Data popularity for the Cache Eviction Algorithms using Random Forests

Olga Chuchuk[1,2], Markus Schulz[1]

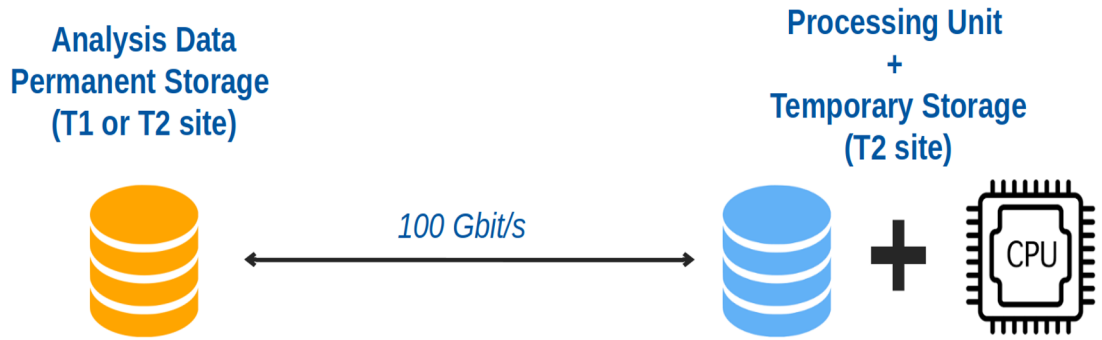
(1) CERN, IT-GOV Group

(2) University of Côte d'Azur

Motivation for research:
Improving Cache Performance for Remote Physics Analysis

Ways to perform analysis remotely in the WLCG:

- The file can be **read** (therefore, moved or replicated) before the computation starts
- The file can be **streamed** while the computations runs (but removed from the processing node right after)
- The file can be **cached** (also streamed during the computation and saved in the cache).



$$Bytes\ Hit\ Ratio\ (BHR) = \frac{amount\ of\ bytes\ retrieved\ from\ the\ cache}{amount\ of\ requested\ bytes}$$

