Online Learning

learning from nonstationary time series

Antonio Carta

antonio.carta@unipi.it

Online Learning Module - Outline

Streams and Concept Drift

- Streaming data
- Prequential Evaluation
- Concept Drift
- Time series analysis

Online Learning Algorithms

- Online classification models
- Ensemble methods

Online Learning Module – Resources

- Streaming Data Analytics Course by Emanuele della Valle
  - Some slides of this module are based on this course
  - http://emanueledellavalle.org/teaching/streaming-data-analytics-2022-23/
  - https://github.com/emanueledellavalle/streaming-data-analytics
- Book: Machine Learning for Data Streams
  - HTML book: https://moa.cms.waikato.ac.nz/book-html/
- River – SML library in python https://riverml.xyz/0.13.0/

Lecture Outline

- Definition of online learning and applications
- Batch vs streaming/online learning
- Online training and evaluation
- Requirements and motivation for online learning

What is Online Learning?

- **Data is generated continuously**
  - Stream of data received over time
  - May be high frequency or high volume -> cannot be stored
- **Forecasting future behavior**
  - «Is my turbine about to break down?»
  - «what will be the next trending topic on reddit?»
  - «should buy or sell my bitcoin?»
- **May have stringent QoS requirements**
  - Latency of the answers
- **Data is changing over time**
  - Example: 2020 mobility data is completely different from 2019 due to covid lockdowns.

Out-of-core Learning

Even when you have a static dataset you may want to use online algorithms due to **memory/computation requirements**.

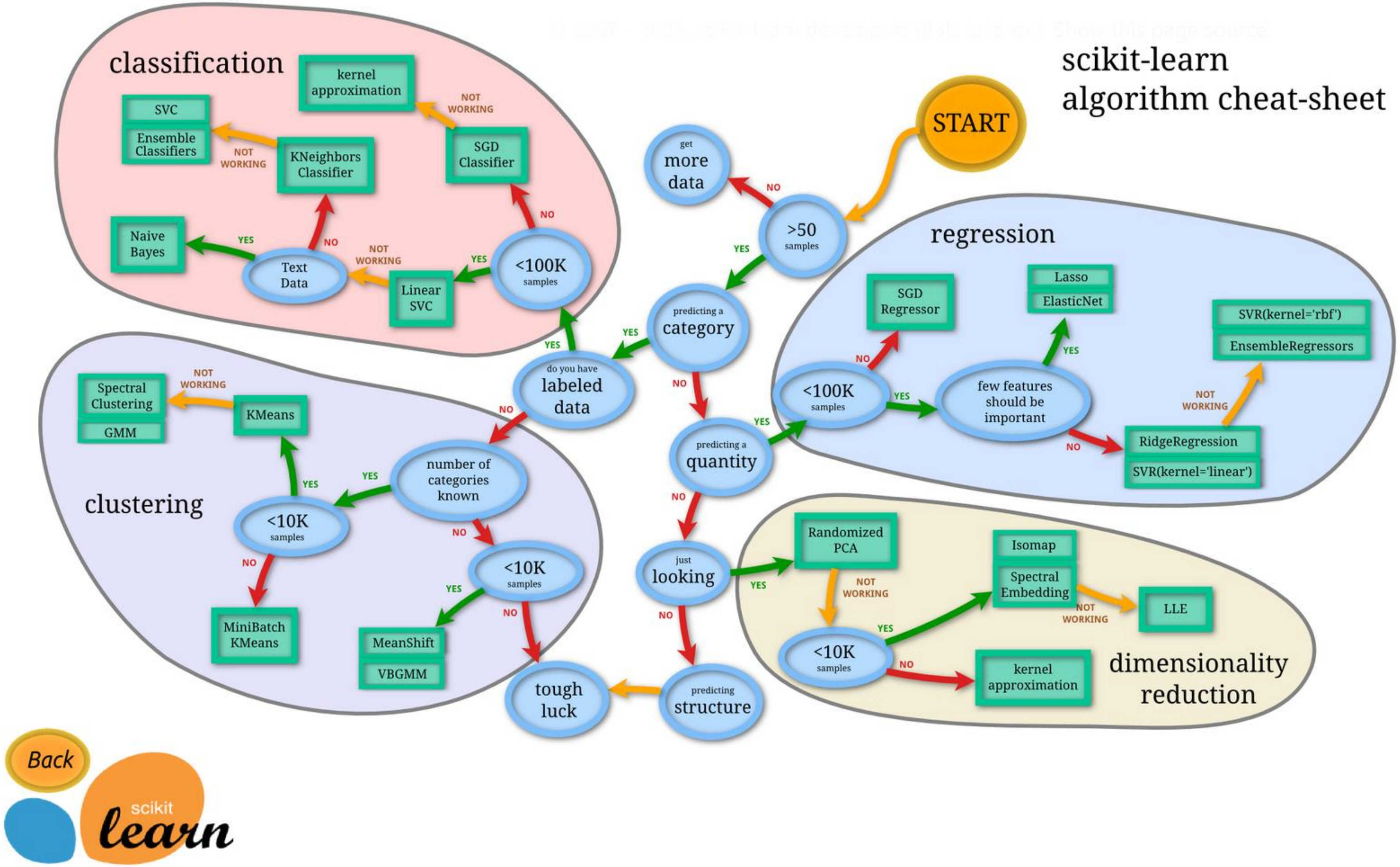
Principal Component Analysis:

