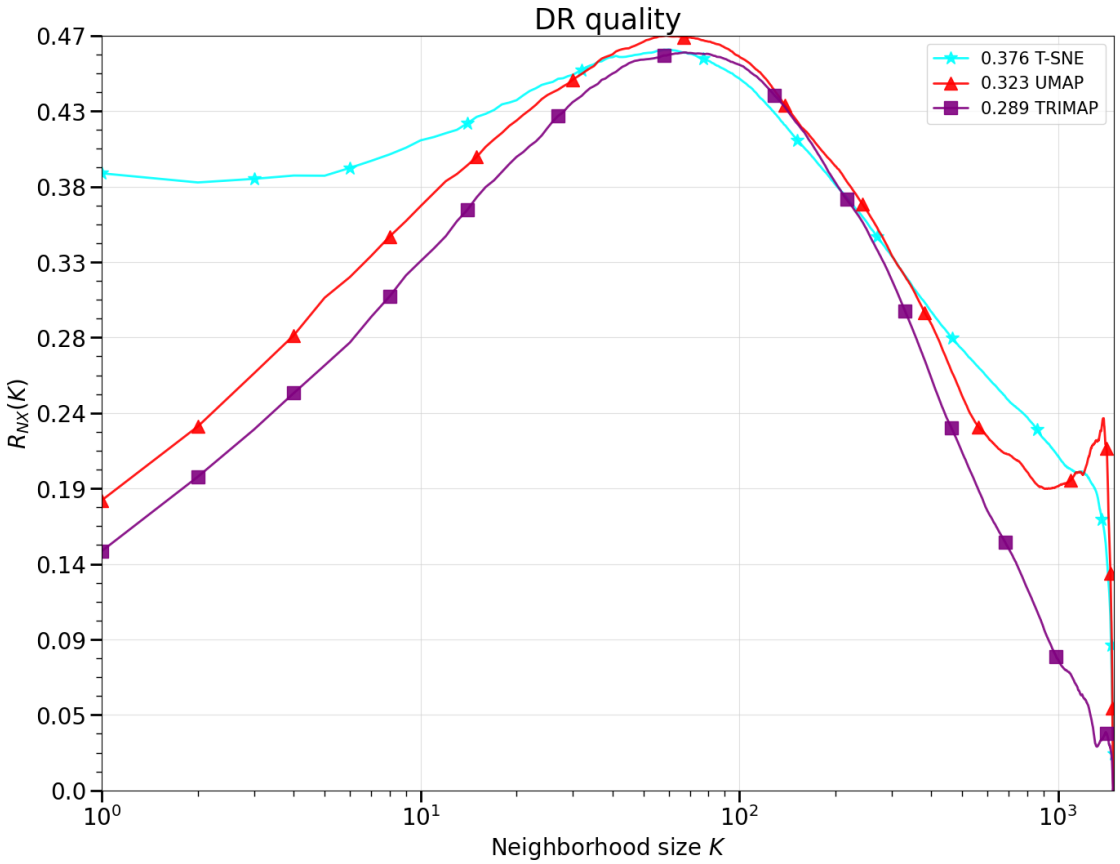
[ ]:
```
local_metrics_mnist.calculate_knn_gain_and_dr_quality(
    X_lds=mnist_trimap,
    X_hds=X_mnist,
    labels=np.array(y_mnist.astype(str).astype(int)),
    method_name='TRIMAP')
```

