- Welcome
- Introduction: Machine Learning concepts
- Module 1. The Predictive Modeling Pipeline
- Module 2. Selecting the best model
- **Module 3. Hyperparameter tuning**
  - Module overview
  - Manual tuning
    Quiz M3
  - Automated tuning
    Quiz M3
  - Wrap-up quiz
    Wrap-up quiz
  - Main take-away
- Module 4. Linear Models
- Module 5. Decision tree models
- Module 6. Ensemble of models

Quiz M3.02

Note: For each question **make sure you select all of the correct options**—there may be more than one! Don't forget to use the sandbox notebook if you need.

Question 1 (1/1 point)

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
- [x] 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

EXPLANATION

Solution: c)

When calling `fit(X, y)` on a grid-search, `X` and `y` will be split by a cross-validation strategy. For instance, if a 10-fold cross-validation was chosen, `X` and `y` are divided into 10 folds and 10 models will be trained on 9 folds and tested on the remaining fold. Each model will have used different test fold. The test scores are then averaged for the 10 models.

This operation is repeated for all combinations of hyperparameters. The combination of hyperparameter values with the best average cross-validation score is selected.

You have used 1 of 1 submissions

performance

- Conclusion
- Appendix

```
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]}`

You have used 1 of 1 submissions

Question 3 (1/1 point)

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

- [ ] b) `RandomizedSearchCV` allows to test all the combinations of a fixed set of parameter values
- [x] c) `GridSearchCV` can become very computationally intensive when the number of parameters grows
- [x] 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

EXPLANATION

Solution: a) c) d)

b) is incorrect: the statement is true for `GridSearchCV` but not for `RandomizedSearchCV`.

e) is incorrect: only `RandomizedSearchCV` can use probability distributions to draw parameter values from. `GridSearchCV` always perform an exhaustive evaluation of all the possible combinations of parameter values.

You have used 1 of 2 submissions

Question 4 (1/1 point)

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 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()
```

We **transformed most axis values by taking a log10 or log2** to spread the active ranges and improve the readability of the plot.
In the parallel coordinate plot obtained by the running the above code snippet, 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`
- [x] c) too large `learning_rate`
- [x] d) too low `learning_rate`
- [ ] e) too large `max_bins`

✔

Select several answers

You have used 1 of 2 submissions

Question 5 (1/1 point)

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`
- [x] c) too large `learning_rate`
- [x] d) too small `learning_rate`
- [ ] e) too large `max_bins`
- [ ] f) too small `max_bins`

✔

Select several answers

You have used 1 of 2 submissions

YOUR EXPERIENCE

According to you, this whole 'Automated tuning' lesson was:

- ○ **Too easy, I got bored**
- ○ **Adapted to my skills**

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)

Topic	Replies
About the M3. Exercise M3.02 category brospars 27 Oct	3

There are no more M3. Exercise M3.02 topics. Ready to start a new conv