- offline: QR decomposition
- Online: Incremental PCA

Linear models:

- offline: Ordinary Least Squares
- Online: Stochastic Gradient Descent

Out-of-Core Learning

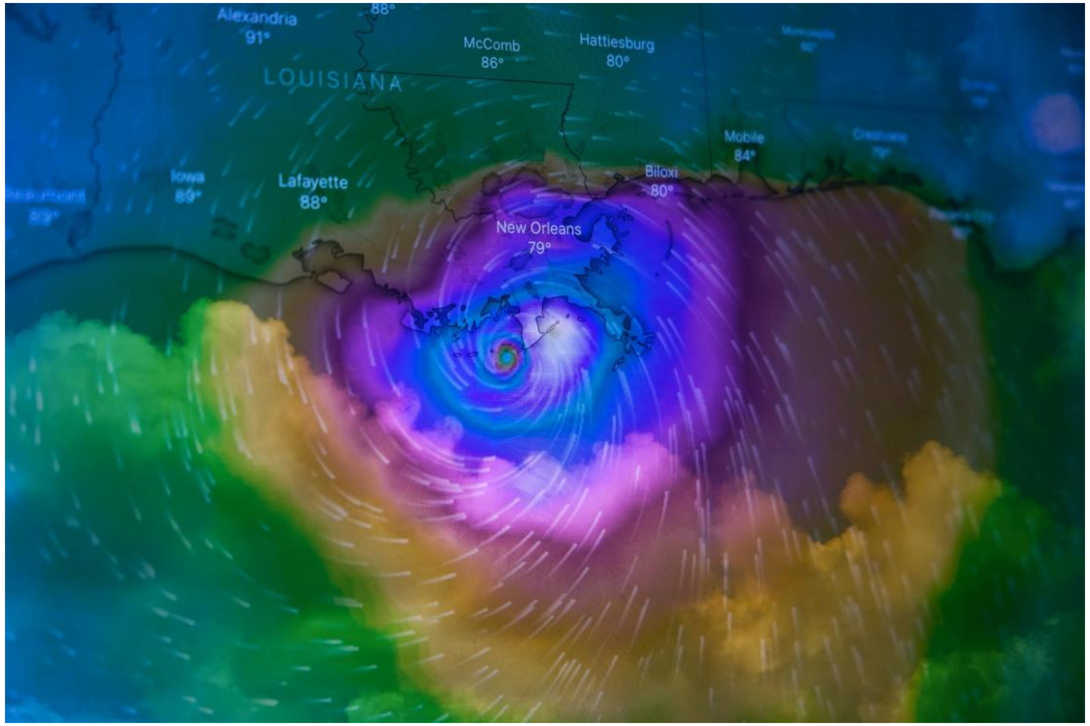


source: scikit-learn docs

Example: Time Series Forecasting

Weather forecasting

- Given current atmospheric pressure/temperature predict next state.
- Chaotic system: predictions far in the future are very difficult.
- If you had infinite precision you would only need present data.



