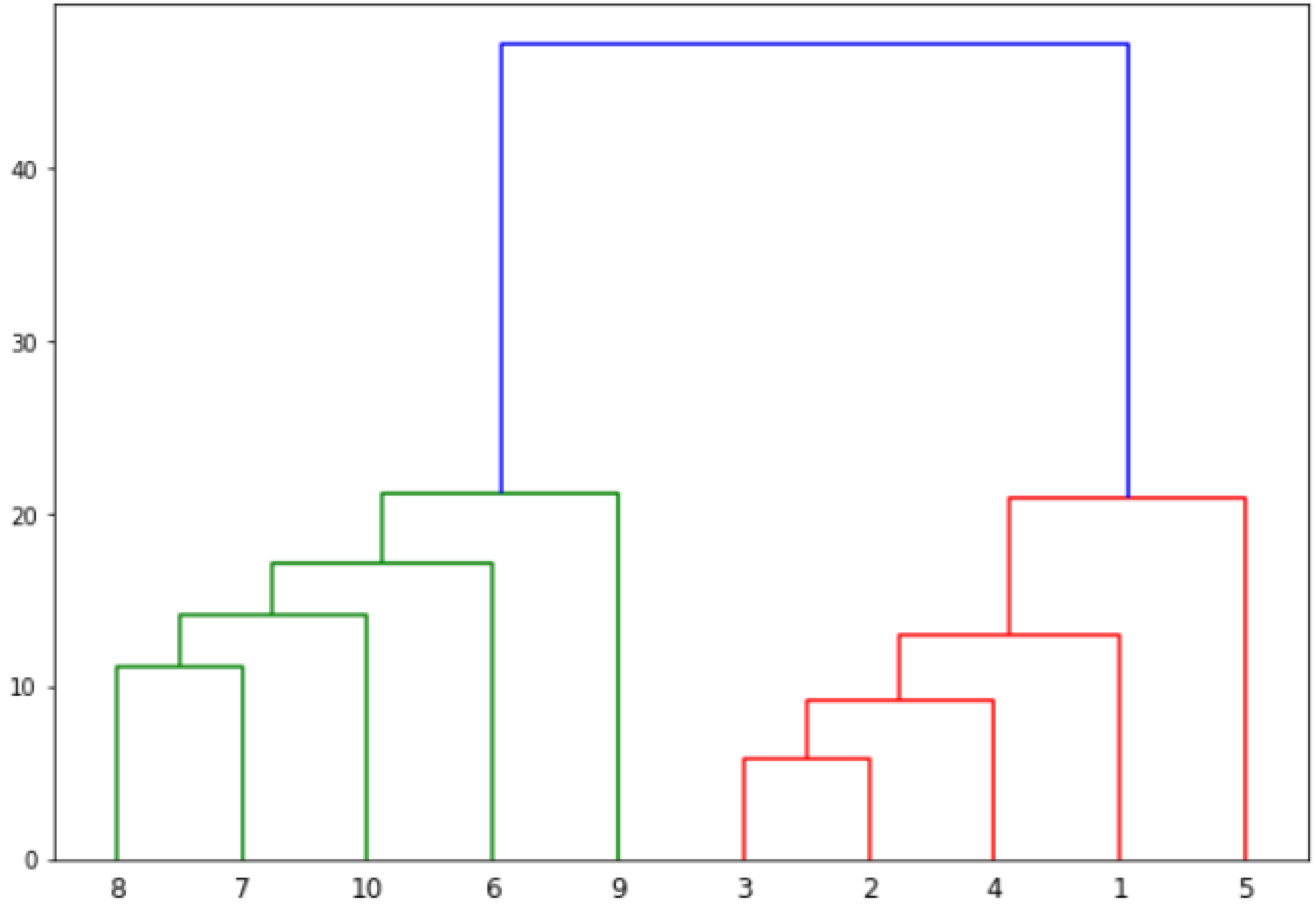
Practice 5

Hierarchical Clustering

Introduction to Big Data Analytics(2020)

Problem

- ➢ **Construct spark environment in your local computer and use Bisecting K-Means**



- Use predefined function in :
  - pyspark**.ml.clustering**
  - pyspark**.ml.linalg**

https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/

Dataset

➢ Digits data set

- Each datapoint has an image of a digit with 8x8 pixels.

