Data Science | Lab 4: Cross Validation and Grid Search

Learning Goals

- setting up an experiment for k-fold cross validation
- computing performance measures for cross validation
- selecting parameters for a grid search
- setting up a pipeline for grid search
- correctly combining grid search and cross validation
- interpretation of the results

k-Fold Cross Validation

Techniques: *k*-fold Cross Validation

1. define the number k of sets
   - usually around $k = 10$
   - higher values imply larger training set and more runs (longer evaluation time!)
   - $k = n$ (number of data available) "leave-one-out": computational expensive!
2. use stratified sampling to define k non-overlapping sets of data $\mathfrak{fold}_1, \ldots \mathfrak{fold}_k$
3. loop i over $1 \ldots k$ and let $\mathfrak{test} = \mathfrak{fold}_i$ and $\mathfrak{train} = \{\mathfrak{fold}_j \; : \; j \neq i\}$
4. for each such fold: apply training, evaluate predictions for testing data and store those
   - the performance measures computed from such a $\mathfrak{test}$ will suffer from the small number of data and be inaccurate
   - leave-one-out: only a single data item is tested, all other will be used in training
5. compute an overall performance estimator by pooling all stored testing data

Since, in the end, the performance measures are computed from a testing set equal to all (!) data available, they have highest possible statistical accuracy.

Recall the principles of a k-Fold Cross Validation (CV) from the lecture. Take a look at scikit-learn's `KFold` documentation and the corresponding User Guide and make yourself familiar with its usage.

Grid Search

Techniques: Grid Search

1. identify parameters that should be optimized:
   - start with presumably high-impact parameters
   - consider optimizing parameters with possible dependencies in a single grid search
2. for each parameter: consider a discrete set of values that should be examined
   - linear vs. logarithmic vs. exponential scale $(1, 2, 3$ or $10, 20, 22$ or $1, 10, 100)$
   - rough scale first, can later be refined around the most promising values
3. set up a loop over all parameter-value configurations and implement excellent documentation of each setup in log files or similar; use a format that can also contain variations of the setup that are introduced manually.
4. consider possible crashes or infinite processing time due to numerical problems or force majeure:
   - safety measures to protect against infinite processing in the inner part of the loop
   - the procedure needs to be restartable after a crash (or catch such an exception)
5. select a performance measure to be optimized (e.g. accuracy)

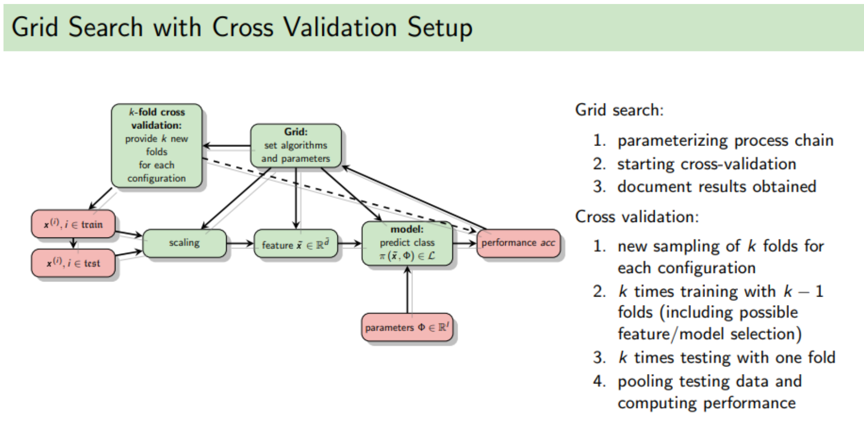
Recall the principles how to set up a grid search (GS) experiment from the lecture. Take a look at scikit-learn's `ParameterGrid` documentation and make yourself familiar with its usage.

Tasks

Sketch the Task

👥 Group work: Sketch the correct order of tasks for GS+CV

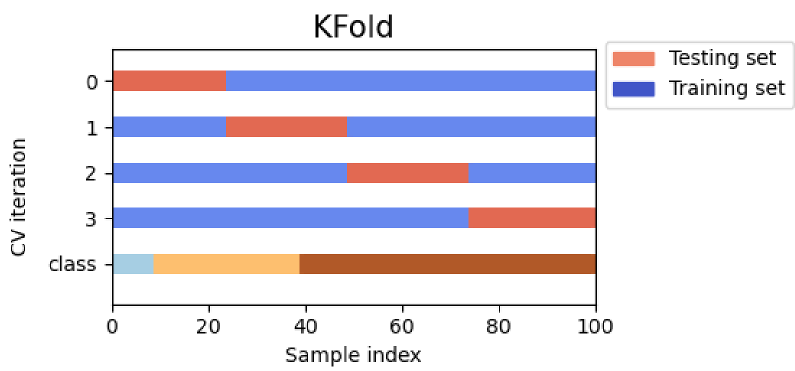
Based on the outline presented in the lecture, try to sketch the final Python script before getting started with the implementation.



Cross Validation

```python
from sklearn.model_selection import KFold
```

1. Reuse the notebook from Lab 3 for the wine data. Make sure to
   - Reuse the same random seed throughout.
   - Use $k = 7$ nearest neighbors.
2. With using `KFold` to produce the data splits, implement cross validation. Make sure to store the predictions on each test fold and print the `classification_report` after having looped over all folds.
3. Try with $k = 3$ and $k = 10$ folds.
4. In order to interpret the results (and fix possible issues), take a close look at the `KFold` visualization from the User Guide (not based on the wine data!):



Grid Search

```python
from sklearn.model_selection import ParameterGrid
```

1. Implement Grid Search in combination with cross validation.
   - Use the following parameters from the `KNeighborsClassifier` for the grid: `n_neighbors` and `p`. Select reasonable values for both.
   - Implement a `for` loop to iterate over all combinations of the grid:

```python
param_grid = ParameterGrid(...)
for pg in param_grid:
    # TODO: cross validation with k folds
    # print classification report for each parameter setting
```

2. Run the Grid Search and print the classification report for each parameter combination.
3. Which parameter combination performs best?

Combining Grid Search and Cross Validation

```python
from sklearn.model_selection import GridSearchCV
```

1. Carefully read the documentation of 📝 GridSearchCV, which combines the mechanisms of the grid search and the cross validation.
2. Reuse the `kNeighborsClassifier` and the `ParameterGrid` (check for correct naming).
3. Set the cross validation splitting strategy to $k = 10$ folds.
4. Evaluate the results using `GridSearchCV`'s built-in methods.
5. Change the parameter `scoring` to use the F1 score for evaluation.
6. Find out how to store/access the best model parametrization.

Homework

Extend the grid with a parameter for switching the scaling of the data on/off. Then, for each test run made so far, enter the cross validation results in your table. Those values are more robust and reliable than those obtained from a single run.

Take the quiz in Moodle. Make sure to have your Python notebook open and your code up and running.

🕘 Deadline: 13.12.2022 (5:00pm)

Further Reading

✨Read the advanced documentation on how to 🌐 control randomness in scikit-learn.