Example: High-frequency Trading

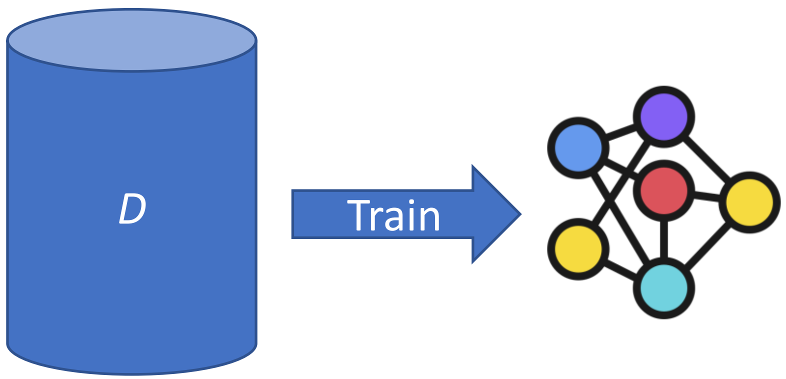
Models need to predict the future price of a financial instrument and decide a proper action (buy/sell)

- Latency requirements: the price is going to change if the prediction is too slow.
- Recent data is much more informative than past data

Batch vs Online Learning

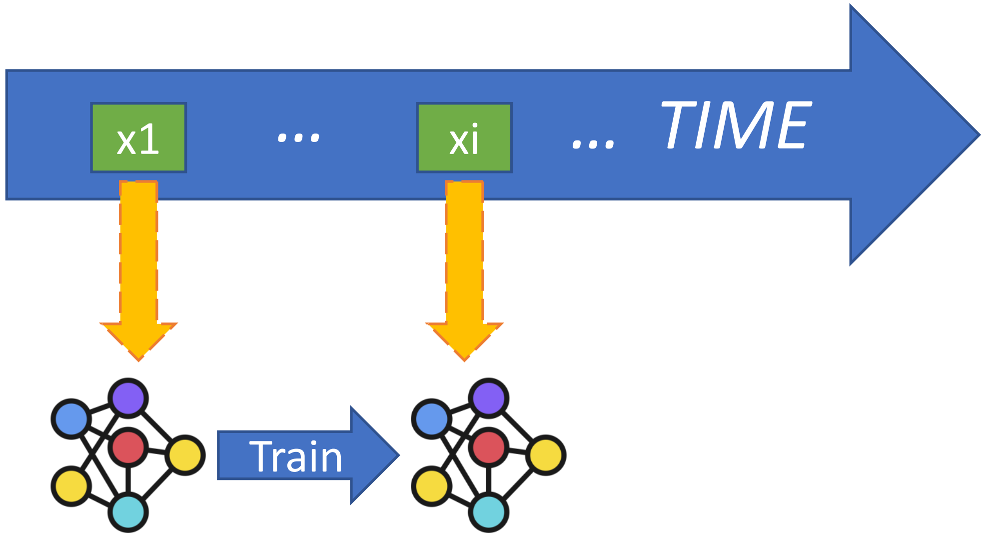
Batch Learning

- i.i.d. sampling / Access to all the data
- No stringent computational constraints/latency of training
- Separate training/eval phase
- Train signature: $\theta = A(D)$