➢ 5 Statistic Features

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	Integers 0-16

* Reference

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

Practice 5

1. **Use predefined classes in *pyspark.ml.clustering : BisectingKMeans, and in pyspark.mllib.linalg : Vectors***

2. **First, preprocess the data using “sort_by_target” function(See next page).**

3. **Second, train Bisecting K-Means model with training data(we don’t use label of training data).**

4. **After training the models, calculate *NMI* score of test data points.**

   Parameters for Bisecting K-Means

   - K = 10, minDivisibleClustersize = 1.0

https://spark.apache.org/docs/latest/ml-clustering.html

https://spark.apache.org/docs/2.1.3/api/python/_modules/pyspark/mllib/linalg.html

Practice 5

5. **You can sort the data by target like this:**

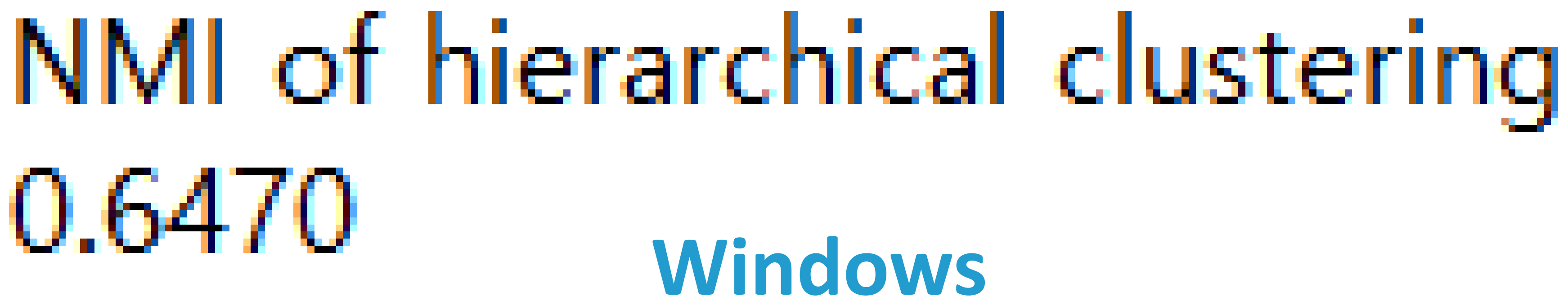
```
nTrain = 1500

def sort_by_target(digits):
    try:
        Data = digits[:,:-1]
        Target = digits[:,-1]

        reorder_train = np.array(sorted([(target, i) for i, target
                                    in enumerate(Target[:nTrain])]))[:,1]
        reorder_test = np.array(sorted([(target, i) for i, target
                                   in enumerate(Target[nTrain:])]))[:,1]
        Data[:nTrain] = Data[reorder_train.astype(np.int64).tolist()]
        Target[:nTrain] = Target[reorder_train.astype(np.int64).tolist()]
        Data[nTrain:] = Data[(reorder_test + nTrain).astype(np.int64).tolist()]
        Target[nTrain:] = Target[(reorder_test + nTrain).astype(np.int64).tolist()]

        digits = np.concatenate((Data,Target.reshape(-1,1)), axis = 1)

        return digits[:nTrain], digits[nTrain:]
    except:
        return None
```

6. ***Call function like this:***

```
trainData, testData = sort_by_target(data_label)
```