$$Bytes\ Miss\ Ratio\ (BMR) = 1 - BHR$$

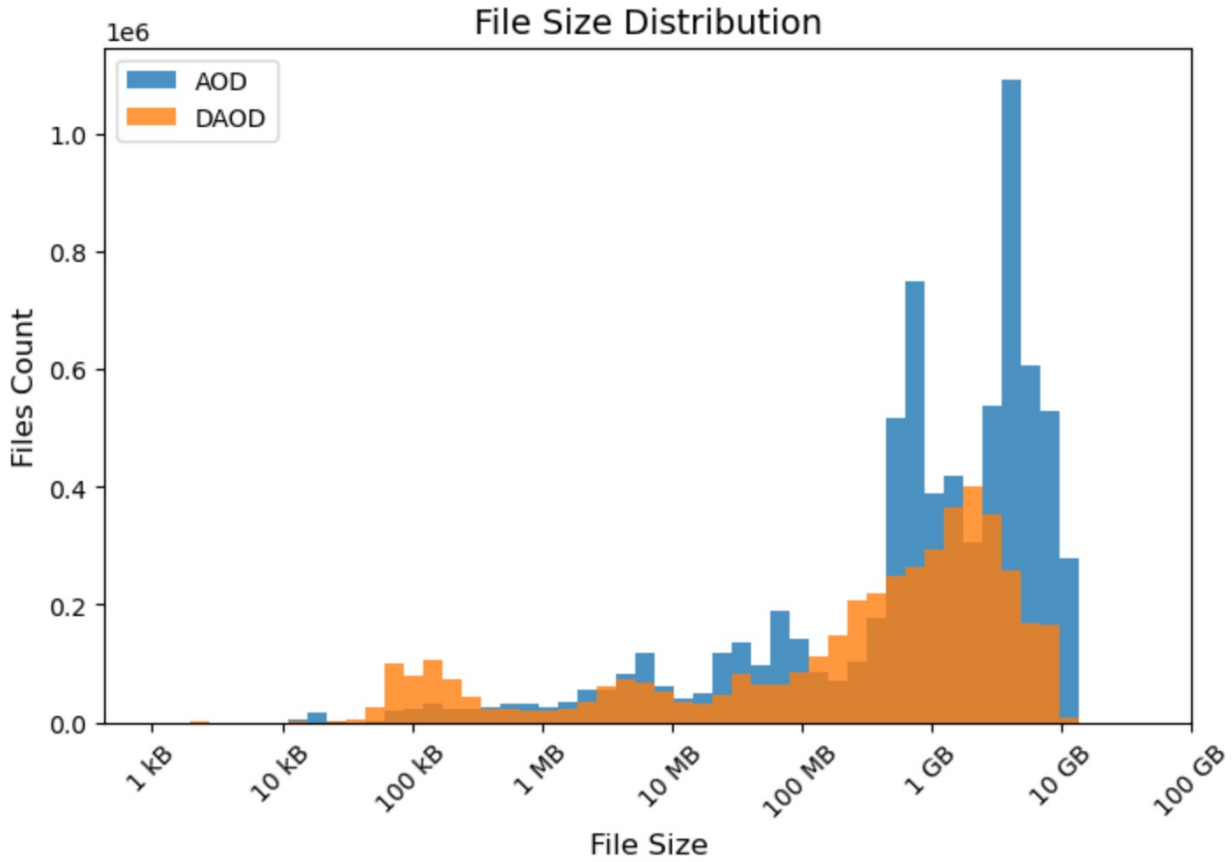
Data access trace

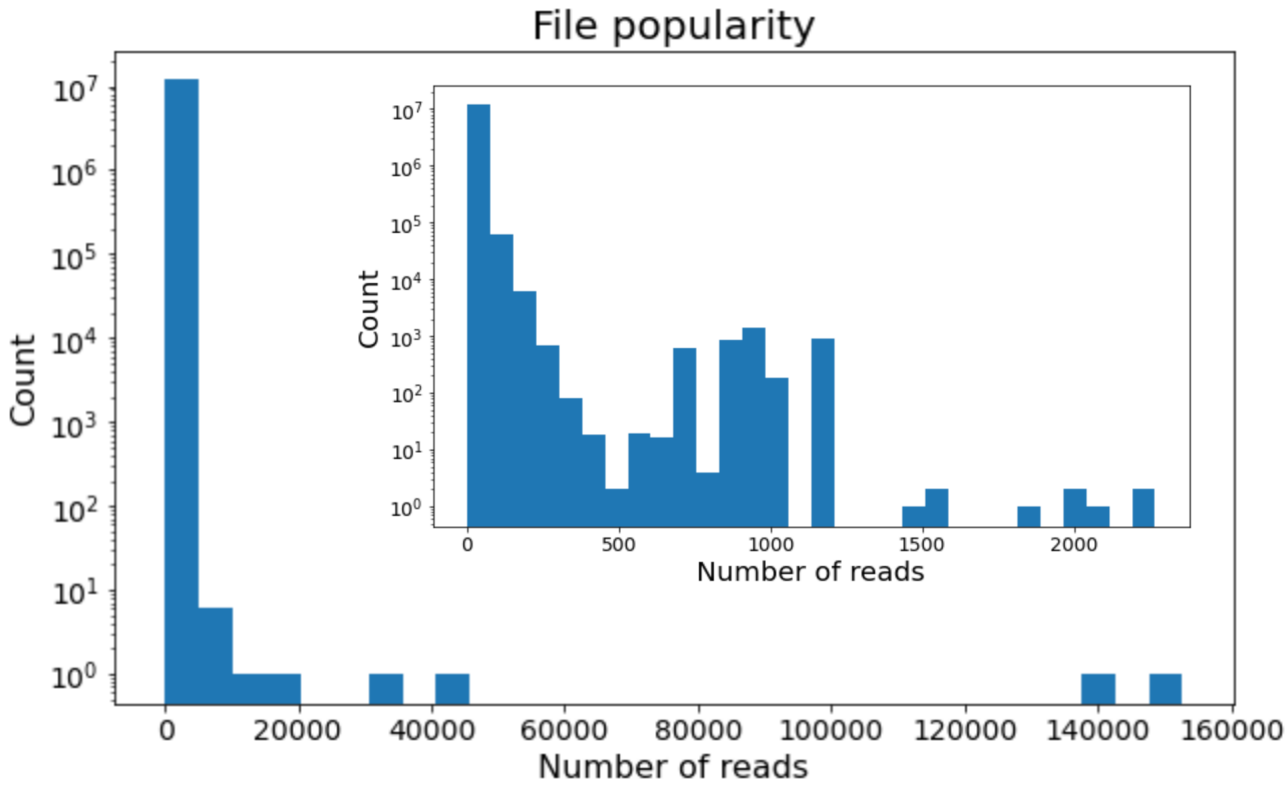
Log data sources:
1. EOS Report Logs (CERN)
2. Rucio Logs

Total number of reads
N = 5.6e8
Total number of files
M = 1.2e7
Total number of datasets
D = 2.9e5

