Multi-Label Classification Strategies

In this task you deal with multiclass classification problem for Glass Classification Data. Lets load the dataset.

In [ ]:
```
# If you are using colab, uncomment this cell

# ! wget https://raw.githubusercontent.com/girafe-ai/ml-mipt/basic/week04_svm_pca/d

# ! mkdir data

# ! mv glass.csv data
```

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv('data/glass.csv')
X, y = data.drop('Type', axis=1), data.Type
data.sample(3)
```

Out[2]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
36	1.51909	13.89	3.53	1.32	71.81	0.51	8.78	0.11	0.00	1
191	1.51602	14.85	0.00	2.38	73.28	0.00	8.76	0.64	0.09	7
175	1.52119	12.97	0.33	1.51	73.39	0.13	11.27	0.00	0.28	5

In [3]:
```
y.value_counts()
```

Out[3]:
```
2    76
1    70
7    29
3    17
5    13
6     9
Name: Type, dtype: int64
```

The features of each glass oject correspond to the fraction of the particular chemical element in the object. The target variable corresponds to the type of glass (6 classes).

In this problem you have to empirically compare the time complexity and performance of several multiclass labeling strategies for different algorithms. Consider the following algorithms:

- KNearestNeighbors (5 neighbors)
- Logistic Regression
- SVC [Support Vector Classification] (linear kernel)

Note that all these algorithms by default support **multiclass labeling**. Nevertheless, compare this approach with **OneVSRest** and **OneVSOne** approaches applied to this algorithms. More precisely, for every pair (algorithm, approach) perform 5-fold cross validation on the data and output the validation score and the computation time (in the table form).

Note that dataset is both multiclass and imbalanced (we will give some points on how to deal with imbalanced datasets later) thus it's important to choose proper quality estimation. Try different metrics to optimize during CV (e.g. accuracy, balanced accuracy, f1, roc-auc).

After that, answer to the following questions:

- Which metric would you choose to optimize during cross validation and why?
- For which algorithms the usage of OneVSRest/OneVSOne approach provides significantly better performance without significant increase in computation time?

In [4]:
```
# proper way to measure performance in modern Python! No time.time()!
# see https://docs.python.org/3/library/time.html#time.perf_counter
from time import perf_counter

# funct to properly display numpy arrays inline, use instead of print
from IPython.display import display

from sklearn.multiclass import OneVsOneClassifier, OneVsRestClassifier
```

In [5]:
```
# your code here
```