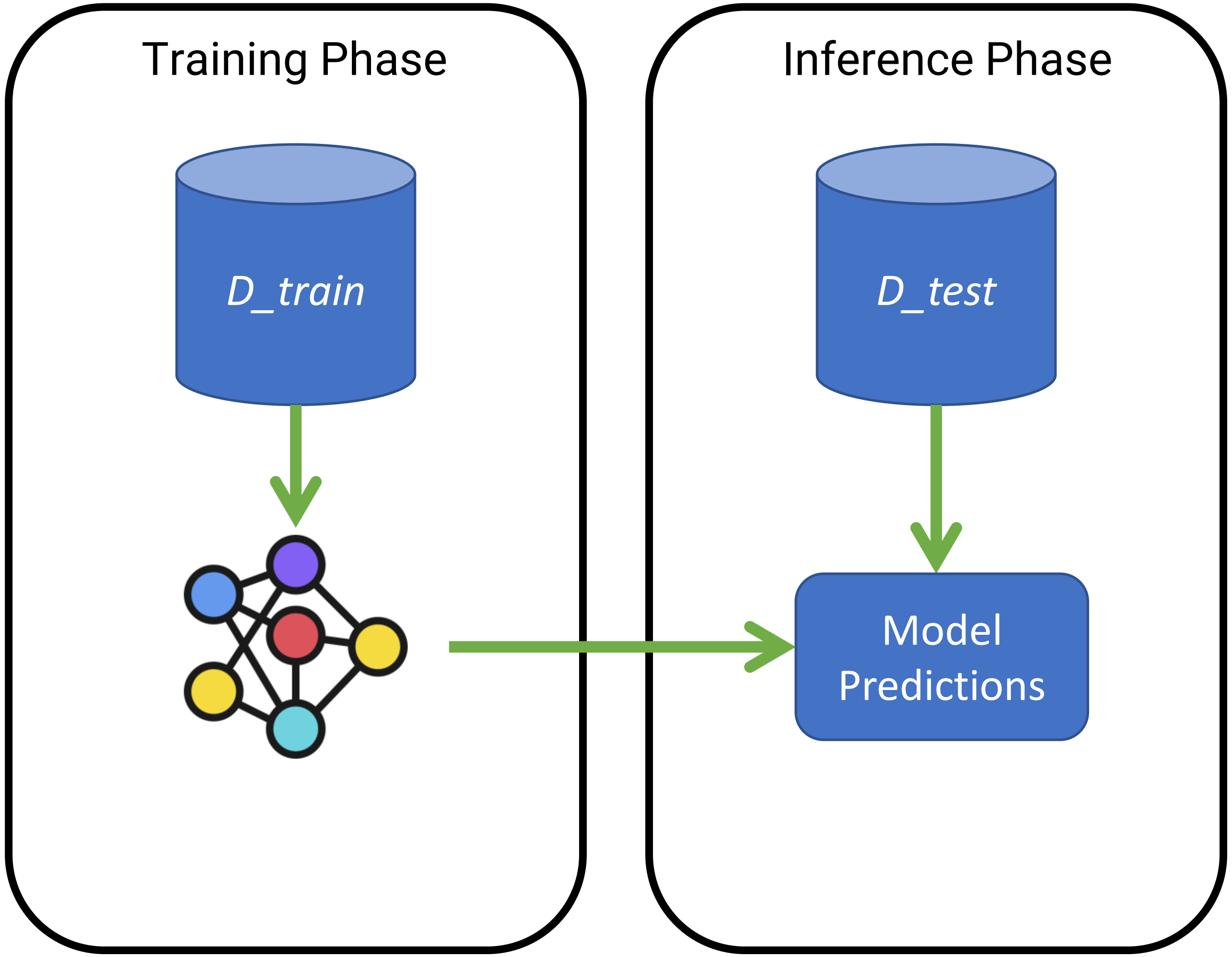


Online Learning

- Sequential access to data
- Computational constraints
- Interleaved training/eval
- Train signature: $\theta_t = A(x_t, \theta_{t-1})$