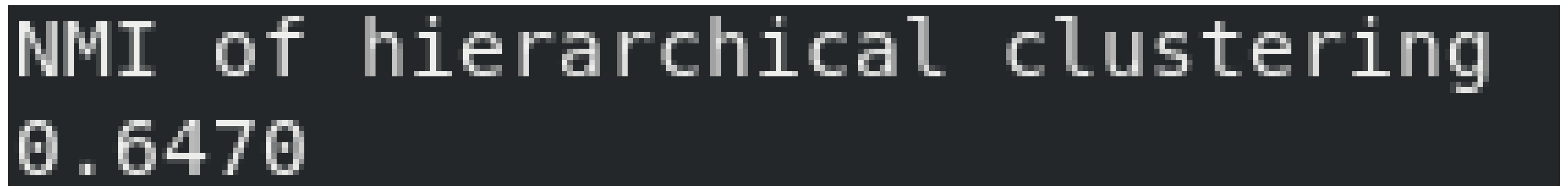
Submission

1. You must submit “**result.txt**” file on I-campus
2. In your **result.txt** file, there must be ***NMI score*** of hierarchical clustering result for digit dataset.
3. ***NMI*** means ***normalized mutual information*** which is a metric to measure some clustering results.
4. Your result.txt file must be like following:



```
NMI of hierarchical clustering
0.6470
```

