Wrap-up quiz 2

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 ↗

Wrap-Up Quiz 2

Importing Pandas

```
In [2]:   import pandas as pd
```

Loading Data

```
In [11]:  blood_transfusion = pd.read_csv("../datasets/blood
          target_name = "Class"
          data = blood_transfusion.drop(columns=target_name)
          target = blood_transfusion[target_name]
          # Checking data
          data.head()
```

Out[11]:

	Recency	Frequency	Monetary	Time
0	2	50	12500	98
1	0	13	3250	28
2	1	16	4000	35
3	2	20	5000	45
4	1	24	6000	77

Checking target type and imbalance

```
In [6]:   target.value_counts()
```

```
Out[6]:   Class
          not donated    570
          donated        178
          Name: count, dtype: int64
```

Open the dataset `blood_transfusion.csv` with the following command:

```
blood_transfusion =
pd.read_csv("../datasets/blood_transfusion.csv")
target_name = "Class"
data = blood_transfusion.drop(columns=target_name)
target = blood_transfusion[target_name]
```

`blood_transfusion` is a pandas dataframe. The column "Class" contains the target variable.

Question 1 (1/1 point)

Select the correct answers 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)

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

✔

Select all answers that apply

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

EXPLANATION

solution: b) d)

We can have a look to the target variable:

```
target.head()
```

have a binary or a multiclass classification problem.

```
target.unique()
```

We see that the problem is a binary classification with 2 classes, namely `donated` and `not donated`. Now, we should check the number of samples in each of the class to know the ratio of the class counts.

```
target.value_counts(normalize=True)
```

~75% of the samples belong to the class `not donated` while only ~25% of the samples belong to the class `donated`.

You have used 1 of 2 submissions

Question 2 (1/1 point)

Using a `sklearn.dummy.DummyClassifier` and the strategy `"most_frequent"`, what is the average of the accuracy scores obtained by performing a 10-fold cross-validation?

○ a) ~25%

○ b) ~50%

● c) ~75% ✔

Hint: You can check the documentation of `sklearn.model_selection.cross_val_score` here and `sklearn.model_selection.cross_validate` here.

You have used 1 of 1 submissions

Question 3 (1/1 point)

when calling `cross_validate` or `cross_val_score` functions, the mean score is:

- ○ a) ~25%

- ● b) ~50% ✔

- ○ c) ~75%

EXPLANATION

solution: b)

The code to compute the balanced accuracy instead of the accuracy is the following:

```
dummy = DummyClassifier(strategy="most_frequent")
scores = cross_val_score(
    dummy, data, target, cv=10, scoring="balanced_accuracy"
)
scores.mean()
```

In this case, the score is corrected such that such a dummy approach will give a score of 50%. You can learn more about how balanced_accuracy is computed in the scikit-learn documentation. Therefore `balanced_accuracy` makes it easier to distinguish good from bad classifiers on imbalanced classification problems.

You have used 1 of 1 submissions

Question 4 (1/1 point)

We will use a `sklearn.neighbors.KNeighborsClassifier` for the remainder of this quiz.

Why is it relevant to add a preprocessing step to scale the data using a `StandardScaler` when working with a `KNeighborsClassifier` ?

○ b) k-nearest neighbors is based on computing some distances. Features need to be normalized to contribute approximately equally to the distance computation. ✔

○ c) This is irrelevant. One could use k-nearest neighbors without normalizing the dataset and get a very similar cross-validation score.

EXPLANATION

solution: b)

Computing distances on scaled or un-scaled data takes similar time but the resulting distance values could be very different and therefore ordering of which sample is a closest neighbor to which sample also.
For instance: if you have two variables A and B, where A has values which vary between 0 and 1000000 (e.g. the price of a house in euros) and B is a variable that varies between 0 and 30 (e.g. the average outside air temperature in Celsius degrees), then computing distances between rows of such a database will be mostly impacted by the differences in values of the A column and the values of B column will be comparatively ignored.
If one applies `StandardScaler` to such a database, both the values of A and B will be approximately between -3 and 3 and the neighbor structure will be impacted by both variables.

You have used 1 of 1 submissions
Question 5 (1/1 point)

Create a scikit-learn pipeline (using `sklearn.pipeline.make_pipeline`) where a `StandardScaler` will be used to scale the data followed by a `KNeighborsClassifier`. Use the default hyperparameters.

- ○ a) 1

- ○ b) 3

- ● c) 5 ✔

- ○ d) 8

- ○ e) 10

Hint: You can use `model.get_params()` to get the parameters of a scikit-learn estimator.

You have used 1 of 1 submissions

Question 6 (1/1 point)

Set `n_neighbors=1` in the previous model and evaluate it using a 10-fold cross-validation. Use the balanced accuracy as a score. What can you say about this model? Compare the average of the train and test scores to argument your answer.