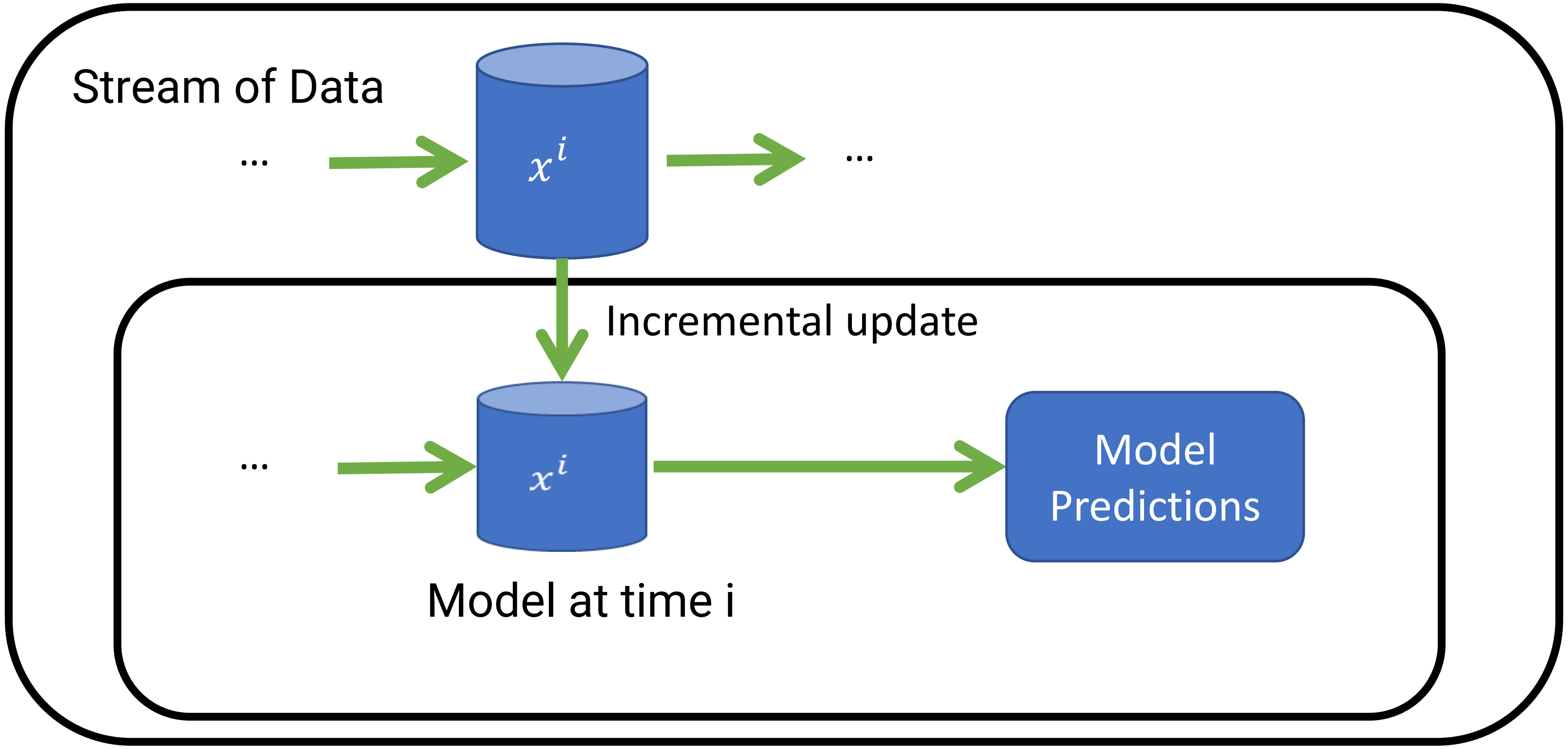


ML Workflow – Train-then-test



OML Workflow – interleaved train-and-test

In OML, we often have only one stream.
How do we do the evaluation?