Windows



```
NMI of hierarchical clustering
0.6470
```

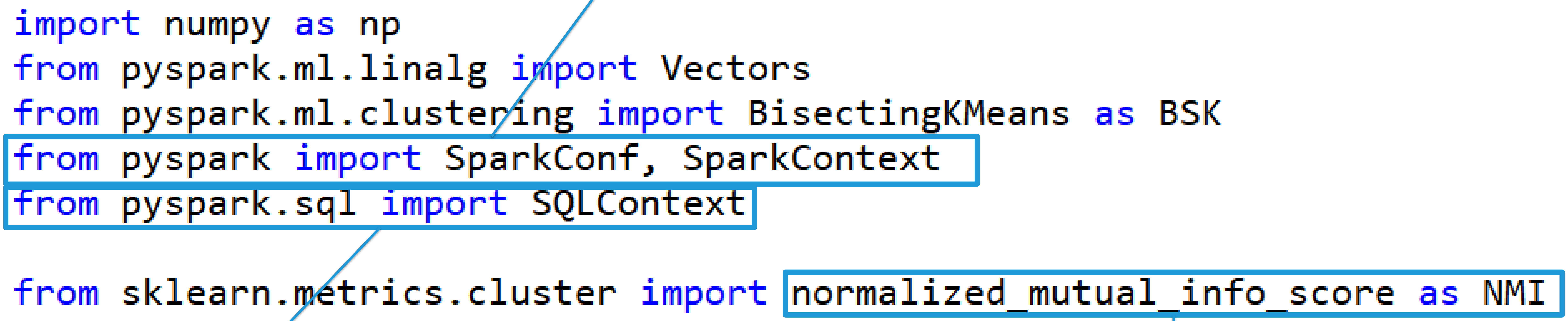
Linux

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html

Solution

- Import package

Import a Spark Package in your program



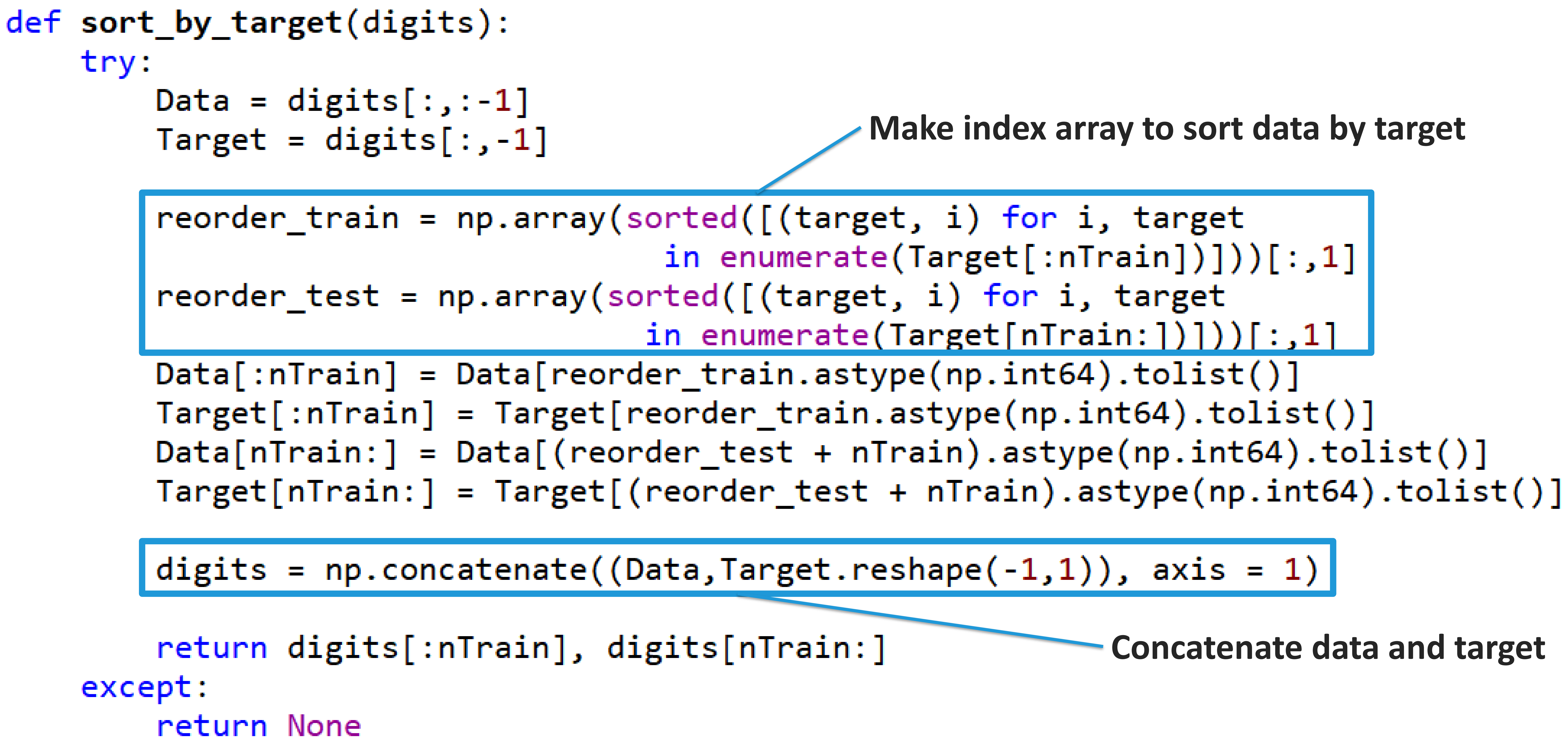
```
import numpy as np
from pyspark.ml.linalg import Vectors
from pyspark.ml.clustering import BisectingKMeans as BSK
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext

from sklearn.metrics.cluster import normalized_mutual_info_score as NMI
```

SQLContext is a class and is used for initializing the functionalities of Spark SQL. SparkContext class object (sc) is required for initializing SQLContext class object.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html

