✅ Quiz M3.02

Question

What does `CV` stand for in `GridSearchCV` and why?

- a) cross-validation: once we found the best parameters we estimate the model performance through cross-validation on the full data
- b) circular values: we do a permutation of all the possible parameter value combinations
- c) cross-validation: the score of each combination of parameters on the grid is computed by using an internal cross-validation procedure
- d) contribution value: we estimate how much each parameter contributes to the model generalization performance

Select a single answer

Question

Given `pipeline` defined by:

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression())
])
```

We want to find the best `C` through a grid-search where `C` takes the values 0.1, 1, and 10:

```
param_grid = ...  # complete this line in your answer
model = GridSearchCV(
    pipeline,
    param_grid=param_grid
).fit(X, y)
model.best_params_
```

How should the `param_grid` variable be defined:

- a) `param_grid = {'logisticregression__C': [0.1, 1, 10]}`
- b) `param_grid = {'classifier__C': [0.1, 1, 10]}`
- c) `param_grid = {'classifier__C': 0.1, 'classifier__C': 1, 'classifier__C': 10}`
- d) `param_grid = {'C': [0.1, 1, 10]}`

Select a single answer

Question

Select the true statements about `RandomizedSearchCV` and `GridSearchCV` below:

- a) `RandomizedSearchCV` has a fixed computation budget through its `n_iter` parameter
- b) `RandomizedSearchCV` allows to test all the combinations of a fixed set of parameter values
- c) `GridSearchCV` can become very computationally intensive when the number of parameters grows
- d) both `GridSearchCV` and `RandomizedSearchCV` have the attributes `cv_results_` and `best_params_`
- e) both `GridSearchCV` and `RandomizedSearchCV` can use probability distributions to draw parameter values from

Select all answers that apply

Copy and execute the following code in the sandbox notebook to load the results of the randomized-search performed in the previous notebook. Executing this code will display an interactive plot to analyze the impact of the hyper-parameters on the test score of the models.

```
import numpy as np
import pandas as pd
import plotly.express as px


def shorten_param(param_name):
    if "__" in param_name:
        return param_name.rsplit("__", 1)[1]
    return param_name


cv_results = pd.read_csv("../figures/randomized_search_results.csv",
                         index_col=0)

fig = px.parallel_coordinates(
    cv_results.rename(shorten_param, axis=1).apply({
        "learning_rate": np.log10,
        "max_leaf_nodes": np.log2,
        "max_bins": np.log2,
        "min_samples_leaf": np.log10,
        "l2_regularization": np.log10,
        "mean_test_score": lambda x: x}),
    color="mean_test_score",
    color_continuous_scale=px.colors.sequential.Viridis,
)
fig.show()
```

Note

We **transformed most axis values by taking a log10 or log2** to spread the active ranges and improve the readability of the plot.

Question

In the parallel coordinate plot obtained by the running the above code snippet, select the bad performing models.

We define bad performing models as the models with a `mean_test_score` below 0.8. You can select the range [0.0, 0.8] by clicking and holding on the `mean_test_score` axis of the parallel coordinate plot.

Looking at this plot, which parameter values always cause the model to perform badly?

- a) too large `l2_regularization`
- b) too small `l2_regularization`
- c) too large `learning_rate`
- d) too small `learning_rate`
- e) too large `max_bins`
- f) too small `max_bins`

Select all answers that apply

Question

In the parallel coordinate plot shown above, select the models with a score higher than 0.85. You can select the range [0.85, max] by clicking and holding on the `mean_test_score` axis of the parallel coordinate plot.

Identify ranges of values for hyperparameters that always prevent the model to reach a test score higher than 0.85, irrespective of the other values. In other words, which hyperparameters values are never used to get a good model (i.e. with `mean_test_score` higher than 0.85).

- a) too large `l2_regularization`
- b) too small `l2_regularization`
- c) too large `learning_rate`
- d) too low `learning_rate`
- e) too large `max_bins`
- f) too low `max_bins`

Select all answers that apply