OML Training with Prequential Evaluation

```python
model = GaussianNB()
PACC = PrequentialAccuracy()

for x, y in stream:
    # PREQUENTIAL EVAL
    # first, we predict on the new sample
    # in a real problem, at this point we don't know the
    # target y yet (e.g. in a forecasting problem)
    y_p = model.predict_one(x)


    # ONLINE LEARNING STEP
    # at some point, y becomes available
    # and we train the model on the new sample
    model.learn_one(x, y)

    # we will also keep track of performance metrics over time
    PACC.update(y, y_p)
```

Holdout vs Prequential Evaluation

- **Holdout**: evaluate current model on a separate test set at regular time intervals.
  - Requires a separate test set
  - In presence of nonstationarity the test set must be updated
  - Expensive: full evaluation at each step
- **Prequential (predictive sequential)**: (interleaved-test-then-train) each sample in the (single) stream is used for testing before training.
  - Does not require a separate test set
  - More efficient than holdout
  - Aggregate prequential loss over time

Gama, J., Sebastião, R. and Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In ACM KDD, 2009.

OML Problem

- At time t
  - The stream has a distribution $p_t(x, y)$, which may change over time
  - We have a single example $< x_t, y_t > \sim p_t(x, y)$
  - We have a model trained on $p_1, \dots, p_{t-1}$, but we predict data from the distribution p_t
- Therefore, there is a difference between the train and test distribution
  - We need to recognize distribution drifts
  - We may need to forget previous data if they are confilicting with the new distribution
  - We also need to update the model quickly when changes are detected

Prequential Evaluation – Forgetting

- **Problems with aggregation of prequential loss over time:**
  - $\mathrm{PA_T} = \frac{1}{\mathrm{T}} \sum_{\mathrm{t}=1}^{\mathrm{T}} 1\{\hat{\mathrm{y}}_\mathrm{t} = \mathrm{y_t}\}$
  - **During the first iterations, the model is underfitted**
    - The model improves over time, but the prequential error «remembers» all the past errors
    - The prequential error overestimates the holdout error of the current model
  - **In the presence of nonstationary distributions**
    - We want to evaluate the model on future data
    - But our loss is computed on old data (and old models)
- **Solution**: controlled forgetting of the past performance
  - **Sliding Window**: compute accuracy only on last k elements
    - $\mathrm{PA_T} = \frac{1}{\mathrm{k}} \sum_{\mathrm{t}=\mathrm{T}-\mathrm{k}+1}^{\mathrm{T}} 1\{\hat{\mathrm{y}}_\mathrm{t} = \mathrm{y_t}\}$
  - **Fading Factor**: running average of the loss
    - $PA_T = \alpha PA_{T-1} + (1-\alpha)\, 1\{\hat{\mathrm{y}}_\mathrm{t} = \mathrm{y_t}\}$

Streaming Cross Validation

How do we split the stream if we want to train (and evaluate) an ensemble? We need to ensure diversity and robust training and evaluation.

K-fold distributed cross-validation:

each sample is used for testing in one classifier selected randomly, and used for training on all the others

- Adaptation of offline cross-validation
- Good use of the data (only one 1/k samples are unused)
- high redundancy

