Wrap-up quiz 3

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 Open Notebook 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 5 - Wrap-up Quiz

Importing Data

In [1]:
```
import pandas as pd

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

columns = ["Body Mass (g)", "Flipper Length (mm)", "Culm
target_name = "Species"

# Remove lines with missing values for the columns of ir
penguins_non_missing = penguins[columns + [target_name]]

data = penguins_non_missing[columns]
target = penguins_non_missing[target_name]
```

Checking Target

In [3]:
```
target.value_counts(normalize=True)
```

Out[3]:
```
Species
Adelie Penguin (Pygoscelis adeliae)          0.441520
Gentoo penguin (Pygoscelis papua)            0.359649
Chinstrap penguin (Pygoscelis antarctica)    0.198830
Name: proportion, dtype: float64
```

Checking Features

In [4]:
```
data.describe()
```

Out[4]:

	Body Mass (g)	Flipper Length (mm)	Culmen Length (mm)
count	342.000000	342.000000	342.000000
mean	4201.754386	200.915205	43.921930
std	801.954536	14.061714	5.459584

Powered by OVHcloud

Load the dataset file named `penguins.csv` with the following command:

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

columns = ["Body Mass (g)", "Flipper Length (mm)", "Culmen
Length (mm)"]
target_name = "Species"

# Remove lines with missing values for the columns of interest
penguins_non_missing = penguins[columns +
[target_name]].dropna()

data = penguins_non_missing[columns]
target = penguins_non_missing[target_name]
```

`penguins` is a pandas dataframe. The column "Species" contains the target variable. We extract the numerical columns that quantify some attributes of such animals and our goal is try to predict their species based on those attributes stored in the dataframe named `data`.

Inspect the loaded data to select the correct assertions:

Question 1 (1/1 point)

Inspect the target variable and select the correct assertions from the following proposals.

- a) The problem to be solved is a regression problem
- b) The problem to be solved is a binary classification problem (exactly 2 possible classes)
- [✓ c) The problem to be solved is a multiclass classification problem (more than 2 possible classes)] ✓

Hint: `target.nunique()` is a helpful method to answer to this question.

EXPLANATION

solution: c)

We can have a look to the target variable:

```
target.nunique()
```

You have used 1 of 1 submissions

Question 2 (1/1 point)

Inspect the statistics of the target and individual features to select the correct statements.

☐ a) The proportions of the class counts are balanced: there are approximately the same number of rows for each class

☑ b) The proportions of the class counts are imbalanced: some classes have more than twice as many rows than others

☐ c) The input features have similar scales (ranges of values)

✓

Select all answers that apply

Hint: `data.describe()`, and `target.value_counts()` are methods that are helpful to answer to this question.

You have used 1 of 2 submissions

Question 3 (1/1 point)

Let's now consider the following pipeline:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
model = Pipeline(steps=[
    ("preprocessor", StandardScaler()),
    ("classifier", KNeighborsClassifier(n_neighbors=5)),
])
```

Evaluate the pipeline using stratified 10-fold cross-validation with the `balanced-accuracy` scoring metric to choose the correct statement in the list below.

You can use:

- `sklearn.model_selection.cross_validate` to perform the cross-validation routine;

provide the string `balanced_accuracy` to the parameter `scoring` of `cross_validate`.

- [✓ a) The average cross-validated test balanced accuracy of the above pipeline is between 0.9 and 1.0] ✓
- b) The average cross-validated test balanced accuracy of the above pipeline is between 0.8 and 0.9
- c) The average cross-validated test balanced accuracy of the above pipeline is between 0.5 and 0.8

You have used 1 of 1 submissions

Question 4 (1/1 point)

Repeat the evaluation by setting the parameters in order to select the correct statements in the list below. We recall that you can use `model.get_params()` to list the parameters of the pipeline and use `model.set_params(param_name=param_value)` to update them. Remember that one way to compare two models is comparing the cross-validation test scores of both models fold-to-fold, i.e. counting the number of folds where one model has a better test score than the other.

- a) Looking at the individual cross-validation scores, using a model with `n_neighbors=5` is substantially better (at least 7 of the cross-validations scores are better) than a model with `n_neighbors=51`
- [✓ b) Looking at the individual cross-validation scores, using a model with `n_neighbors=5` is substantially better (at least 7 of the cross-validations scores are better) than a model with `n_neighbors=101`] ✓
- [✓ c) Looking at the individual cross-validation scores, a 5 nearest neighbors using a `StandardScaler` is substantially better (at least 7 of the cross-validations scores are better) than a 5 nearest neighbors using the raw features (without scaling).] ✓

✓

EXPLANATION

Solution: b) c)

It is possible to change the pipeline parameters and re-run a cross-validation with:

```
model.set_params(preprocessor=StandardScaler(),
classifier__n_neighbors=51)
cv_results_ss_51 = cross_validate(
    model, data, target, cv=10, scoring="balanced_accuracy"
)
cv_results_ss_51["test_score"].mean(),
cv_results_ss_51["test_score"].std()
```

which gives slightly worse test scores but the difference is not necessarily significant: they overlap a lot. So given the definition of **better**, we can check the individual score for each fold and count how many times the 5-NN classifier is better than the 51-NN classifier. With some python code (you could have do it by visualizing the `"test_score"` columns as well), we obtain:

```
print(
    "5-NN is strictly better than 51-NN for "
    f"{sum(cv_results_ss_5['test_score'] > cv_results_ss_51['test_score'])}"
    " CV iterations out of 10."
  )
```

Here, 5-NN is strictly better than 51-NN only 4 times and thus we cannot conclude that it is substantially better.

