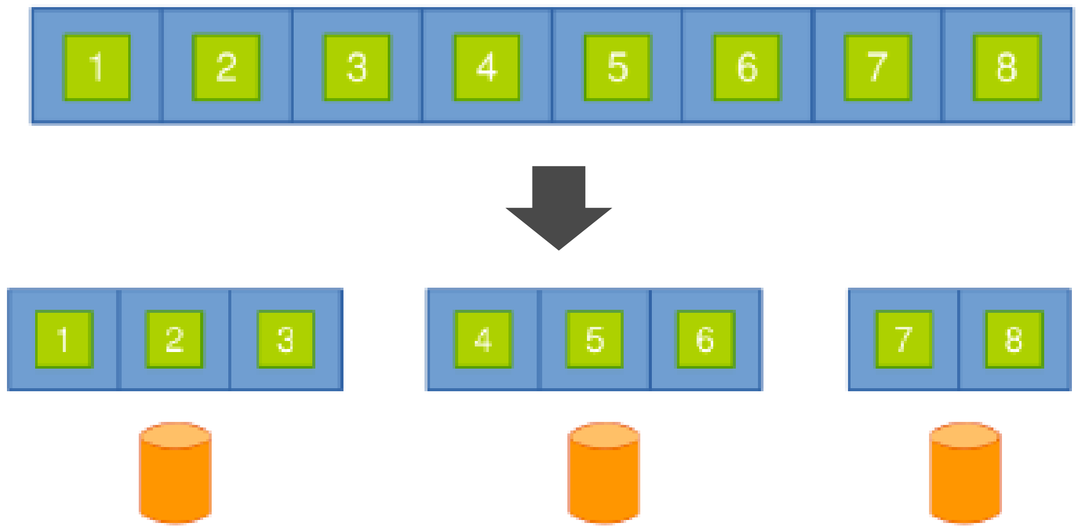
Practice 3

K-Nearest Neighbor

Problem

- Multi-threading problem: predict MNIST dataset's label using K-Nearest Neighbors
- Use "--master local" argument *to select number of **N** cores and execute your pythonscript.py*
  - spark-submit --master local[*N*] YOUR_PYTHON_SCRIPT.py



Dataset for KNN

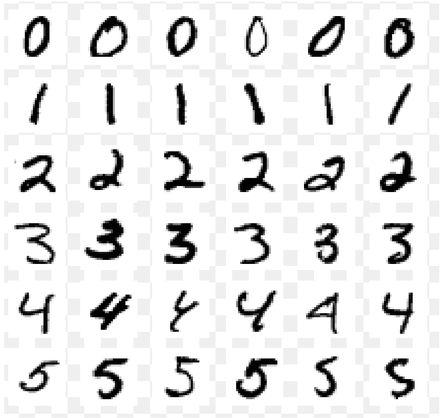
➢ **MNIST : Recognition of handwritten digits**

- There are 10 handwritten digits(0~9) in bitmap format.

➢ 784 Features (28 x 28 pixel values)

1. Pixel 1
2. Pixel 2
…
…
…
783. Pixel 783
784. Pixel 784

* The MNIST Database :



➢ **You can download dataset using *sklearn.datsets.fetch_openml* library**

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html

Practice 3

1. **Compare processing time for classification when you use Multi-threading or single-threading**

   ※ A few minutes will be needed for loading large dataset

   - You can use only one core with "**spark-submit --master local PYTHON_SCRIPT.py**" command
   - Or, maximum number of cores with "**local[*]**"

2. **Use predefined classes in *sklearn.neighbors.KNeighborClassifier***

   Parameters for the method

   - **n_neighbors: 11 (Don't change the other parameters)**

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Practice 3

3. How to train the model using RDD data format

- Before training the model, you need to save data into your memory using ***cache()*** function.
- For example

```
trRDDs.cache()
tsRDDs.cache()
```

- In this example, ***trRDDs***: training data points & ***tsRDDs***: test data points

- Then, you can easily train KNN model provided by scikit-learn using ***fit()*** function
- For example

```
Knn = KNN(n_neighbors = K).fit(trRDDs.collect(), trLabel)
```

- In this example, ***n_neighbors***: number of neighbors to use for kneighbors queries & ***trLabel***: label of training data points, ***collect()***: Return all the elements of datasets as an array at the driver program.

https://spark.apache.org/docs/latest/rdd-programming-guide.html
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Practice 3

4. **After training the models, get the accuracy & F1 score for each label using test data points**
5. **You need to use predefined arguments we suggest.**
   - **Number of train data points: 30,000**

     Use first thirty thousands(30,000) data points as training datasets

   - **Number of test data points: 10,000**

     Use next ten thousands(10,000) data points as test datasets

   - **Number of partitions: 500**

     You can split data when you make it RDDs.

     For example, “ ***RDD = sc.parallelize(Data, numPartition)*** ”

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Submission

- **You need to submit two files(result.txt and time.txt)**
  - **result.txt**

    Write accuracy score of KNN result, using ***sklearn.metrics.accuracy_score*** library

    Then, write F1 score of KNN result, using ***sklearn.metrics.f1_score*** library

    When you calculate F1 score, you need to use parameter **average = 'macro'**
  - **time.txt**

    Write time difference, when you use multi-threading(full thread) and single-threading

```
accuracy : 0.9580
f1score: 0.9578
```

```
multi-threading time: 498.9002
single-threading time: 1816.3231
```

Windows

```
accuracy : 0.9580
f1score: 0.9578
```

```
multi-threading time: 569.6994
single-threading time: 1528.6397
```

Linux