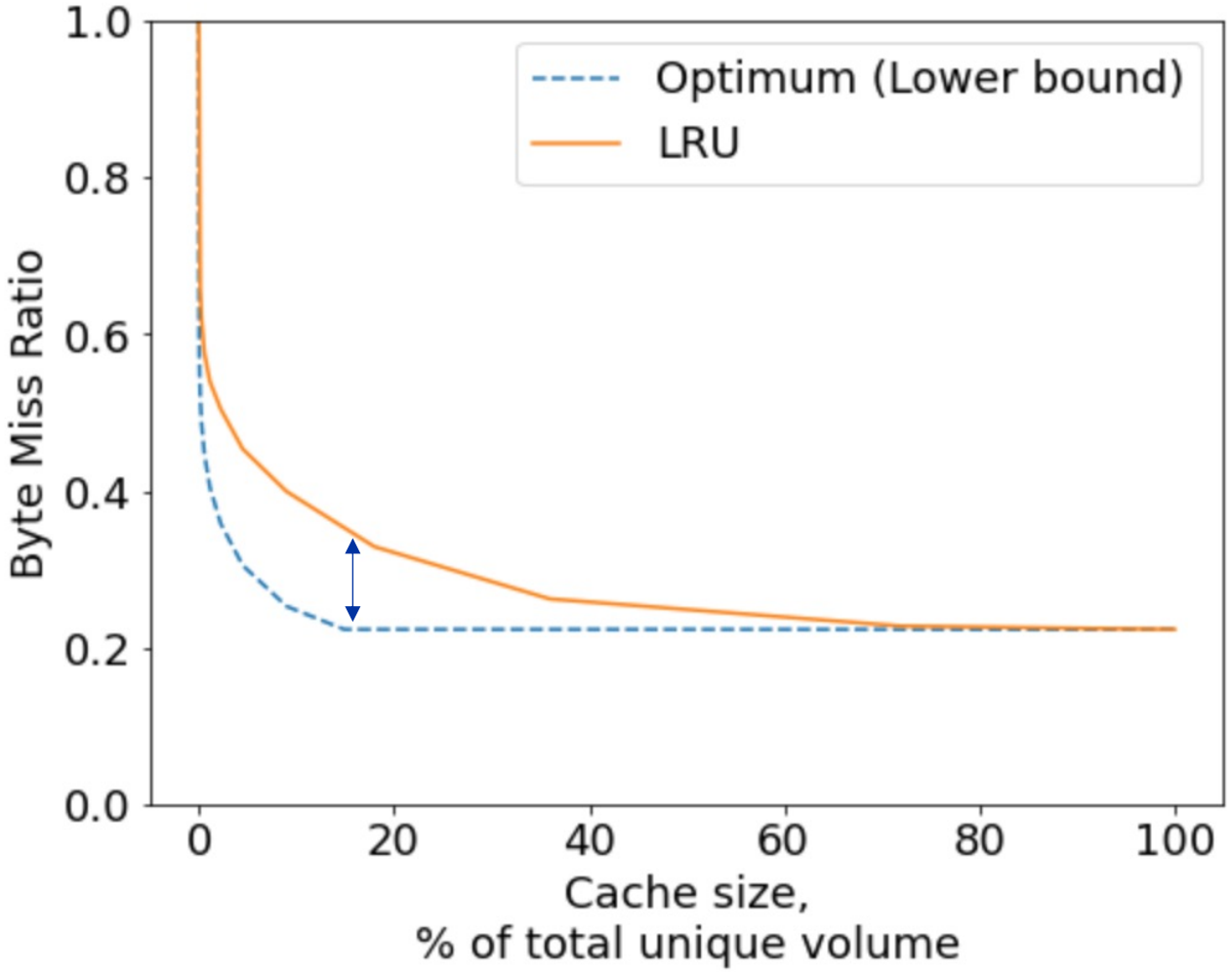
Average File Size
2.13GB

Max File Size
51.30GB





Comparing caching algorithms



Cache performance plots:

- Decays from 1 to a particular value as the cache size increases
- Cold misses determine the min value
- Allow to compare the performance of different cache eviction policies

Least Recently Used (LRU)

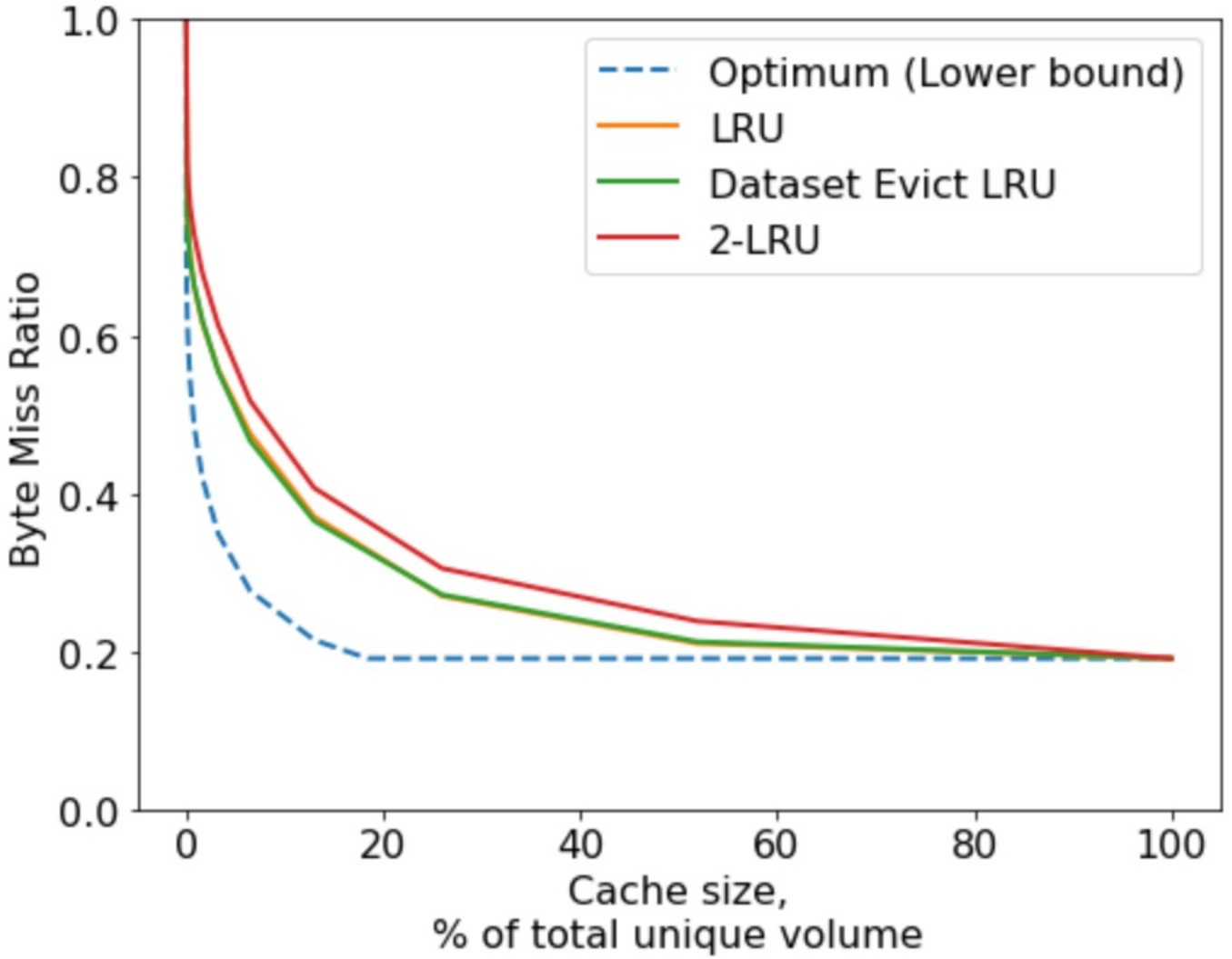
- Easy to implement and maintain
- Performs well on this trace