- ○ a) The model clearly underfits

- ○ b) The model generalizes

- ● c) The model clearly overfits ✔

Hint: compute the average test score and the average train score and compare them. Make sure to pass `return_train_score=True` to the `cross_validate` function to also compute the train score.

The code to perform the cross-validation is the following:

```
from sklearn.model_selection import cross_validate

model = make_pipeline(StandardScaler(),
KNeighborsClassifier(n_neighbors=1))

cv_results = cross_validate(
    model, data, target, cv=10, scoring="balanced_accuracy",
    return_train_score=True,
)
cv_results = pd.DataFrame(cv_results)
cv_results[["train_score", "test_score"]].mean()
```

We see that the gap between train and test scores is large. In addition, the average score on the training sets is good while the average scores on the testing sets are really bad. Those are the signs of an overfitting model.

You have used 1 of 1 submissions

Question 7 (1/1 point)

We now study the effect of the parameter `n_neighbors` on the train and test score using a validation curve. You can use the following parameter range:

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

Also, use a 5-fold cross-validation and compute the balanced accuracy score instead of the default accuracy score (check the `scoring` parameter). Finally, plot the average train and test scores for the different value of the hyperparameter. We recall that the name of the parameter can be found using `model.get_params()`.

Select the true affirmations stated below:

○ a) The model underfits for a range of `n_neighbors` values

○   b) The model underfits for a range of `n_neighbors` values between 10 to 100

●   c) The model underfits for a range of `n_neighbors` values between 100 to 500 ✔

EXPLANATION

solution: c)

The code to compute and plot the validation curve is the following:

```python
import numpy as np
from sklearn.model_selection import ValidationCurveDisplay

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

disp = ValidationCurveDisplay.from_estimator(
    model,
    data,
    target,
    param_name="kneighborsclassifier__n_neighbors",
    param_range=param_range,
    scoring="balanced_accuracy",
    std_display_style="errorbar",
    n_jobs=2,
)

_ = disp.ax_.set(
    xlabel="Value of hyperparameter n_neighbors",
    ylabel="Balanced accuracy score",
    title="Validation curve of K-nearest neighbors",
)
```

Underfitting happens when the gap between train and test scores is low and that both scores are low. When `n_neighbors` is high, underfitting occurs. The model lacks expressivity because it always considers the majority class of a large number of data points. Its prediction tends to be always the same, irrespective of the test point of interest.

Question 8 (1/1 point)

Select the most correct of the affirmations stated below:

○ a) The model overfits for a range of `n_neighbors` values between 1 to 10 ✔

○ b) The model overfits for a range of `n_neighbors` values between 10 to 100

○ c) The model overfits for a range of `n_neighbors` values between 100 to 500

EXPLANATION

solution: a)

Overfitting happens when the gap between train and test scores is high and that the test score is low. When `n_neighbors` is low, the model looks at small number of data points around the test points of interest. In this case the model is too flexible: it is sensible to noise rather than the underlying structure of the data.

You have used 1 of 1 submissions

Question 9 (1/1 point)

Select the most correct of the affirmations stated below:

○ a) The model best generalizes for a range of `n_neighbors` values between 1 to 10

○   c) The model best generalizes for a range of `n_neighbors`
values between 100 to 500

EXPLANATION

solution: b)

Generalization happens when the gap between train and test
scores is relatively low and that the test scores is maximum. In our
case, the range between 20 to 50 neighbors seems to be the best.
Indeed, for 20 neighbors, the gap between train and test scores is
larger than with 50 neighbors; however the test score is better with
20 neighbors.

Considering an intermediate value for n_neighbors makes it
possible to limit overfitting by averaging out the influence of a few
noisy samples. On the other hand, the model can also avoid
underfitting by focusing its decision on the details of a region of
interest close to the test sample. The optimum value for
`n_neighbors` is typically very dataset-dependent and can be
influenced by the choice of the distance metric, the scaling of the
features, the presence of noisy samples, the balance between
classes, etc.

Finally notice that, while k-nearest neighbors is useful baseline, it
can be too slow to predict on large datasets with many samples
and features.
Practitioners would instead favor other models such as gradient
boosted trees for instance. On the particular dataset used for this
quiz, the best accuracy achieved by the nearest neighbors model
does not seem to be high enough to make it useful in practice.

You have used 1 of 1 submissions
YOUR EXPERIENCE
According to you, the 'Wrap-up Quiz' of this module was:

○ **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)

Topic	Replies	Last reply
📌 About the M2. Wrap-up quiz 2 category **brospars** 27 Oct	0	**27 Oct** **brospars**

There are no more M2. Wrap-up quiz 2 topics. Ready to start a new conversation?

Terms and conditions