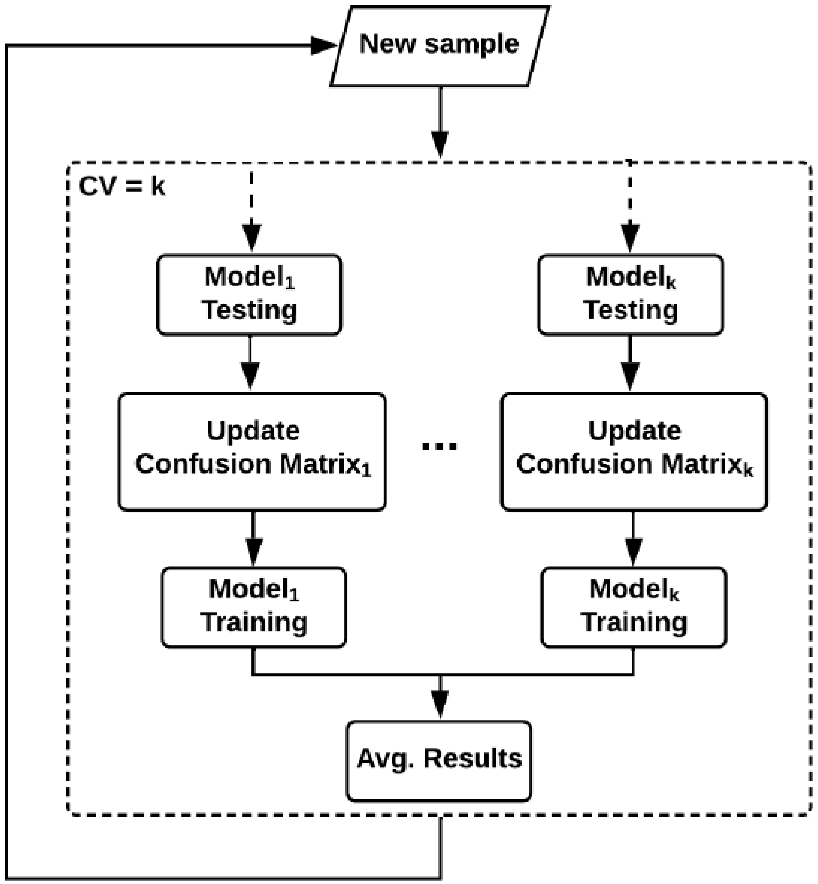
K-fold distributed split-validation:

each sample is used for training in one classifier selected randomly, and for testing in all the other classifiers

- Models trained on disjoint data
- Under utilization of data. Each model uses 1/k data for training

K-fold distributed bootstrap-validation:

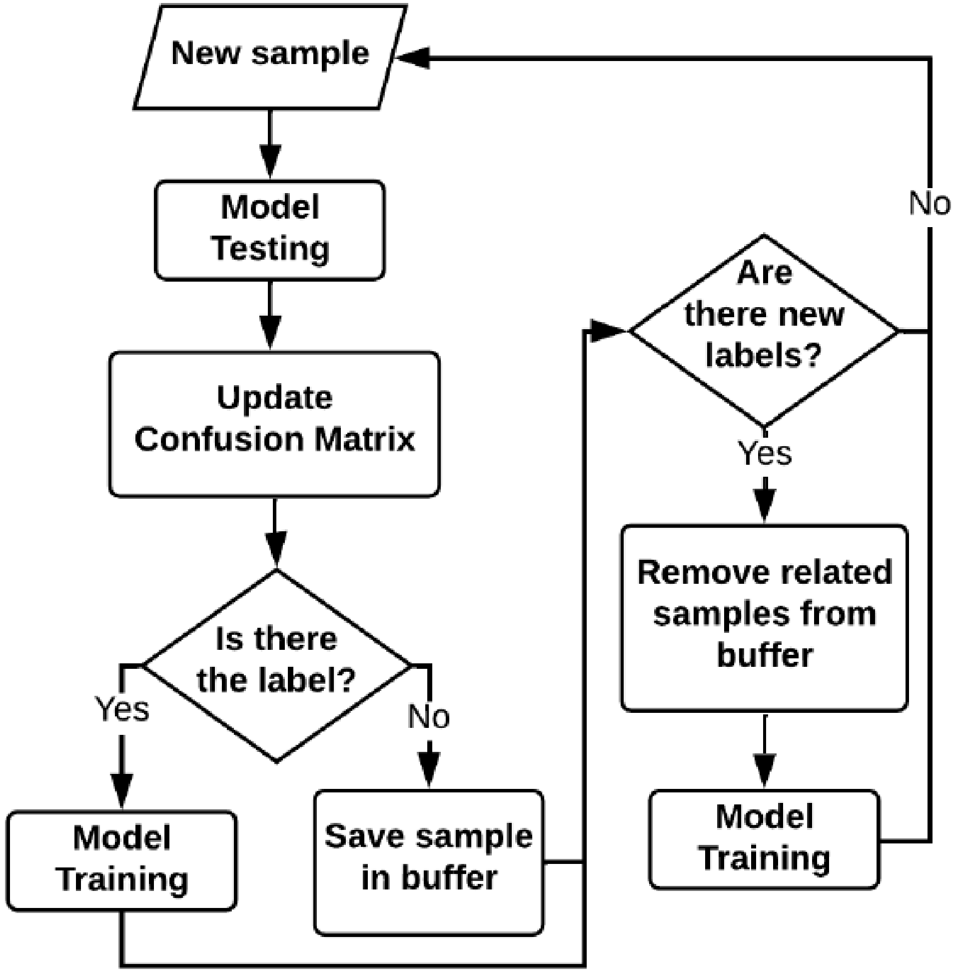
each sample is used for training in approximately 2/3 of the classifiers, with a separate weight in each classifier, and for testing in all the classifiers *(we will see online boostrap in the ensembling lecture)*