We can repeat the same experiment for a 101-NN:

```
model.set_params(preprocessor=StandardScaler(),
classifier__n_neighbors=101)
cv_results_ss_101 = cross_validate(
    model, data, target, cv=10, scoring="balanced_accuracy"
)
cv_results_ss_101["test_score"].mean(),
cv_results_ss_101["test_score"].std()
```

We observe that the average test accuracy of this last model seems to be substantially lower that the previous models. Let's check the number of CV folds where this is actually the case:

```
        "CV iterations out of 10."
    )
```

In this case, we observe that 5-NN is always better.

We can disable the preprocessor by setting the `preprocessor` parameter to `None` (while resetting the number of neighbors to 5) as follows:

```
model.set_params(preprocessor=None, classifier__n_neighbors=5)
cv_results_none_5 = cross_validate(
    model, data, target, cv=10, scoring="balanced_accuracy"
)
cv_results_none_5["test_score"].mean(),
cv_results_none_5["test_score"].std()
```

This gives results with a mean balanced accuracy of ~0.74 which is much worse than the same result with preprocessing enabled. We can confirm that preprocessing the dataset lead to a substantially better model:

```
print(
    "NN with scaling is better NN without scaling for "
    f"{sum(cv_results_ss_5['test_score'] > cv_results_none_5['test_score'])}"
    "CV iterations out of 10."
  )
```

Here, the model with feature scaling is performing better 10 times over 10 than the model that does not preprocess the dataset.

You have used 2 of 2 submissions

Question 5 (1/1 point)

We will now study the impact of different preprocessors defined in the list below:

```
from sklearn.preprocessing import PowerTransformer


all_preprocessors = [
    None,
    StandardScaler(),
    MinMaxScaler(),
    QuantileTransformer(n_quantiles=100),
    PowerTransformer(method="box-cox"),
]
```

The Box-Cox method is common preprocessing strategy for positive values. The other preprocessors work both for any kind of numerical features. If you are curious to read the details about those method, please feel free to read them up in the preprocessing chapter of the scikit-learn user guide but this is not required to answer the quiz questions.

Use `sklearn.model_selection.GridSearchCV` to study the impact of the choice of the preprocessor and the number of neighbors on the stratified 10-fold cross-validated `balanced_accuracy` metric. We want to study the `n_neighbors` in the range `[5, 51, 101]` and `preprocessor` in the range `all_preprocessors`.

Which of the following statements hold:

- [ ] a) Looking at the individual cross-validation scores, the best ranked model using a `StandardScaler` is substantially better (at least 7 of the cross-validations scores are better) than using any other preprocessor
- [✓] b) Using any of the preprocessors has always a better ranking than using no preprocessor, irrespective of the value `of n_neighbors`
- [ ] c) Looking at the individual cross-validation scores, the model with `n_neighbors=5` and `StandardScaler` is substantially better (at least 7 of the cross-validations scores are better) than the model with `n_neighbors=51` and `StandardScaler`
- [✓] d) Looking at the individual cross-validation scores, the model with `n_neighbors=51` and `StandardScaler` is substantially better (at least 7 of the cross-validations scores are better) than the model with `n_neighbors=101` and `StandardScaler`

Hint: pass `{"preprocessor": all_preprocessors, "classifier__n_neighbors": [5, 51, 101]` for the `param_grid` argument to the `GridSearchCV` class.

You have used 1 of 2 submissions

Question 6 (1/1 point)

Evaluate the generalization performance of the best models found in each fold using nested cross-validation. Set `return_estimator=True` and `cv=10` for the outer loop. The scoring metric must be the `balanced-accuracy`.

The mean generalization performance is :

- ○ a) better than 0.97
- ● b) between 0.92 and 0.97
- ○ c) below 0.92

You have used 1 of 1 submissions

Question 7 (1/1 point)

Explore the set of best parameters that the different grid search models found in each fold of the outer cross-validation. Remember that you can access them with the `best_params_` attribute of the estimator. Select all the statements that are true.

- [✓] a) The tuned number of nearest neighbors is stable across folds
- [ ] b) The tuned number of nearest neighbors changes often across folds
- [ ] c) The optimal scaler is stable across folds
- [✓] d) The optimal scaler changes often across folds

✓

field of the CV results. If you forgot to do for the previous question, please re-run the cross-validation with that option enabled.

You have used 1 of 2 submissions

YOUR EXPERIENCE

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 answer this wrap-up quiz, 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)

New Topic

Topic	Replies	Last reply
About the M3. Wrap-up quiz 3 category brospars 27 Oct	1	2d javi12

There are no more M3. Wrap-up quiz 3 topics. Ready to start a new conversation?