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

```
import pandas as pd



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
b) The problem to be solved is a binary classification problem (exactly 2 possible classes), d) The proportions of the class counts are imbalanced: some classes have more than twice as many rows than others, - correct
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()
```

Since the target has categorical values, it is a classification problem. We can check how many classes are present to know if we 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`.

Hide Answer

You have used 2 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% c) ~75% - correct

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

EXPLANATION

solution: c)

The code below allows to compute the average of the accuracy scores obtained by performing a 10-fold cross-validation.

```python
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import cross_val_score

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

This is not a surprise to get a score a high as ~75%. This is due to the class imbalanced nature of the dataset: always predicting `not donated` class will be correct ~75% of the time since this the natural proportion of samples belonging to this class in the data.

Hide Answer

You have used 1 of 1 submissions

Question 3

(1/1 point)

Repeat the previous experiment but compute the balanced accuracy instead of the accuracy score. Pass `scoring="balanced_accuracy"` when calling `cross_validate` or `cross_val_score` functions, the mean score is:

a) ~25% b) ~50% b) ~50% - correct 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.

Hide Answer

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`?

a) faster to compute the list of neighbors on scaled data b) k-nearest neighbors is based on computing some distances. Features need to be normalized to contribute approximately equally to the distance computation. b) k-nearest neighbors is based on computing some distances. Features need to be normalized to contribute approximately equally to the distance computation. - correct 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.

Hide Answer

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. Inspect the parameters of the created pipeline. What is the value of K, the number of neighbors considered when predicting with the k-nearest neighbors?

a) 1 b) 3 c) 5 c) 5 - correct d) 8 e) 10

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

EXPLANATION

solution: c)

The code below allow to list all parameters that can be set in our scikit-learn pipeline.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

model = make_pipeline(StandardScaler(), KNeighborsClassifier())
model.get_params()
```

An entry in the returned dictionary should be:

```
'kneighborsclassifier__n_neighbors': 5
```

In this case, the number of neighbors K is 5. This means that when the model has to predict the class of a given test sample, it will compute the distances between that test sample and all the samples of the training set, order them by ascending distances and return the class of the majority of the 5 closest neighbors.

Hide Answer

You have used 1 of 1 submissions

Question 6

(1 point possible)
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 a) The model clearly underfits - incorrect 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.

EXPLANATION

solution: c)

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.

Hide Answer

You have used 1 of 1 submissions

Question 7

(1 point possible)

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 between 1 to 10 a) The model underfits for a range of <code>n_neighbors</code> values between 1 to 10 - incorrect 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:

```
import numpy as np

from sklearn.model_selection import ValidationCurveDisplay
```

```python
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.

Hide Answer

You have used 1 of 1 submissions

Question 8

(1 point possible)
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 c) The model overfits for a range of <code>n_neighbors</code> values between 100 to 500 - incorrect

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.

Hide Answer
You have used 1 of 1 submissions

Question 9

(1 point possible)
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 a) The model best generalizes for a range of <code>n_neighbors</code> values between 1 to 10 - incorrect b) The model best generalizes for a range of `n_neighbors` values between 10 to 100 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.