Solution

- Define function to sort given data points by target



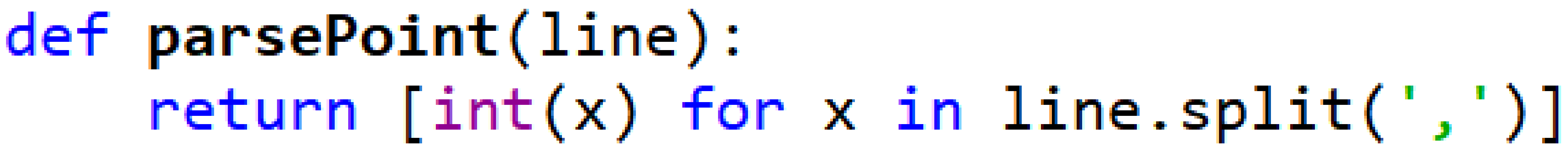
```python
def sort_by_target(digits):
    try:
        Data = digits[:,:-1]
        Target = digits[:,-1]

        reorder_train = np.array(sorted([(target, i) for i, target
                                  in enumerate(Target[:nTrain])]))[:,1]
        reorder_test = np.array(sorted([(target, i) for i, target
                                 in enumerate(Target[nTrain:])]))[:,1]
        Data[:nTrain] = Data[reorder_train.astype(np.int64).tolist()]
        Target[:nTrain] = Target[reorder_train.astype(np.int64).tolist()]
        Data[nTrain:] = Data[(reorder_test + nTrain).astype(np.int64).tolist()]
        Target[nTrain:] = Target[(reorder_test + nTrain).astype(np.int64).tolist()]

        digits = np.concatenate((Data,Target.reshape(-1,1)), axis = 1)

        return digits[:nTrain], digits[nTrain:]
    except:
        return None
```

Make index array to sort data by target

Concatenate data and target

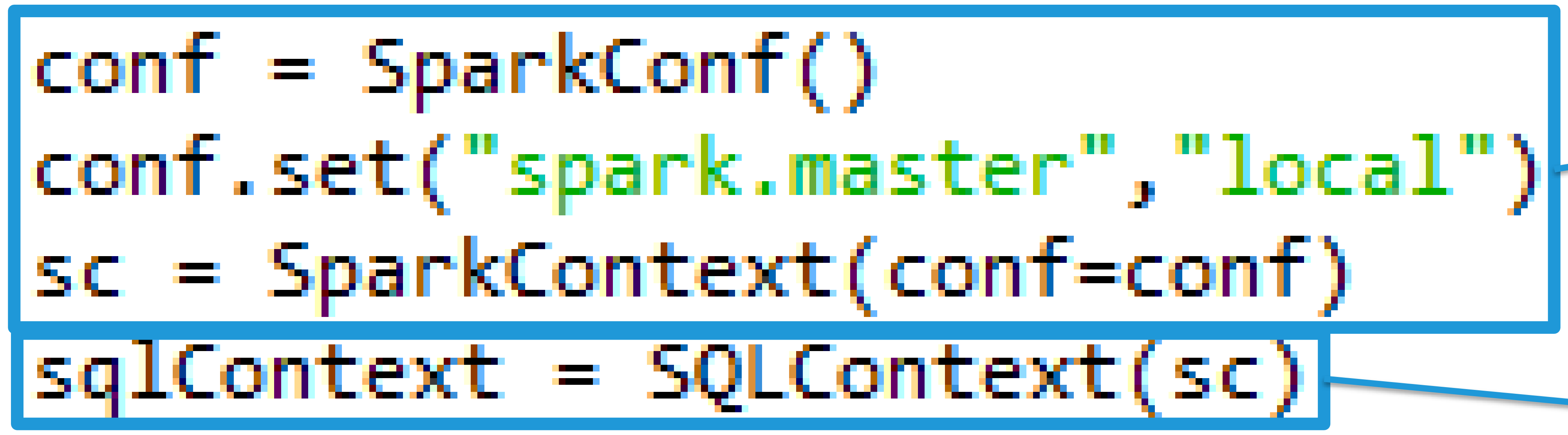
Solution

- ➢ **Define preprocessing function**



```
def parsePoint(line):
    return [int(x) for x in line.split(',')]
```

