Mini Quiz 1

以下は、SVRによる予測を 説明変数データ X, 目的変数 (教師) データ y に適用したときの、グリッドサーチとクロスバリデーションによる汎化性能見積もり、ハイパーパラメータ最適化と最終的な予測器構築・保存の一連の手順の概要である (各行の左端は行番号)。このとき以下の空欄を埋めよ。

Mini Quiz 1

```
01: X, y を設定

02: pipe = Pipeline( [ ('scaler': ..., 'svr': ...) ] )
03: param_grid = { ... }
04: grid_cv = KFold(n_splits=5, ...)
05: gen_cv = KFold(n_splits=4, ...)

06: gs = GridSearchCV(pipe, param_grid, cv=grid_cv, scoring=...)
07: nested_score = cross_val_score(gs, X=X, y=y, cv=gen_cv, scoring=...)
08: print(np.sqrt(-nested_score.mean()))

09: gs.fit(X,y)
10: gs_best = gs.best_estimator_
11: print(gs_best)

12: gs_bestをファイルに保存
```

内側CVの分割数:　　外側CVの分割数:
各ハイパーパラメータの探索範囲を指定するための変数の名前:
入力データを標準化してからSVRを適用する一連の手順のためのオブジェクトの名前:
汎化性能が表示される行の行番号:
最終的な最適ハイパーパラメータセットが表示される行の行番号:

Ans. of Mini Quiz 1

```
01: X, y を設定

02: pipe = Pipeline( [ ('scaler': ..., 'svr': ...) ] )
03: param_grid = { ... }
04: grid_cv = KFold(n_splits=5, ...)
05: gen_cv = KFold(n_splits=4, ...)

06: gs = GridSearchCV(pipe, param_grid, cv=grid_cv, scoring=...)
07: nested_score = cross_val_score(gs, X=X, y=y, cv=gen_cv, scoring=...)
08: print(np.sqrt(-nested_score.mean()))

09: gs.fit(X,y)
10: gs_best = gs.best_estimator_
11: print(gs_best)

12: gs_bestをファイルに保存
```

内側CVの分割数： **5**　外側CVの分割数： **4**
ハイパーパラメータの探索範囲を指定するための変数の名前： **param_grid**
入力データを標準化してからSVRを適用する一連の手順のためのオブジェクトの名前： **pipe**
汎化性能が表示される行の行番号： **08**
最終的な最適ハイパーパラメータセットが表示される行の行番号: **11**

Mini Quiz 1

The following is the summary of a series of procedures for estimating generalization performance by grid search and cross-validation, hyperparameter optimization, and final predictor construction and storage when SVR prediction is applied to explanatory variable data X and target variable (teacher signal) data y. The leftmost line of each row is the row number. Fill in the following blanks.

Mini Quiz 1

```
01: SET X, y

02: pipe = Pipeline( [ ('scaler': ..., 'svr': ...) ] )
03: param_grid = { ... }
04: grid_cv = KFold(n_splits=5, ...)
05: gen_cv = KFold(n_splits=4, ...)

06: gs = GridSearchCV(pipe, param_grid, cv=grid_cv, scoring=...)
07: nested_score = cross_val_score(gs, X=X, y=y, cv=gen_cv, scoring=...)
08: print(np.sqrt(-nested_score.mean()))

09: gs.fit(X,y)
10: gs_best = gs.best_estimator_
11: print(gs_best)

12: SAVE gs_best INTO FILE
```

#divisions of the inner CV:　　#divisions of the outer CV:
Name of the variable to specify the search range of each hyperparameter:
Name of the object for the series of procedures, standardize input data and apply SVR:
Line number of the program to show of the generalization performance:
Line number of the program to show the final best hyperparameter set:

Ans. of Mini Quiz 1

```
01: SET X, y

02: pipe = Pipeline( [ ('scaler': ..., 'svr': ...) ] )
03: param_grid = { ... }
04: grid_cv = KFold(n_splits=5, ...)
05: gen_cv = KFold(n_splits=4, ...)

06: gs = GridSearchCV(pipe, param_grid, cv=grid_cv, scoring=...)
07: nested_score = cross_val_score(gs, X=X, y=y, cv=gen_cv, scoring=...)
08: print(np.sqrt(-nested_score.mean()))

09: gs.fit(X,y)
10: gs_best = gs.best_estimator_
11: print(gs_best)

12: SAVE gs_best INTO FILE
```

#divisions of the inner CV: **5** #divisions of the outer CV: **4**
Name of the variable to specify the search range of each hyperparameter: **param_grid**
Name of the object for the series of procedures, standardize input data and apply SVR: **pipe**
Line number of the program to show of the generalization performance: **08**
Line number of the program to show the final best hyperparameter set: **11**