Lower Bound of the Optimal $\mathrm{Policy}^{(*)}$

- Optimum is NP-hard to construct
- Only theoretical as requires the knowledge of the future reads

(∗) Chuchuk, O., Neglia, G., Schulz, M., & Duellmann, D. (2022, March). Caching for dataset-based workloads with heterogeneous file sizes. In ISGC 2022-International Symposium on Grids & Clouds 2022.

Previous attempts to improve BMR



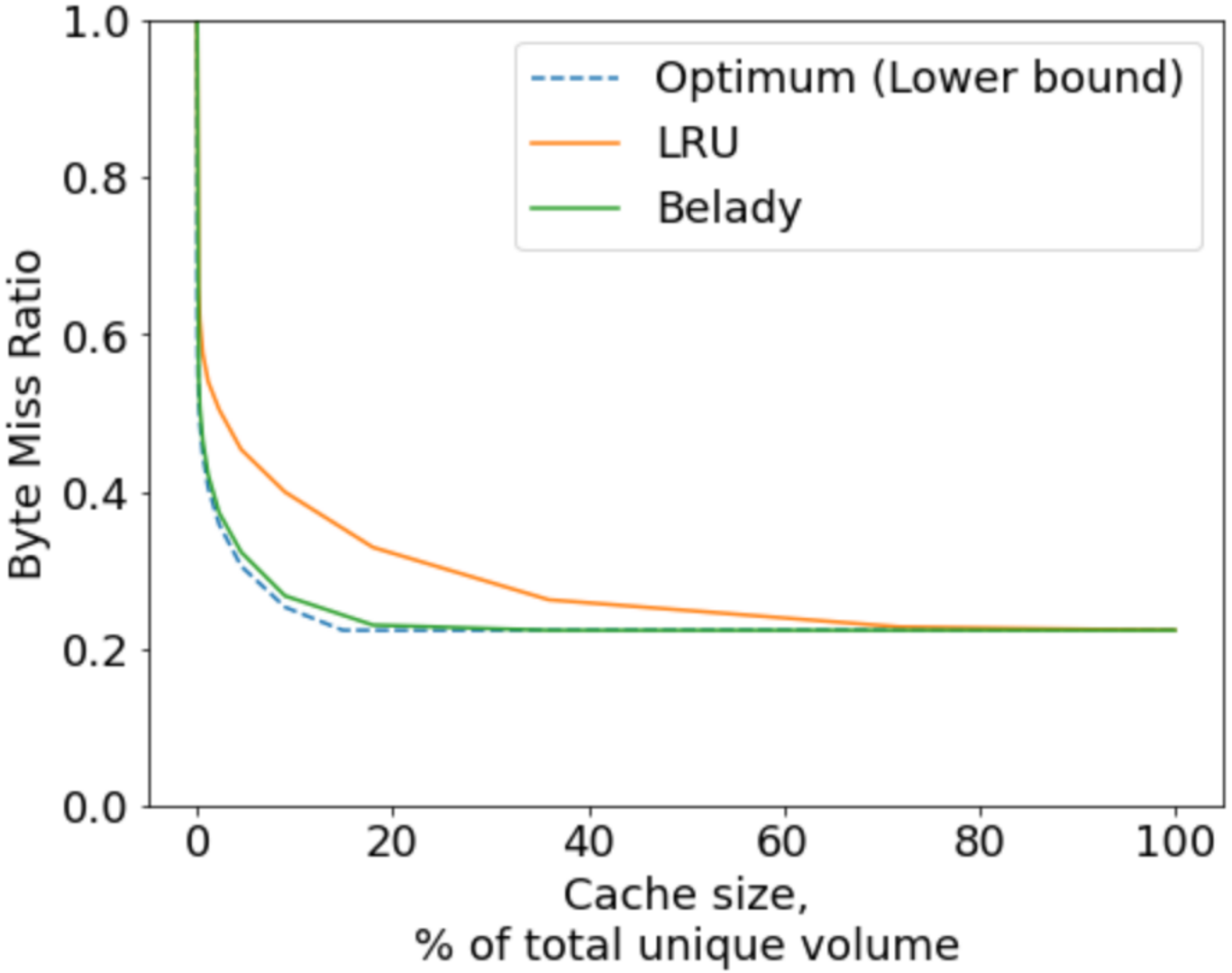
Why is our problem different?

- Optimization of the BMR
- Different from CDNs and Web
  - File size
  - Number of users

Directions that we explored:

- Existing cache eviction policies (like 2-LRU and 2-staged LRU)
- Dataset-specific policies (having in mind that files within one dataset tend to be read together)

Performance of the clairvoyant algorithm



The clairvoyant algorithm (or Belady):

- Removes the files that will be used the furthest into the future.
- Only theoretical as requires the knowledge of the future.
- Performs optimally when the file sizes are equal.