- ➢ **Initialize a SparkContext**



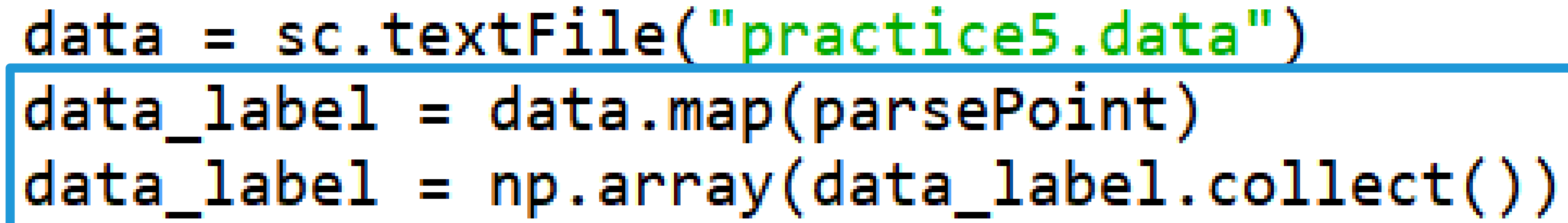
```
conf = SparkConf()
conf.set("spark.master","local")
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
```

Configure Spark with SparkConf
Spark master as Local computer

Later, we use this to make data have dataframe

Solution

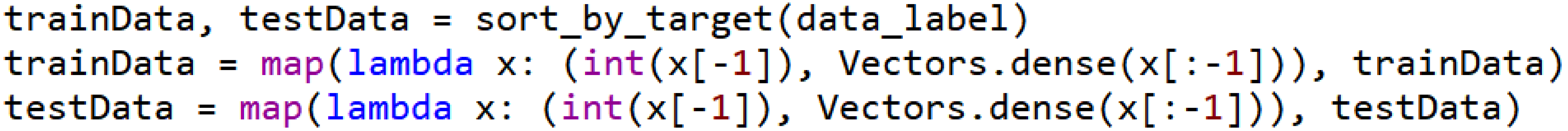
- Load & preprocess data



```
data = sc.textFile("practice5.data")
data_label = data.map(parsePoint)
data_label = np.array(data_label.collect())
```

Preprocess the data and make their type as numpy array

- Sort data and make them RDDs

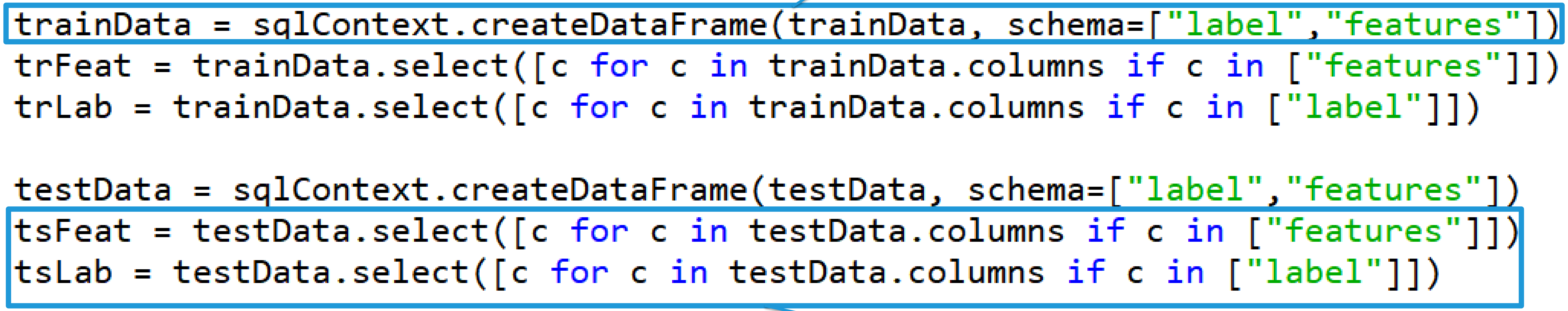


```
trainData, testData = sort_by_target(data_label)
trainData = map(lambda x: (int(x[-1]), Vectors.dense(x[:-1])), trainData)
testData = map(lambda x: (int(x[-1]), Vectors.dense(x[:-1])), testData)
```

Solution

- ➢ Use sqlContext to make RDDs as data frame

Using sqlContext, create data frame which has label attribute and features attribute

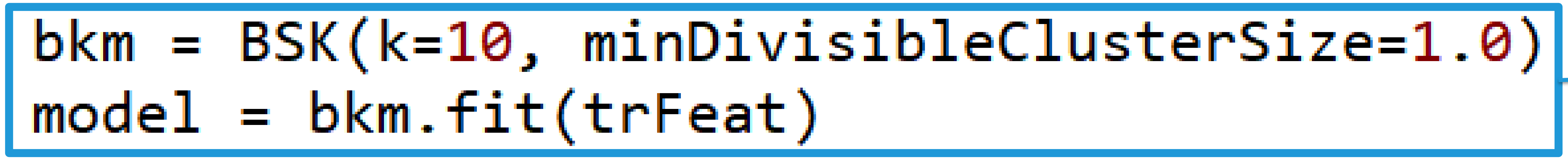


```
trainData = sqlContext.createDataFrame(trainData, schema=["label","features"])
trFeat = trainData.select([c for c in trainData.columns if c in ["features"]])
trLab = trainData.select([c for c in trainData.columns if c in ["label"]])

testData = sqlContext.createDataFrame(testData, schema=["label","features"])
tsFeat = testData.select([c for c in testData.columns if c in ["features"]])
tsLab = testData.select([c for c in testData.columns if c in ["label"]])
```

Split data frame into Features and Labels

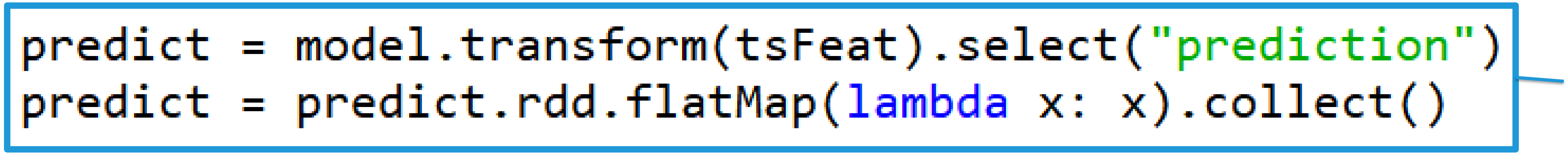
Solution

➢ Train the model & predict clusters



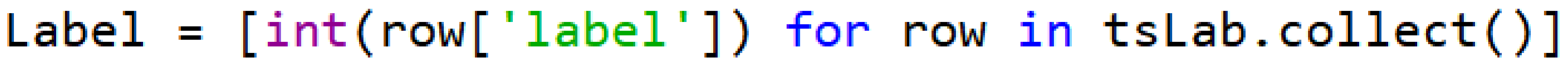
```
bkm = BSK(k=10, minDivisibleClusterSize=1.0)
model = bkm.fit(trFeat)
```

Train bisecting k-means(hierarchical clustering) model



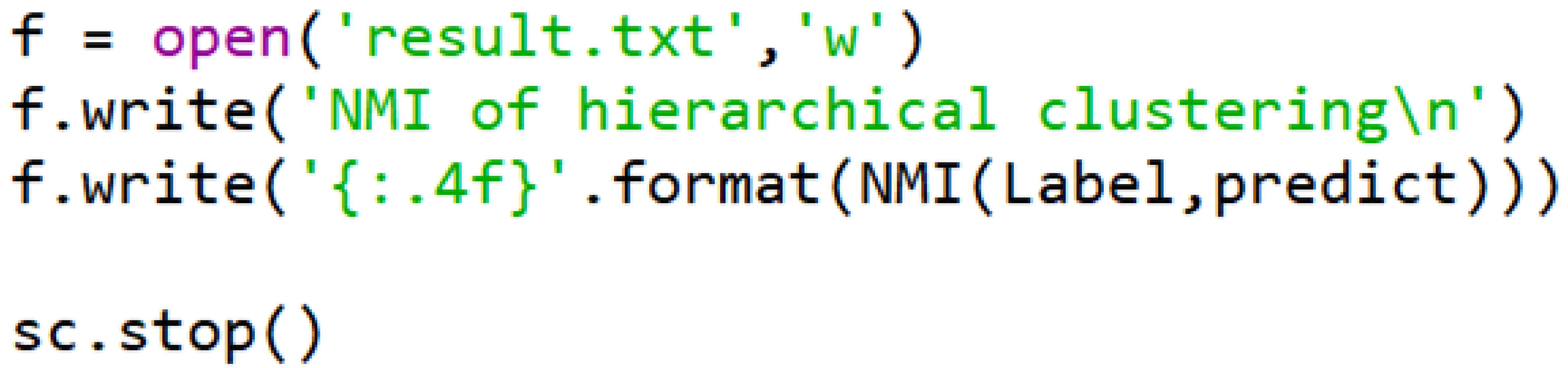
```
predict = model.transform(tsFeat).select("prediction")
predict = predict.rdd.flatMap(lambda x: x).collect()
```

Predict test data's clusters and make the type of it as List



```
Label = [int(row['label']) for row in tsLab.collect()]
```

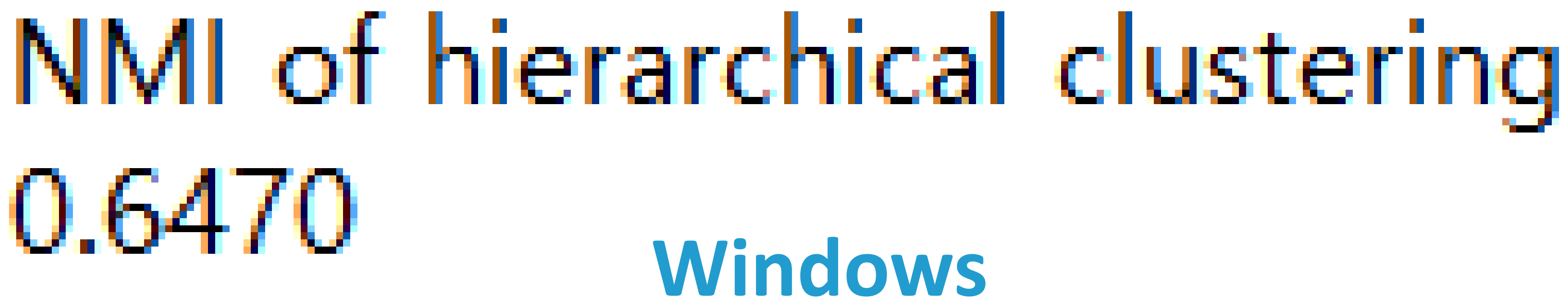
➢ Save the result



```
f = open('result.txt','w')
f.write('NMI of hierarchical clustering\n')
f.write('{:.4f}'.format(NMI(Label,predict)))

sc.stop()
```

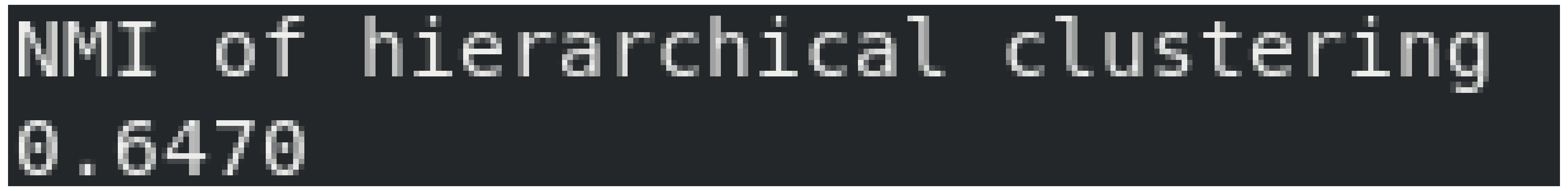
Solution

- Result
  - In your result.txt file, there must be *NMI score* of hierarchical clustering result for digit dataset.
  - *Due date*: May 14th 23:59 PM.
  - Your result.txt file must be like following:



```
NMI of hierarchical clustering
0.6470
```

Windows



```
NMI of hierarchical clustering
0.6470
```

Linux