
LAB N° 3 : Pandas, sklearn and hyperparameter tuning

Reminder : All labs are done by pairs. Your submission file should be named `firstname1_LASTNAME1_firstname2_LASTNAME2.ipynb` (e.g. `Badr_MOUFAD_Mathurin_MASSIAS.ipynb`).

Write the question number in each corresponding notebook cell..

- TRAINING A MODEL AND TUNING ITS HYPERPARAMETER -

In the first part, we consider a dataset where the goal is to predict the number of passengers on a given flight.

- 1) What kind of problem is it ? Regression or classification ? Supervised or unsupervised ?
- 2) Load the training data from Moodle (`train.csv.bz2`; `bz2` is a compression format, pandas can decompress it itself). The target variable is called `log_PAX`. Do a quick inspection of the dataset. What are the types of the columns ?
- 3) Convert dates to proper dates. Create new integers columns containing respectively the day (day of the month : from 1 to 31), the weekday (day of the week : from 1 to 7), the week, the month, the year, a binary variable indicating if this is a bank holiday (in the US calendar), a binary variable indicating if this is the weekend or not.

In the following block, use only numerical features.

- 4) First, select numerical features in an automated fashion (not by hand). You can for example use a list comprehension, or `df.select_dtypes`.
- 5) We will use the Root Mean Squared Error (RMSE) as a figure of merit (performance measure) for this prediction task. Explain how it is defined and why it is relevant here.
- 6) Do a train-test split of the data (a single one, so far. You'll do K -fold cross validation later) and tune the `max_depth` parameter of a `DecisionTreeRegressor`. Explain briefly how this estimator does its prediction. Plot the RMSE on train and test sets as a function of this parameter.
- 7) Test the impact of using or not a `StandardScaler` on the features, for this estimator with the found value of `max_depth` (use a `Pipeline`). Explain the results.
- 8) For a `LinearRegression` model with `fit_intercept=True`, test the impact of using a `StandardScaler`. Explain.

Now, we use again all features. We will encode the categorical features with a `OneHotEncoder`

- 9) Create a one hot encoder instance, fit it on the data, transform the data and display all categories inferred by the transformer. Delete the transformed data.
- 10) Create a `Pipeline` standardizing the numerical features, and one-hot encoding categorical features, followed by the application of a `RandomForestRegressor` to the transformed data.
- 11) Perform grid-search on the cross-validation error to tune simultaneously the `n_estimators` and `max_depth` of the prediction step of your pipeline. Comment on the execution time.
- 12) Get the estimator with the best params. Save both the full pipeline and the best model to disk with `joblib`. Load them from disk. Why is the ability to dump estimators useful ?

K-nearest neighbors We now move to simulated data and a different estimator, K -nearest neighbors (KNN). K -nearest neighbors is an algorithm for classification that computes the K -nearest neighbors of a point

$$V(x) = \{i \in [1, n], \|x_i - x\| \text{ amongst } K \text{ smallest values}\}$$

and uses as prediction for x , the most represented class in the set $\{y_i, i \in V(x)\}$.

- 13) What is the cost of fitting a KNN ? and of predicting for one new point ?

- 14) Implement a `KNearestNeighbor` class with `__init__`, `fit` and `predict`. `scipy.stats.mode` may be useful for prediction.
- 15) Generate data with the function `rand_checkers` on Moodle. Describe the data.
- 16) Use 10 fold cross validation to tune the parameter K of your estimator on this dataset (it may help to have your class inherit from `BaseEstimator` and `ClassifierMixin`, that can be imported from `sklearn.base`). Plot the average scores on the train and test sets as a function of K . Comment.

- ENCODING AND HYPERPARAMETER TUNING WITH OPTUNA -

- 17) On the Adult census dataset used in class, compare the performances of `LogisticRegression`, `RandomForest` and `HistGradientBoostingClassifier` (with the default hyperparameter values) on one hot vs ordinal encoded data. How does the chosen encoding affects each model?
- 18) Use `optuna` to tune the learning rate of a `HistogramGradientBoostClassifier`. Compare to the performance of the other two models.

- PROCESSING FUZZY CATEGORICAL DATA -

- 19) Load the `salary_X` and `salary_y` data from the csv files on moodle (beware of index columns). What's this dataset about?
- 20) How many distinct modalities are there per column of `X`? When using a One Hot Encoder, which columns may cause issues?
- 21) Inspect the column `employee_position_title`. Is there a natural notion of distance on those modalities? Are the modalities completely unordered? Which encoding would you use?

To tackle this problem of fuzzy labelling, we will use the `GapEncoder` from the `skrub` package ([documentation](#)). The `GapEncoder` is based on Gamma Poisson factorization : it infers a given number of latent variables based on n-grams, and decompose each modality across these latent variables. Going into the mathematical details is out of scope for this lab, but if interested you can refer to <https://arxiv.org/abs/1907.01860>.

- 22) Detail the output of a `GapEncoder` when used on the following data :

```
X = [{"Math, optimization"}, {"mathematics"}, {"maths, ml"}, {"ml.maths"},
     {"machine learning"}, {"physics"}, {"phy"},
     {"statistical physics"}, {"computational phys."}]
```

Compare the output of the trained encoder on clean and dirty modalities, eg `["physics"]` vs `["phycis"]`. Is the behavior you observe a good thing or a bad thing? What does the `n_components` represent? Print the learned components.

- 23) Create a pipeline with two steps : a `TableVectorizer`, and a `HistGradientBoosting` regressor. Fit it on the full `X` and `y`. Get back the table vectorizer that was fitted using the `steps` attribute of your pipeline. What did it do here?