🏁 Wrap-up quiz 6

This quiz requires some programming to be answered.

This wrap-up quiz uses the penguins dataset, but notice that **we do not use the traditional `Species` column** as predictive target:

```
import pandas as pd

dataset = pd.read_csv("../datasets/penguins.csv")

feature_names = [
    "Culmen Length (mm)",
    "Culmen Depth (mm)",
    "Flipper Length (mm)",
]
target_name = "Body Mass (g)"

dataset = dataset[feature_names + [target_name]].dropna(axis="rows", how="any")
dataset = dataset.sample(frac=1, random_state=0).reset_index(drop=True)
data, target = dataset[feature_names], dataset[target_name]
```

We therefore define our problem as a regression problem: we want to predict the body mass of a penguin given its culmen and flipper measurements.

Notice that we randomly shuffled the rows of the dataset after loading it (`dataset.sample(frac=1, random_state=0)`). The reason is to break a spurious order-related statistical dependency that would otherwise cause trouble with the naive cross-validation procedure we use in this notebook. The problem of order-dependent samples will be discussed more in detail on the model evaluation module and is outside of the scope of this quiz for now. Now, evaluate the following tree-based models:

- a decision tree regressor, i.e. `sklearn.tree.DecisionTreeRegressor`
- a random forest regressor, i.e. `sklearn.ensemble.RandomForestRegressor`

Use the default hyper-parameter settings for both models. The only exception is to pass `random_state=0` for all models to be sure to recover the exact same performance scores as the solutions to this quiz.

Evaluate the generalization performance of these models using a 10-fold cross-validation:

- use `sklearn.model_selection.cross_validate` to run the cross-validation routine
- set the parameter `cv=10` to use a 10-fold cross-validation strategy. Store the training score of the cross-validation by setting the parameter `return_train_score=True` in the function `cross_validate` as we will use it later on.

Question

By comparing the cross-validation test scores fold-to-fold, count the number of times a random forest is better than a single decision tree. Select the range which this number belongs to:

- a) [0, 3]: the random forest model is substantially worse than the single decision tree regressor
- b) [4, 6]: both models are almost equivalent
- c) [7, 10]: the random forest model is substantially better than the single decision tree regressor

Select a single answer

Now, train and evaluate with the same cross-validation strategy a random forest with 5 decision trees and another containing 100 decision trees. Once again store the training score.

Question

By comparing the cross-validation test scores fold-to-fold, count the number of times a random forest with 100 decision trees is better than a random forest with 5 decision trees. Select the range which this number belongs to:

- a) [0, 3]: the random forest model with 100 decision trees is substantially worse than the random forest model with 5 decision trees
- b) [4, 6]: both models are almost equivalent
- c) [7, 10]: the random forest model with 100 decision trees is substantially better than the random forest model with 5 decision trees

Select a single answer

Plot the validation curve of the `n_estimators` parameters defined by:

```
n_estimators = [1, 2, 5, 10, 20, 50, 100, 200, 500, 1_000]
```

Question

Select the correct statements below.

- a) the **train score** decreases when `n_estimators` become large (above 500 trees)
- b) the **train score** reaches a plateau when `n_estimators` become large (above 500 trees)
- c) the **train score** increases when `n_estimators` become large (above 500 trees)
- d) the **test score** decreases when `n_estimators` become large (above 500 trees)
- e) the **test score** reaches a plateau when `n_estimators` become large (above 500 trees)
- f) the **test score** increases when `n_estimators` become large (above 500 trees)

Select all answers that apply

Repeat the previous experiment but this time, instead of choosing the default parameters for the random forest, set the parameter `max_depth=5` and build the validation curve.

Question

Comparing the validation curve (train and test scores) of the random forest with a full depth and the random forest with a limited depth, select the correct statements.

- a) the **test score** of the random forest with a full depth is (almost) always better than the **test score** of the random forest with a limited depth
- b) the **train score** of the random forest with a full depth is (almost) always better than the **train score** of the random forest with a limited depth
- c) the gap between the train and test scores decreases when reducing the depth of the trees of the random forest
- d) the gap between the train and test scores increases when reducing the depth of the trees of the random forest

Select all answers that apply

Let us now focus at the very beginning of the validation curves, and consider the training score of a random forests with a single tree while using the default `max_depth=None` parameter setting:

```
rf_1_tree = RandomForestRegressor(n_estimators=1, random_state=0)
cv_results_tree = cross_validate(
    rf_1_tree, data, target, cv=10, return_train_score=True
)
cv_results_tree["train_score"]
```

should return:

```
array([0.83120264, 0.83309064, 0.83195043, 0.84834224, 0.85790323,
       0.86235297, 0.84791111, 0.85183089, 0.82241954, 0.85045978])
```

The fact that this single-tree Random Forest can never reach a perfect R2 score of 1.0 on the training can be surprising.

Indeed, if you we evaluate the training accuracy of the single `DecisionTreeRegressor` one gets perfect memorization of the training data:

```
tree = DecisionTreeRegressor(random_state=0)
cv_results_tree = cross_validate(
    tree, data, target, cv=10, return_train_score=True
)
cv_results_tree["train_score"]
```

which outputs the expected perfect score:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Question

From the following statements, select the one that explains that a single-tree random forest cannot achieve perfect training scores.

- a) the single tree in the random forest is trained using a bootstrap of the training set and not the training set itself (because `bootstrap=True` by default)
- b) for a given feature, the single tree in the random forest uses random splits while the single decision tree uses the best split
- c) the random forest automatically limits the depth of the single decision tree, which prevents overfitting

Select a single answer

Build a validation curve for a `sklearn.ensemble.HistGradientBoostingRegressor` varying `max_iter` as follows:

```
max_iter = [1, 2, 5, 10, 20, 50, 100, 200, 500, 1_000]
```

We recall that `max_iter` corresponds to the number of trees in the boosted model.

Plot the average train and test score for each value of `max_iter`.

Question

Select the correct statements.

- a) for a small number of trees (between 5 and 10 trees), the gradient boosting model behave like the random forest algorithm: the train scores are high while the test scores are not optimum
- b) for a small number of trees (between 5 and 10 trees), the gradient boosting model behave differently to the random forest algorithm: both the train and test scores are small
- c) with a large number of trees (> 100 trees) adding more trees in the ensemble causes the gradient boosting model overfit (increasing the gap between the train score and test score)
- d) with a large number of trees (> 100 trees) adding more trees in the ensemble does not impact the generalization performance of the gradient boosting model

Select all answers that apply
