In this task, as an illustration of a real-world problem, you are asked to predict the electricity prices in Switzerland given price information of some other countries and additional features. You will encounter typical ML workflow challenges of missing features and low predictivity in this task.

The following sections provide more details on the dataset, submission and evaluation.

**Data description**

In the handout for this project, you will find the the following files:

* **train.csv** - the training set
* **test.csv** - the test set file to make predictions on
* **sample.csv** - a sample submission file in the correct format
* **template\_solution.py** - a template file that will guide you through the implementation of the solution
* **template\_solution.ipynb** - a template file in jupyter notebook format that will guide you through the implementation of the solution

You are free to use either jupyter notebook or the .py template file.

Each line in **train.csv** represents one data point and consists of the following structure:

season,price\_AUS,price\_CHF,price\_CZE,price\_GER,price\_ESP,price\_FRA,price\_UK,price\_ITA,price\_POL,price\_SVK

spring,,9.644027877268496,-1.6862480951361345,-1.7480763846576997,-3.666005401185653,,-1.822719793809122,-3.931031226630509,,-3.238196806151894  
...

As mentioned in the introduction, we are interested in predicting the electricity (log) prices in Switzerland (corresponds to price\_CHF column), given the prices in some other countries (corresponds to all columns starting with "price\_" except price\_CHF), and additional features (corresponds to season). Note that there might be missing values in the price\_CHF column as well.

**test.csv** is the file you submit predictions on. The file consists of the following structure:

season,price\_AUS,price\_CZE,price\_GER,price\_ESP,price\_FRA,price\_UK,price\_ITA,price\_POL,price\_SVK

spring,,0.4729846640395274,0.7079565159369821,,-1.1364407652408537,-0.5967028557351153,,3.298693142239604,1.9218859561660853  
...

For your convenience, a sample solution file **sample.csv** is provided with the following structure:

price\_CHF  
0.635389524381449  
0.635389524381449  
0.635389524381449  
0.635389524381449

**template\_solution.py** provides a starting template structure for how you can solve the task, by filling in the TODOs in the skeleton code. It is not mandatory to use this solution template but it is recommended since it should make getting started on the task easier.

**Modelling Tips**

**Data Imputation**

Missing data is a commonly encountered artifact in several machine learning tasks. Typical imputation strategies to deal with missing data include:

* Discarding rows or columns with missing data
* Replacing missing values with the corresponding mean or median
* Advanced imputation strategies based on iterative model fitting.

Additional resources on data imputation can be found in this [kaggle notebook](https://www.kaggle.com/code/residentmario/simple-techniques-for-missing-data-imputation/notebook) or [sklearn website](https://scikit-learn.org/stable/modules/impute.html).

**Handling Categorical (Non-Numeric) Data**

Some data in this task is categorical (non-numeric). This is a common challenge in machine learning. Some strategies on how to handle categorical data can be found in this [kaggle notebook](https://www.kaggle.com/alexisbcook/categorical-variables) or [sklearn website](https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features).

**Kernelized Regression Models**

You saw kernelized estimators in the lecture notes. In this task you might find them useful. The core of the challenge is to pick the right kernel for the regression. Commonly used kernels are the linear (or dot product) kernel, squared exponential (or RBF) kernel, polynomial, Matern, and RationalQuadratic kernels among many others. A probabilistic (Bayesian) equivalent of kernelized ridge regression goes by the name Gaussian processes. It provides a principled modelling paradigm with uncertainty estimates. The uncertainty component is not important for this task, however a lot of machine learning software packages implement this method in a very efficient manner and its mean prediction does the same as kernelized ridge regression. The following code block gets you started with gaussian processes in sklearn (more resources can be found [here](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.gaussian_process))

from sklearn.gaussian\_process import GaussianProcessRegressor

from sklearn.gaussian\_process.kernels import DotProduct, RBF, Matern, RationalQuadratic

gpr = GaussianProcessRegressor(kernel=DotProduct())

gpr.fit(X\_train, y\_train)

Finding the right kernel can be done in multiple ways as you saw in the lectures:

* Using a validation set
* Cross-validation
* Maximizing the evidence of a Bayesian model. In this case, we are maximizing the total probability of the data given its generative model. This is also referred to as Bayesian model selection. More information can be found here: [Bayesian model selection - lecture notes.](https://www.cse.wustl.edu/~garnett/cse515t/fall_2019/files/lecture_notes/7.pdf)

All the above methods are implemented in scikit-learn. Note that Gaussian processes implement Bayesian model selection automatically.

A white screen with black text

Description automatically generatedA white background with black text

Description automatically generated