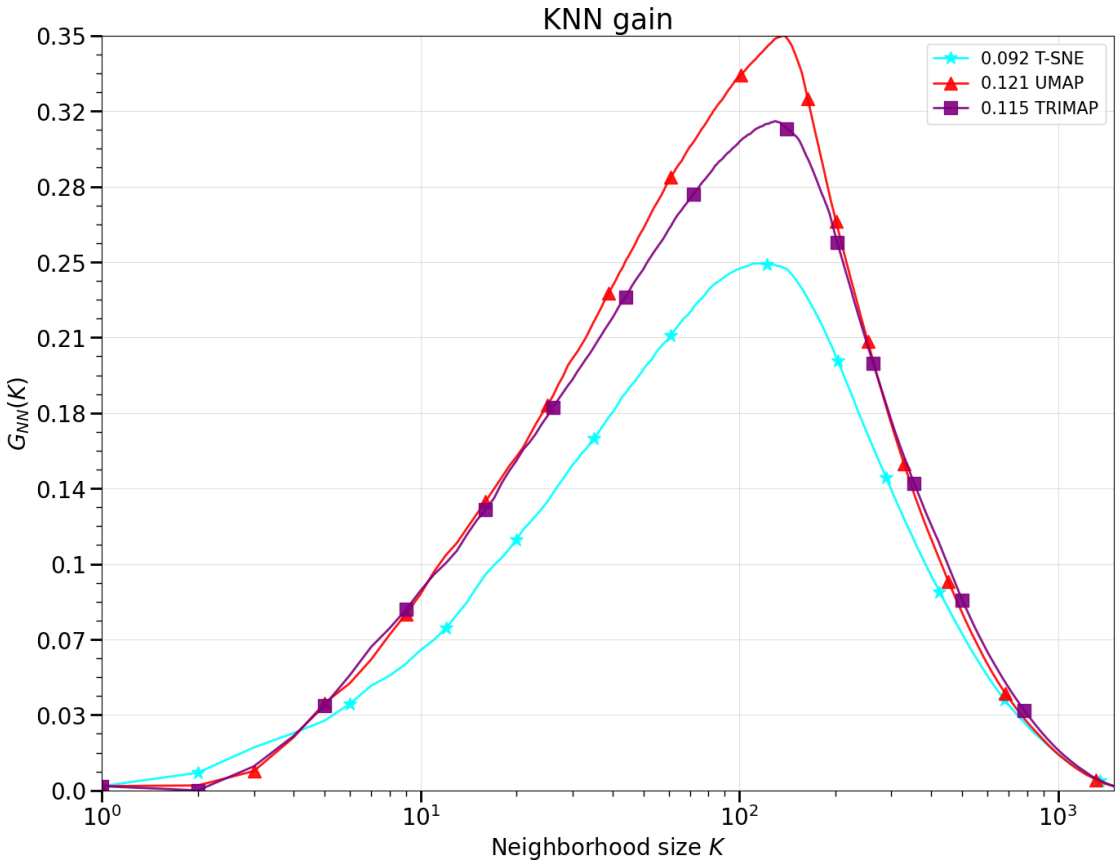
```
Calculating d_hd
TRIMAP
```

[ ]:
```
from importlib import reload
import local_score
reload(local_score)
from local_score import LocalMetric
```

[ ]:
```
local_metrics_mnist.visualize()
```

```
Finished.
```


DR quality
0.376 T-SNE
0.323 UMAP
0.289 TRIMAP
$R_{NX}(K)$
0.47
0.43
0.38
0.33
0.28
0.24
0.19
0.14
0.09
0.05
0.0
10^0
10^1
10^2
10^3
Neighborhood size K



[ ]:
```
local_metrics_fmnist.calculate_knn_gain_and_dr_quality(
    X_lds=fmnist_tsne,
    X_hds=X_fmnist,
    labels=np.array(y_fmnist.astype(str).astype(int)),
    method_name='T-SNE')
```

```
Calculating d_hd
T-SNE
```

[ ]:
```
local_metrics_fmnist.calculate_knn_gain_and_dr_quality(
    X_lds=fmnist_umap,
    X_hds=X_fmnist,
    labels=np.array(y_fmnist.astype(str).astype(int)),
    method_name='UMAP')
```

```
Calculating d_hd
UMAP
```

[ ]:
```
local_metrics_fmnist.calculate_knn_gain_and_dr_quality(
    X_lds=fmnist_trimap,
    X_hds=X_fmnist,
```

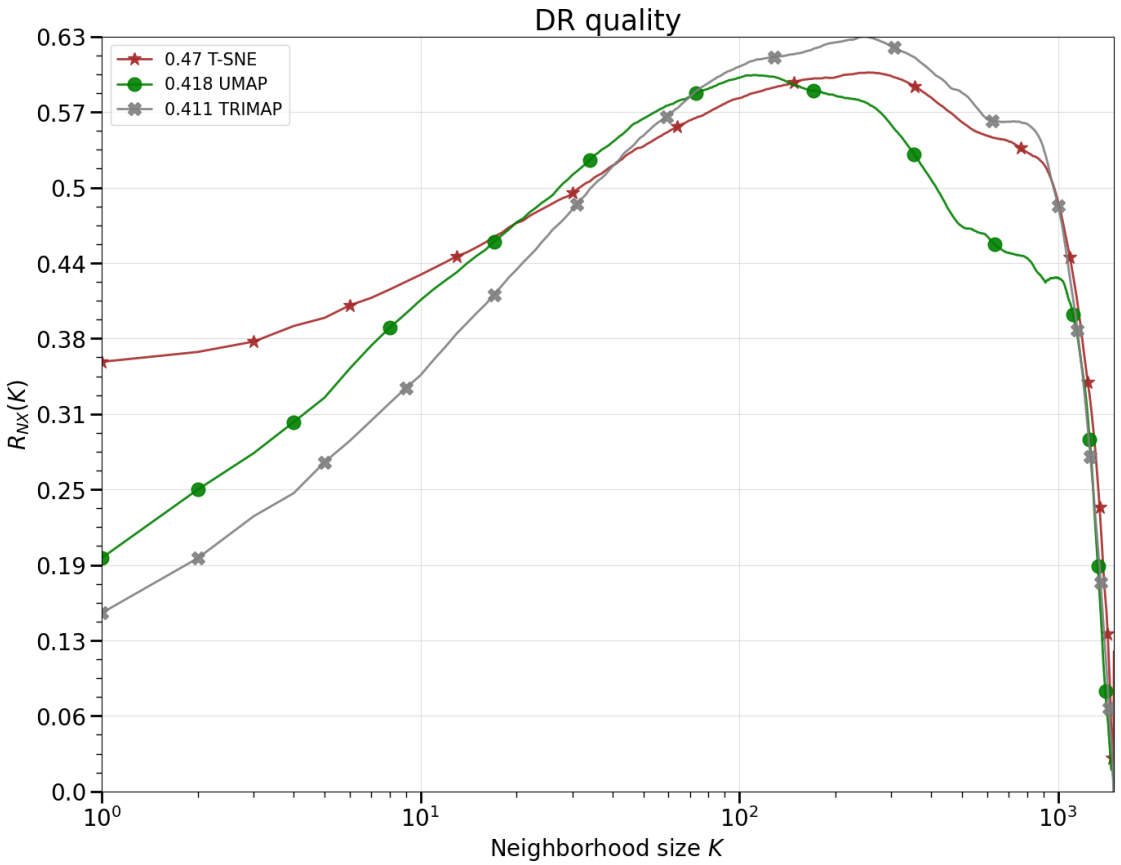
```
    labels=np.array(y_fmnist.astype(str).astype(int)),
    method_name='TRIMAP')
```