When applied to our trace:

- Approaches optimum
- Significantly outperforms LRU

Architecture of the ML-based approach

Features of existing files

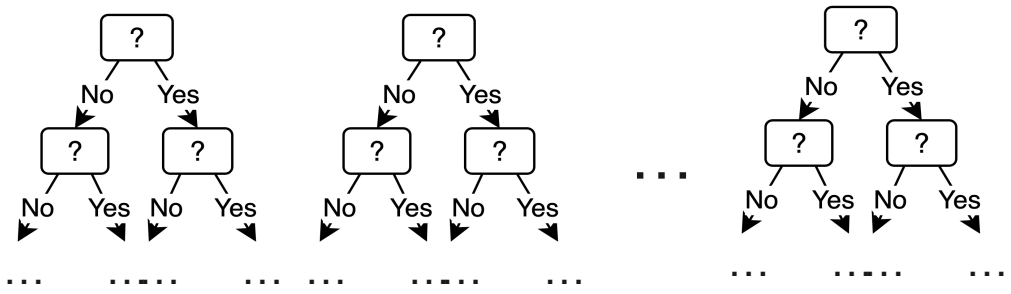
Predicting future file reads

Predicted time (or probability) of file reuse

Integration into the caching model

Cache eviction decisions

(**Random Forest** model):



Random Forest advantages:

- Classification and regression
- No need for feature normalization
- Parallelizable
- Interpretable

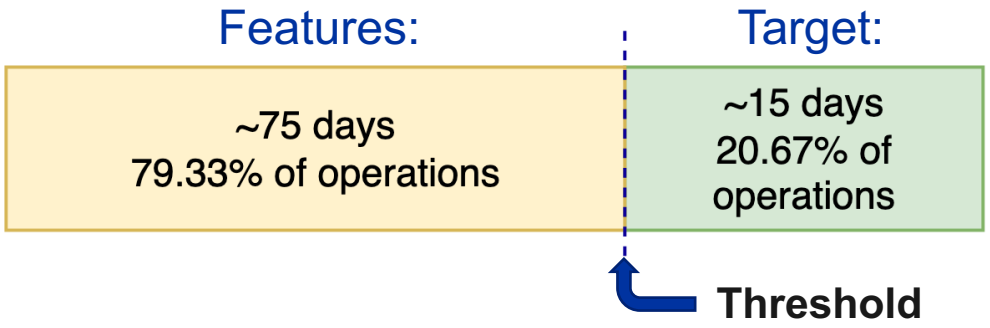
Prediction of the file reads. Regression

Predicting the logarithm of the reuse distance

List of features:

- File size
- Dataset size (volume)
- Frequency of the reads
- Recency of the reads
- Duration of the reads
- Dataset size (number of files)
- …

(18 features in total)



Data volume:

Filtering stage	Number of files	Percentage of files
Before the threshold	10,556,833	88.98%
With 2+ accesses	4,719,609	39.78%
With creation time	2,378,787	20.05%
With target access	342,878	2.89%

Results:

0.29 – RMSE* on the training data (70%)

0.34 – RMSE on the test data (30%)

*RMSE = Root Mean Squared Error

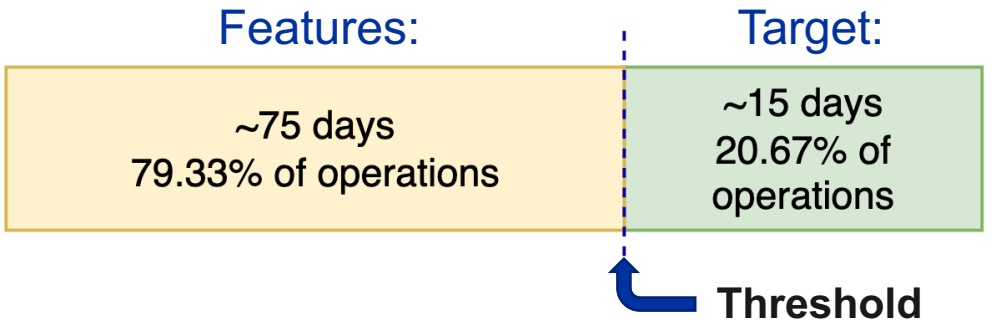
Prediction of the file reads. Classification

Predicting if reread in the next 15 days

List of features:

- File size
- Dataset size (volume)
- Frequency of the reads
- Recency of the reads
- Duration of the reads
- Dataset size (number of files)
- …

(18 features in total)



Data volume:

Filtering stage	Number of files	Percentage of files
Before the threshold	10,556,833	88.98%
With 2+ accesses	4,719,609	39.78%
With creation time	2,378,787	20.05%
After target balancing	686,942	5.79%

Results:

0.11 – RMSE on the training data (70%)

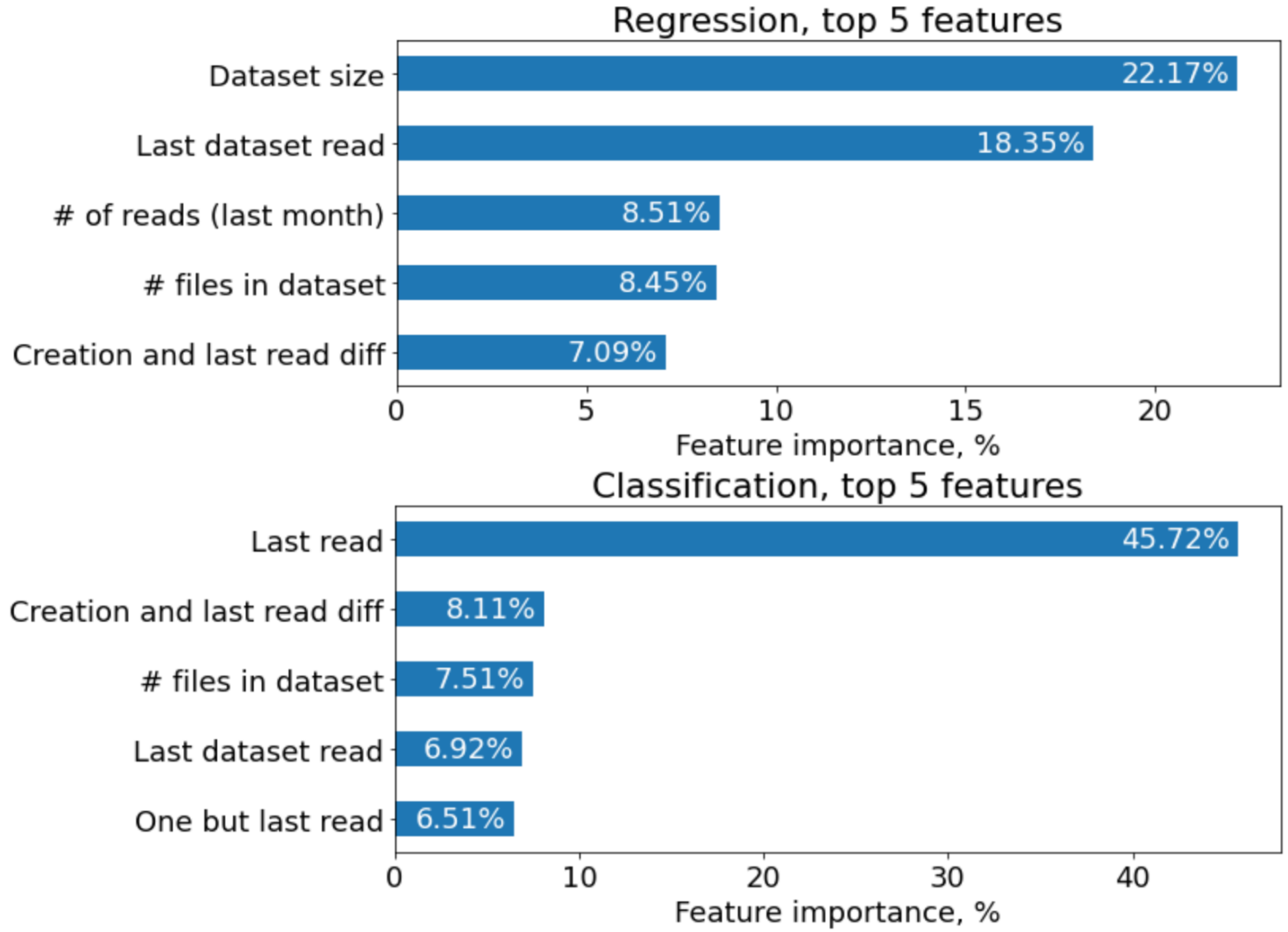
0.12 – RMSE on the test data (30%)

0.99 – AreadUnderROC* on the training data

0.99 – AreadUnderROC on the test data

*ROC = (Receiver Operating Characteristic). Plots the true positive rate (TPR) against the false positive rate (FPR)

Feature importance



- **Feature importance** is calculated based on how frequently the feature is chosen for a split, and the subsequent decrease in the impurity of the decision tree nodes.
- The higher the feature importance score, the more important the feature is for the model's predictions.
- Dataset-related features tend to contribute greatly to the models, as well as the recency (last read, last dataset read).

ML model with the features existing in the trace

List of features
(6 in total):

- File size
- Time of last read
- Duration of last read
- Dataset size (volume)
- Dataset size (number of files)
- Time of last dataset read

Regression:

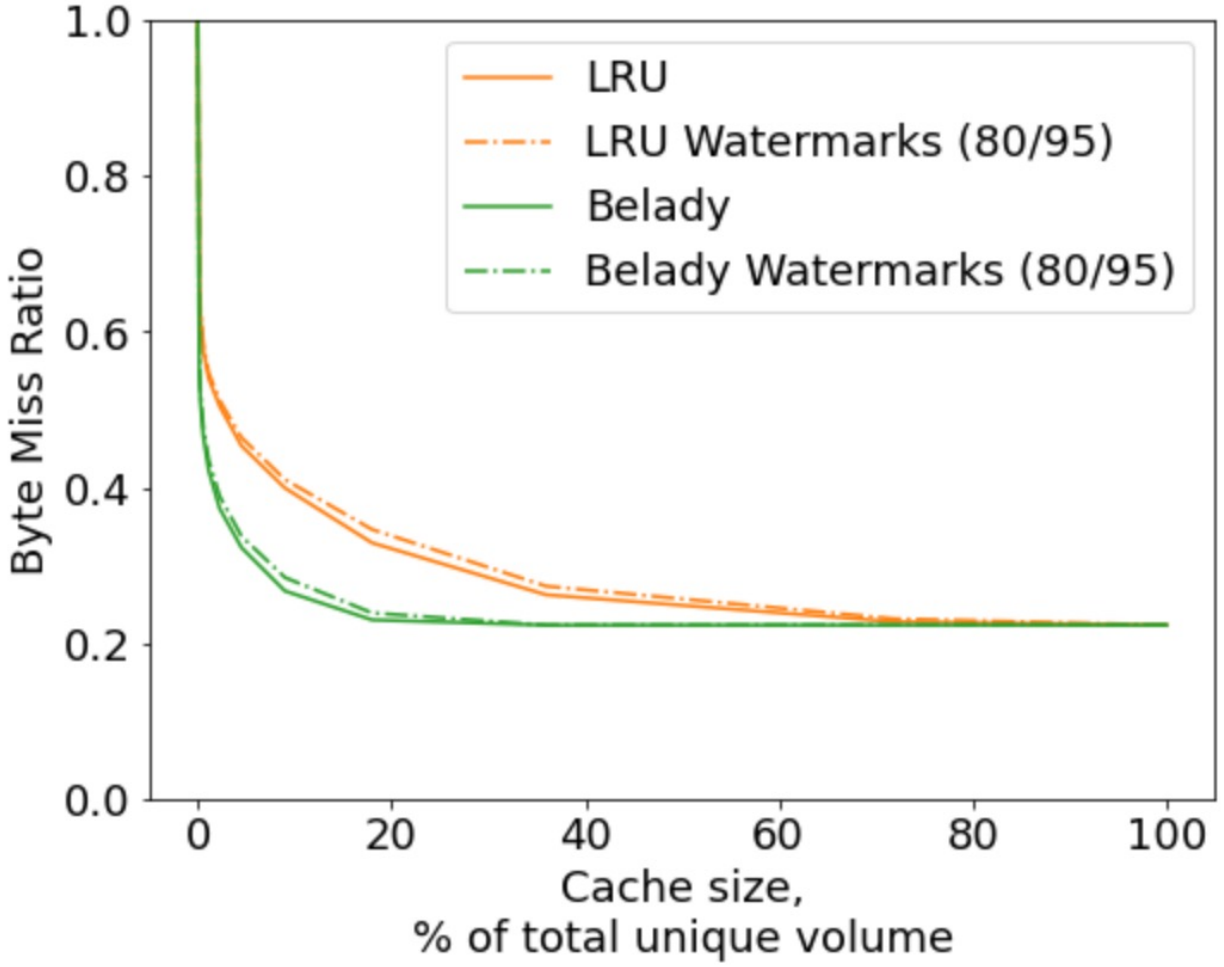
0.43 *(0.29)* – RMSE on the train data (70%)
0.46 *(0.34)* – RMSE on the test data (30%)

Classification:

0.19 *(0.11)* – RMSE on the train data (70%)
0.19 *(0.12)* – RMSE on the test data (30%)

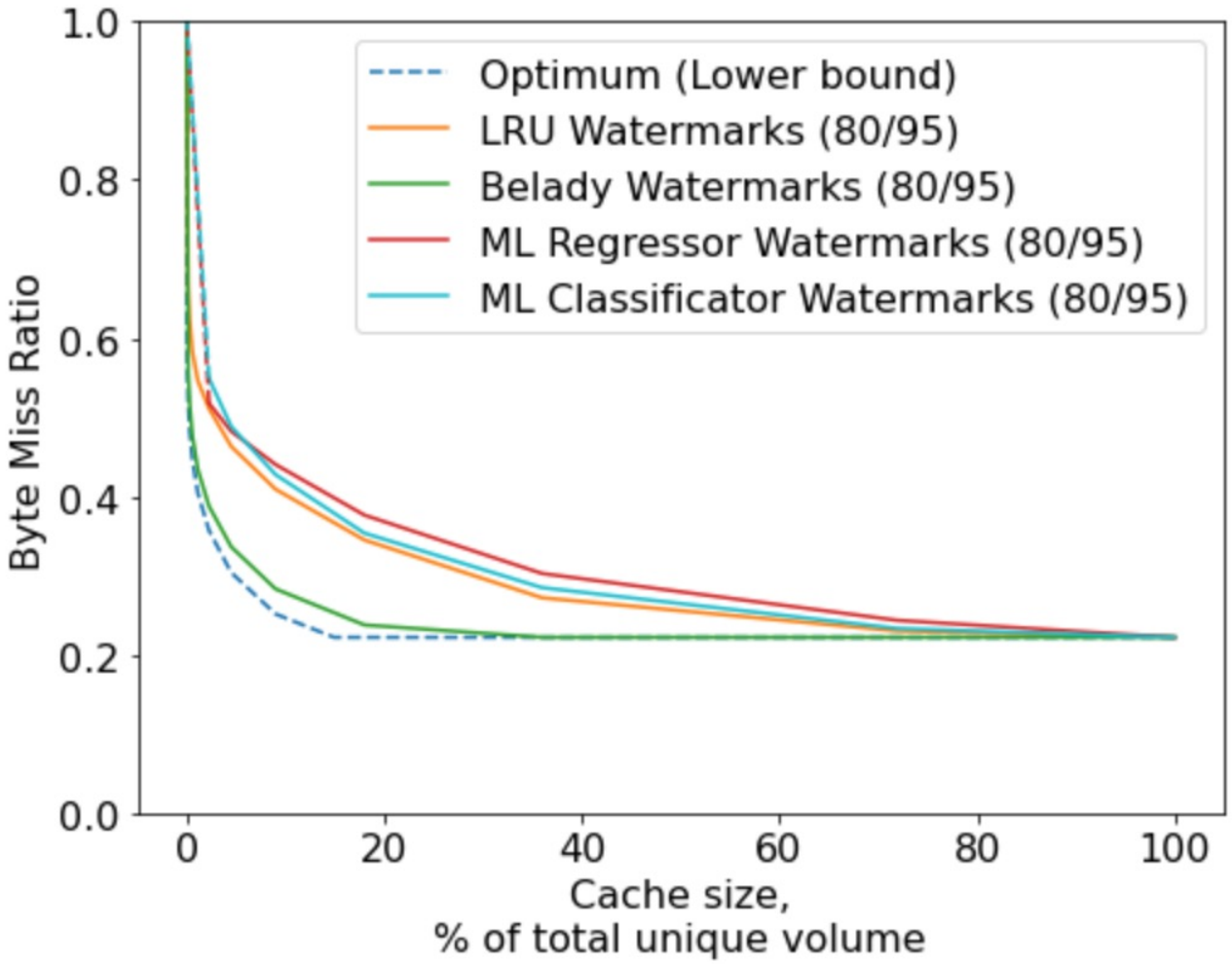
0.98 *(0.99)* – AreadUnderROC on the train data
0.98 *(0.99)* – AreadUnderROC on the test data