Bifet, A., et al: Efficient Online Evaluation of Big Data Stream Classifiers. In ACM SIGKDD, 2015.

Prequential Evaluation – Delayed

- **We don't necessarily have the targets in real-time**
  - In forecasting problems (unsupervised) we always get them in the future
  - In classification (supervised) problems labels may be **delayed** due to offline manual labeling
- **Delayed Evaluation**:
  - Store samples without targets in a buffer
  - Train the model as soon as the targets become available
- Prequential evaluation also works with sparse targets



Gomes, HM., et al: Adaptive random forests for evolving data stream classification. In Machine Learning, 2017.

Evaluation Metric – Kappa statistic

- **In OML, data may be unbalanced**
  - Accuracy is not a good measure
  - Trivial baselines (most common class) get high accuracy
- **Kappa Statistic** compares the accuracy p of a models against the random baseline accuracy p_{rand}
  - **Random baseline** chooses a random class with the same proportion of classes predicted by the model under test
  - Relative improvement w.r.t. the baseline accuracy
  - K=1 perfect classifier
  - K=0 random classifier
  - Very fast to compute online compared to other measures used in imbalanced scenarios such as the AUROC

$$k = \frac{p - p_{rand}}{1 - p_{rand}}$$

Bifet, A., et al: Efficient Online Evaluation of Big Data Stream Classifiers. In ACM SIGKDD, 2015.

Evaluation metric – Kappa-Temporal statistic

- If the proportion of classes predicted by the model is different from that of the stream, k is not a good estimate
  - The classifier may be underfitted
  - There has been a change in the <input,output> distribution
- The **persistent classifier** is a better baseline: predicts the next label is the same as the last seen label.
  - It captures simple correlations in the stream

$$k = \frac{p - p_{per}}{1 - p_{per}}$$

I. Žliobaitè et al. Evaluation methods and decision theory for classification of streaming data with temporal dependence. In Machine Learning, 2015.

OML – Data Ordering

- In general, the data ordering will affect the model's performance. Few methods are order-independent
- Easy ordering with strong correlations:
  - Example: 0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2
- Adversarial ordering:
  - Adversarial environments can result in data designed to «break» the model
  - Example: in finance, other actors may exploit a previously profitable strategy
  - Some online methods are designed for optimal worst-case performance (we won't study the theory).

OML – Benefits and Challenges

Benefits

- Efficient algorithms
- Model update
- Forget the past when it's not relevant

Challenges

- Concept Drift (non-stationarity)
- Imbalance
- Hyperparameter Tuning

Take Home Messages

- **Online ML requires a change of paradigm** compared to offline
  - Efficiency as a key focus
  - iid assumption is broken
  - Prequential evaluation

Next Lecture

- **Concept Drift**
  - Definition
  - Estimation
  - Detection

Notebook – Example of OML with River

- Notebook on moodle OML1-intro.ipynb
  - Dependencies: river, seaborn, scikit-learn