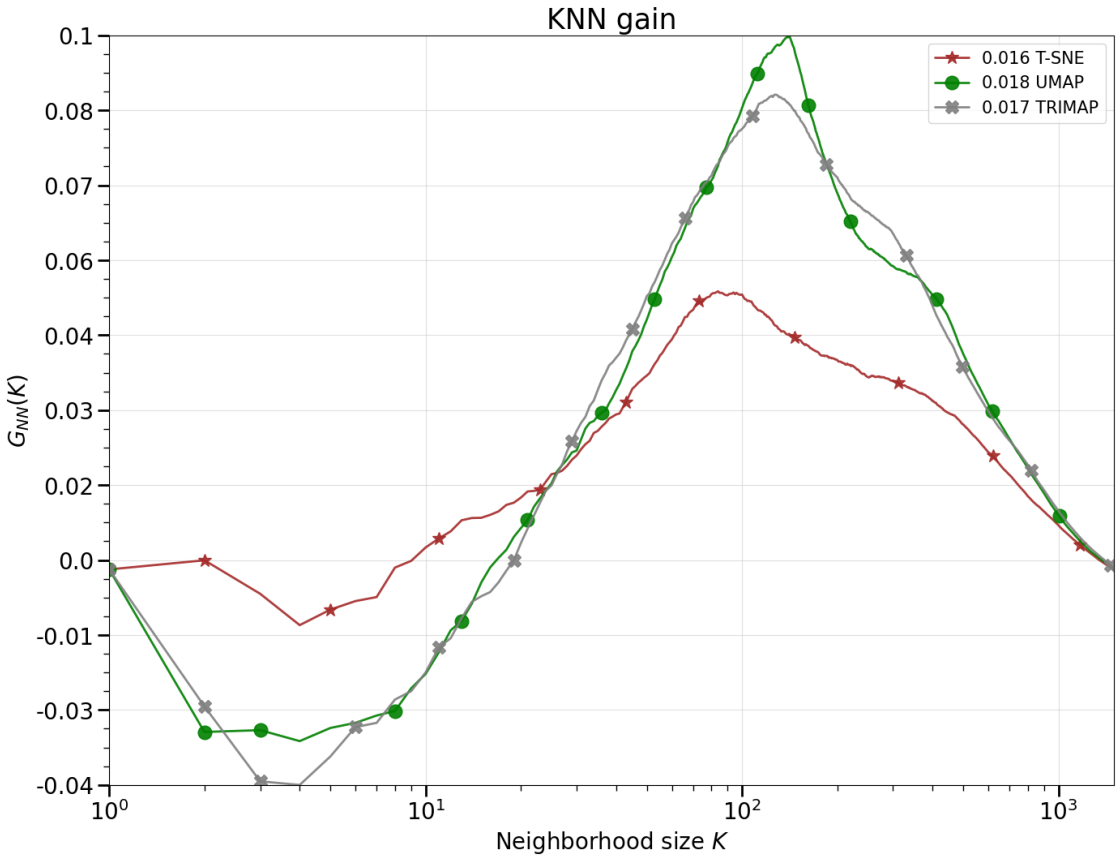
```
Calculating d_hd
TRIMAP
```

[ ]:
```
from importlib import reload
import local_score
reload(local_score)
from local_score import LocalMetric
```

[ ]:
```
local_metrics_fmnist.visualize()
```

```
Finished.
```




KNN gain
0.016 T-SNE
0.018 UMAP
0.017 TRIMAP
$G_{NN}(K)$
Neighborhood size K
0.1
0.08
0.07
0.06
0.04
0.03
0.02
0.0
-0.01
-0.03
-0.04
10^0
10^1
10^2
10^3

[ ]: