CSCD01 Deliverable 4

songzhif

zhukevi6

gaoyife5

xuchunan

xuxinzhe

wangs314

Table of Contents

Feature #1 ---------- 3

- https://github.com/scikit-learn/scikit-learn/issues/19072
- Description ---------- 3
- Design ---------- 3
- Changes ---------- 4
- Diagram ---------- 5

Feature #2 ---------- 5

- https://github.com/scikit-learn/scikit-learn/issues/14214
- Description ---------- 5
- Changes ---------- 6
- Diagram ---------- 6

Feature Choice ---------- 7

- Feature #1 ---------- 7
- Feature #2 ---------- 7

Acceptance Test ---------- 7

Feature Request 1

#19072 [New Feature] GroupedTimeSeriesSplit with params - gap and test_size

Link: https://github.com/scikit-learn/scikit-learn/issues/19072

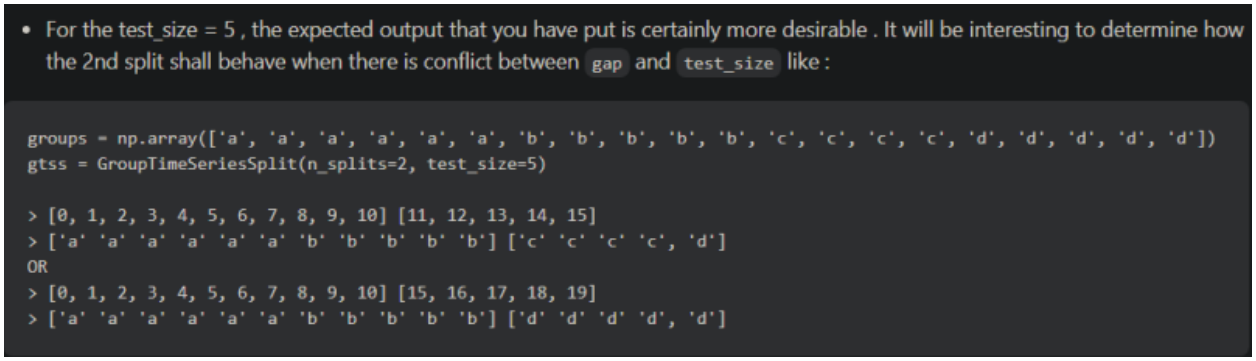
Description:

TimeSeriesSplit is a cross validation strategy that can split an array to multiple test and train arrays. GroupTimeSeriesSplit is the group aware version of TimeSeriesSplit, which takes in an extra group parameter representing the grouping relationship among the elements in the input array. “gap” and “test_size” are attributes of TimeSeriesSplit which regulate the behavior of the split method, yet GroupTimeSeriesSplit does not support these two attributes. Feature # 19072 is to make these two parameters available in GroupTimeSeriesSplit.

- The attribute “gap” indicates the number of groups skipped between the test and train arrays, and is always 0 in the previous implementation of GroupTimeSeriesSplit.
- The attribute “test_size” indicates the size of the returned test arrays. The current behaviour of GroupTimeSeriesSplit always considers test_size as 1.

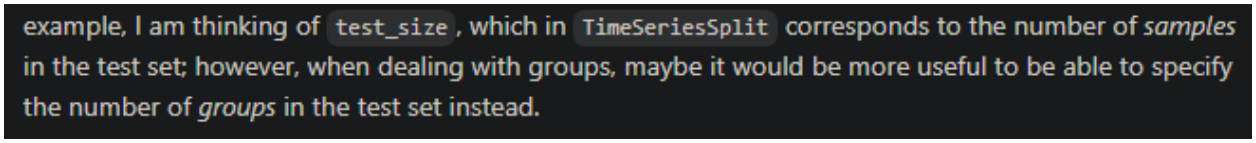
Design:

Produce designs for the selected feature, describing your plans for the organization of new code, as well as all interactions between new code and existing code.



issue creator @getgaurav2 gives two possible behavior

In our opinion, because n_splits already has an error condition that is n_splits+1 > the number of groups, this shows that a group cannot be split from the middle, leading to the first situation mentioned above is inconsistent with existing implementations.



feedback/suggested behavior given by @albertvillanova - Data Scientist

The remaining two possibilities are:

```
version A:
    > [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      [15, 16, 17, 18, 19]
    > ['a' 'a' 'a' 'a' 'a' 'a' 'b' 'b' 'b' 'b' 'b']
      ['d' 'd' 'd' 'd' 'd']  /* 5 elements <= test_size = 5 */
version B:
    /*Let test_size and gap represent the number of groups instead of the
number of samples, to keep the splits within the range, we change test_size
= 1 */
    > [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      [11, 12, 13, 14]
    > ['a' 'a' 'a' 'a' 'a' 'a' 'b' 'b' 'b' 'b' 'b' 'c' 'c' 'c' 'c']
      ['d' 'd' 'd' 'd' 'd'] /* 5 elements in one group <= test_size = 1 */
```

The answer to this question has not yet been discussed. Still, our team thinks that GroupTimeSeriesSplit is the group version of TimeSeriesSplit, so GroupTimeSeriesSplit should perform all its operations on the group level. Moreover, if we choose version B, users can apply a TimeSeriesSplit outside of GroupTimeSeriesSplit, this achieves the same effect as version A. But on the contrary, if we implement version A, users will be unable to perform the behavior mentioned in version B. So based on the consideration of consistency and practicality, we think that test_size and gap should represent the number of groups instead of the number of samples is the best design choice. Hence we choose to switch to(implement) version B based on @getgaurav2 's code

Changes:

doc/modules/cross_validation.rst:

- Described GroupTimeSeriesSplit in the cross_validation's user guide

doc/modules/group_time_series_split.rst:

- Added to user guide for GroupTimeSeriesSplit

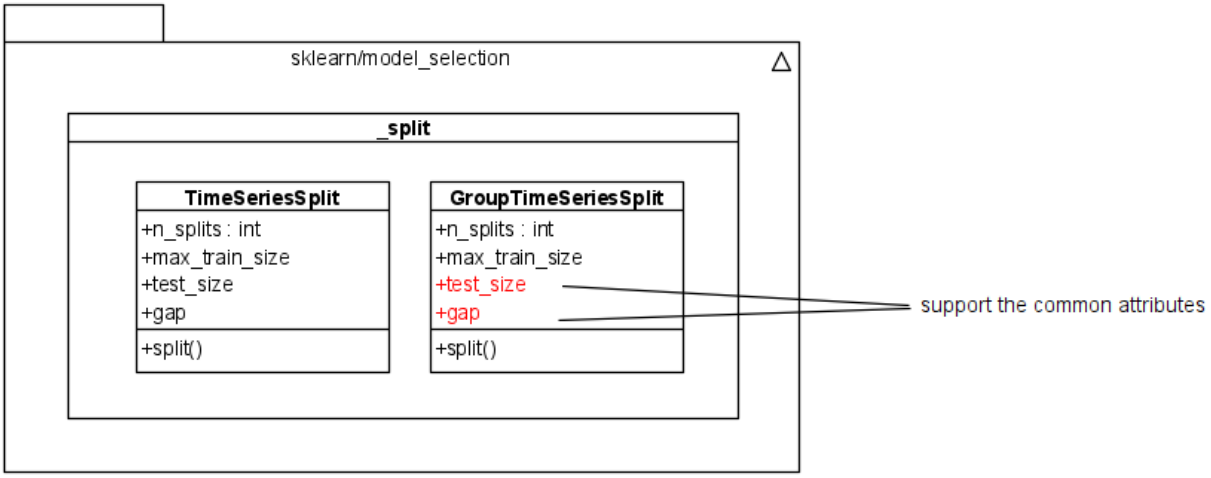
sklearn/model_selection/_split.py:

- Added attributes "gap" and "test_size" to class "GroupTimeSeriesSplit".
- Reimplemented attribute class max_train_size

sklearn/model_selection/tests/test_split.py:

- Added test cases

Diagram:



Feature Request 2

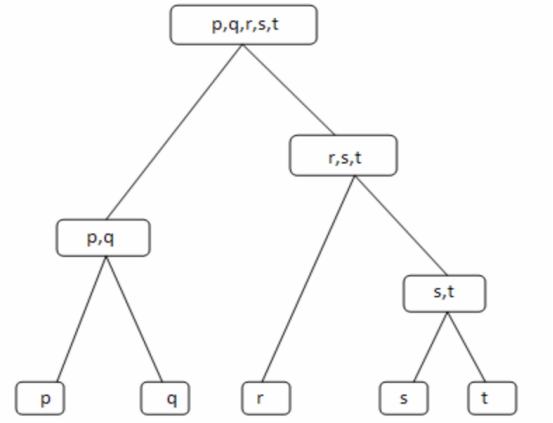
#14214 [New Feature] Add a new algorithm called Bisecting K-means

Link: https://github.com/scikit-learn/scikit-learn/issues/14214

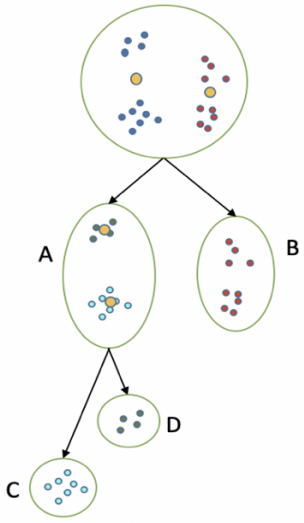
Description:

Bisecting k-means is an approach which combines Divisive Hierarchical Clustering and K-means Clustering. Instead of partitioning the data set into K clusters in each iteration, bisecting k-means algorithm splits one cluster into two sub clusters by using k-means where k=2 at each bisecting step, until k clusters are obtained.

- The implementation of K-means already exists in the sklearn/cluster/kmeans.py file
- A new function called 'DivisiveHierarchical' which starts with a single cluster and divides it towards the bottom level should be implemented under sklearn/cluster folder.
- The new function should call k-means(n_clusters=2) each bisecting step.



Divisive Hierarchical Clustering

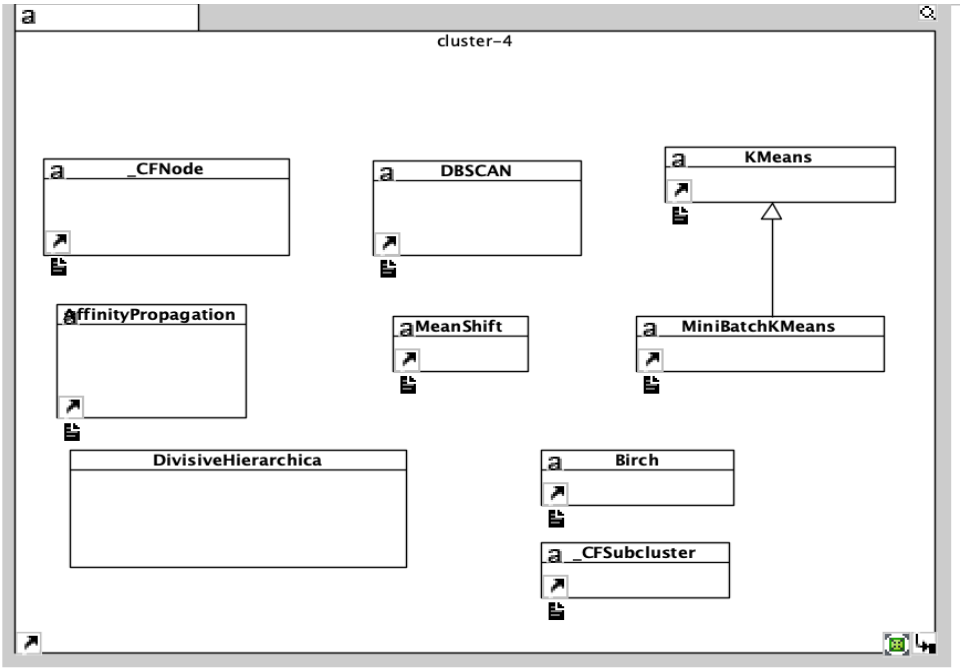


Bisecting k-means

Changes:

The implementation of k-means does not require change, a new function in the folder cluster should be implemented instead. The new function should perform top to down clustering, taking a set of data and set it as a single cluster, then use k-means with k=2 to split the cluster. After splitting, take the sub-cluster with higher SSE(sum of error) to further split until the number of clusters is obtained.

Diagram:



Feature Choice:

#19072

Feature #19072 is the extension of #14257. As we already have some understanding of cross validation strategies, we want to make the new strategy more complete. Adding attributes “gap” and “test_size” to class “GroupTimeSeriesSplit” gains more flexibility in generating the train and test data, and more control allows users to produce more accurate overall results. We believe it is interesting and time worthy to investigate deeper in improving the functionality of module “model_selection”, hence this is the issue we decided to implement.

#14214

The feature#14214 required some knowledge of k-means clustering and divisive hierarchical clustering which may take us more time than the feature #19072 which we have worked before. Additionally, implementing a completely new clustering algorithm is more complex and complicated, adding a new estimator is relatively time consuming. We are definitely willing to implement this new feature if time permits since this new feature seems interesting and helpful. But we have already worked on ‘GroupTimeSeriesSplit’ for A3 and have some understanding of it, implement feature #19072 is more time worthy. Therefore we chose #19072 rather than #14214.

Acceptance Test:

Check number of splits

Steps:

1. Create groups:

```
groups = np.array(['a', 'a', 'a', 'a', 'a', 'a',
                   'b', 'b', 'b', 'b', 'b',
                   'c', 'c', 'c', 'c',
                   'd', 'd', 'd'])
```

2. Create two arrays:

```
X = y = np.ones(len(groups))
```

3. Create a GroupTimeSeriesSplit:

```
gtss = GroupTimeSeriesSplit(n_splits=3)
```

4. Print the number of splits:

```
print(gtss.get_n_splits(X, y, groups))
```

Expected outcome:

- 3 should be printed as the correct output

.

Check indices of train and test data

Steps:

1. Follow the steps 1-3 in **Check number of splits.**
2. Print the indices of train and test data:

```
for train_index, test_index in gtss.split(X, y, groups):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Expected outcome:

```
TRAIN: [0, 1, 2, 3, 4, 5] TEST: [6, 7, 8, 9, 10]
TRAIN: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] TEST: [11, 12, 13, 14]
TRAIN: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] TEST: [15,
16, 17]
```

should be printed as the correct output.

Check indices of train and test data with different max_train_size, n_splits, and gap

Steps:

1. Follow the steps 1-2 in **Check number of splits**.
2. Create a GroupTimeSeriesSplit with 3 as maximum number of groups for a single training set, 2 as number of splits, 1 as number of groups in the test set, and 1 as number of groups in samples to exclude from the end of each train set before the test set:

```
gtss = GroupTimeSeriesSplit(max_train_size=3, n_splits=2,
                            test_size=1, gap=1)
```

3. Print the indices of train and test data:

```
for train_index, test_index in group_timeseriessplit.split(X, y,
groups):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Expected outcome:

```
TRAIN: [0, 1, 2, 3, 4, 5] TEST: [11, 12, 13, 14]
TRAIN: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] TEST: [15, 16, 17]
```

should be printed as the correct output.

Create a GroupTimeSeriesSplit with no groups

Steps:

1. Follow step 1 in **Check number of splits.**
2. Try to create a GroupTimeSeriesSplit:

```
next(GroupTimeSeriesSplit(n_splits=3).split(groups))
```

Expected outcome:

```
ValueError: The 'groups' parameter should not be None
```

Create a GroupTimeSeriesSplit with more folds than groups

Steps:

1. Create groups:

```
groups = np.array([1, 1, 1, 2, 2])
```

2. Create two arrays:

```
X = y = np.ones(len(groups))
```

3. Try to create a GroupTimeSeriesSplit with more folds than groups:

```
next(GroupTimeSeriesSplit(n_splits=3).split(X, y, groups))
```

Expected outcome:

```
ValueError: Cannot have number of folds=4 greater than the number of
groups=2
```

Create a GroupTimeSeriesSplit with non-continuous groups

Steps:

1. Create non-continuous groups:

```
groups = np.array(['a', 'a', 'a', 'a', 'a', 'a',
                   'b', 'b', 'b', 'b', 'b',
                   'c', 'c', 'c', 'c', 'a',
                   'd', 'd'])
```

2. Create two arrays:

```
X = y = np.ones(len(groups))
```

3. Try to create a GroupTimeSeriesSplit with non-continuous groups:

```
next(GroupTimeSeriesSplit(n_splits=3).split(X, y, groups))
```

Expected outcome:

```
ValueError: The groups should be continuous. Found a non-continuous
group at index=15
```

Create a GroupTimeSeriesSplit with too many splits that cause conflict with other parameters

Steps:

1. Follow the steps 1-2 in **Check number of splits**
2. Try to create a GroupTimeSeriesSplit with 3 as number of splits and 2 as number of groups in samples to exclude from the end of each train set before the test set:

```
next(GroupTimeSeriesSplit(n_splits=3, gap=2).split(X, y, groups))
```

Expected outcome:

```
ValueError: Too many splits=3 for number of groups=4 with test_size=1
and gap=2.
```