Wrap-up quiz 6

In this wrap-up quiz you will need to write some code in order to answer quiz questions:

- an empty notebook is available just below to write your code
- quiz questions are located after the notebook here
- the button 

at the bottom right of the screen allows you to open the notebook in full page at any time

+ Click here to see a demo video of the notebook user interface

Open Notebook

Module 6 - Wrap-up Quiz

Data Import

In [1]:
```
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]].d
dataset = dataset.sample(frac=1, random_state=0).r
data, target = dataset[feature_names], dataset[tar
```

Baseline Model Evaluation

Decision Tree Regressor

In [2]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_validate

baseline_DTR = DecisionTreeRegressor(random_state=

blDTR_cv_results = cross_validate(baseline_DTR, da
```

Random Forest Regressor

In [4]:
```
from sklearn.ensemble import RandomForestRegressor
```


Powered by OVHcloud

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

```
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 in more 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:

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 1 (1/1 point)

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 ✔

You have used 1 of 1 submissions

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 2 (1/1 point)

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

◉ c) [7, 10]: the random forest model with 100 decision trees is substantially better than the random forest model with 5 decision trees ✔

You have used 1 of 1 submissions

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

```
import numpy as np
n_estimators = np.array([1, 2, 5, 10, 20, 50, 100, 200,
500, 1_000])
```

Question 3 (1/1 point)

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)

You have used 1 of 2 submissions

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 4 (1/1 point)

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
- [x] 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
- [x] 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

You have used 1 of 2 submissions

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:

```
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 5 (1/1 point)

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) ✔

○ c) the random forest automatically limits the depth of the single decision tree, which prevents overfitting

EXPLANATION

solution: a)

By default, random forests train their trees with a bootstrapping procedure. Since each tree is trained on a bootstrap sample, some data points from the original training set are not seen by each individual trees in the forest. As a result the single-tree RF model does not make perfect predictions on those data-points (named out-of-bag samples) which prevents the training score to reach 1.0.

We can confirm this hypothesis by deactivating the bootstrap option and checking again the train scores (leaving all the other hyper-parameters unchanged):

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

Out: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

In this case, we fall back to the same algorithm as the single decision tree by training on the original data and thus overfitting the original training dataset.

We can also note, that when increasing the number of trees to 5 or 10, the sub-optimal training caused by bootstrapping vanishes quickly away. Random Forests are typically trained with at least 10 trees and typically much more than that, so this effect is almost never observed in practice.

Furthermore, even with a single bootstrapped tree, an R2 score larger than 0.85 is still very high and typically much larger than the test score for the same model. As a result, even if the training score of that model is not perfect, one cannot conclude that the

limited by overfitting when there's depth is not bound.

You have used 1 of 1 submissions

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

```
max_iters = np.array([1, 2, 5, 10, 20, 50, 100, 200, 500])
```

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 6 (1/1 point)

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
- [x] 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
- [x] 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

According to you, the 'Wrap-up Quiz' of this module was:

- **Too easy, I got bored**
- **Adapted to my skills**
- **Difficult but I was able to follow**
- **Too difficult**

Submit

To follow this lesson, I spent:

- **less than 30 minutes**
- **30 min to 1 hour**
- **1 to 2 hours**
- **2 to 4 hours**
- **more than 4 hours**
- **I don't know**

Submit

FORUM (EXTERNAL RESOURCE)

There are no more M6. Wrap-up quiz 6 topics. Ready to start a new conversation?

Terms and conditions