X/HEC Datascience for Business Master
2022/2023 **Mathurin Massias, Sylvain Combettes**

Lab n° 3 : Pandas, sklearn and hyperparameter tuning

- 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 of 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, the week, the month, the year, a binary variable indicating if this is a work day or holiday (in the US calendar).

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 validaiton 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 the full dataset. 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 neighbor. 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 predictin 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 having your class inherit from `BaseEstimator` and `ClassifierMixin`, that can be imported from `sklearn.base`). Plot the average loss on the train and test sets as a function of K. Comment.