Realistic cache implementation
High and Low Watermarks



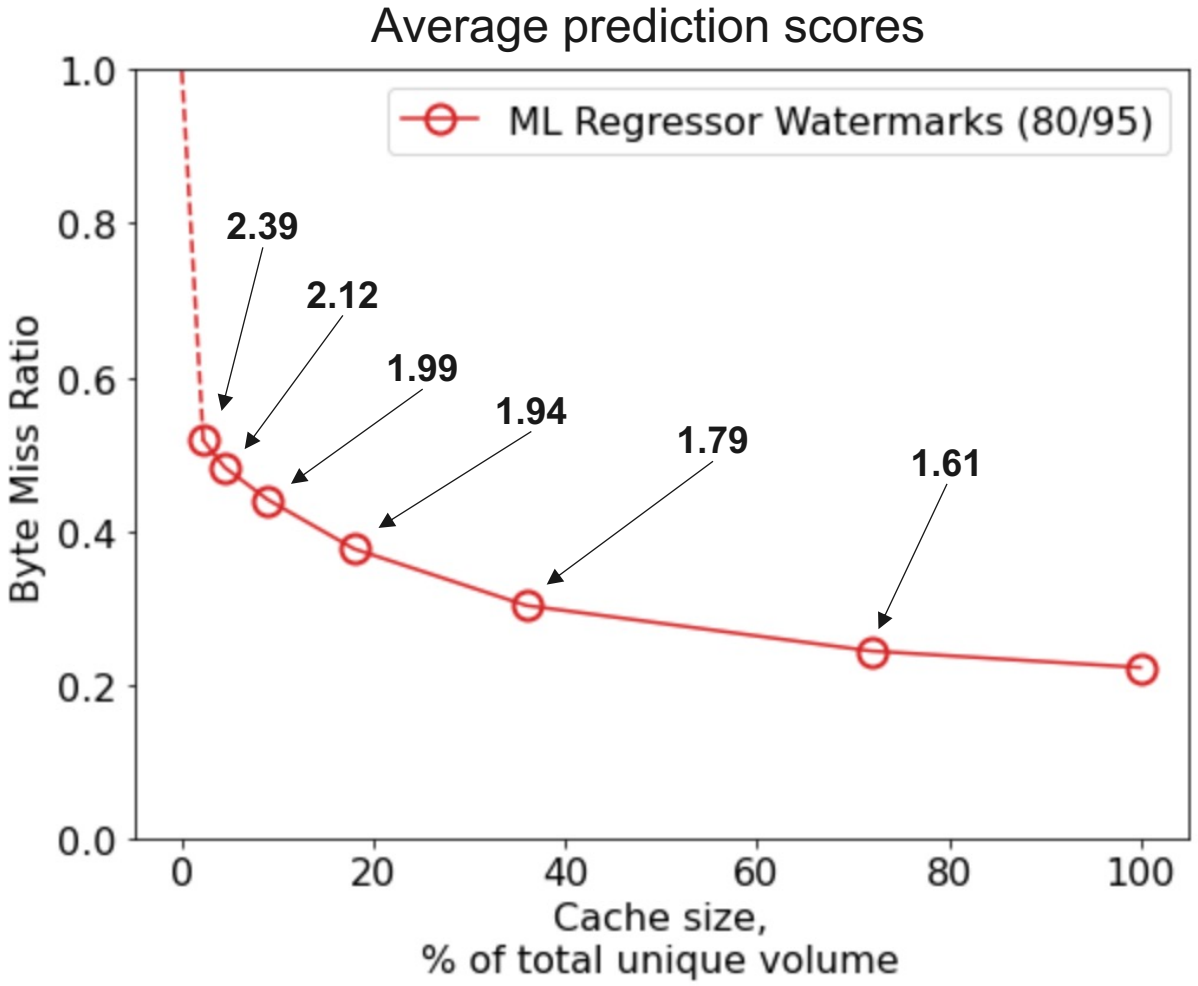
- **High and Low Watermarks** indicate the maximum and minimum amount of data that should be stored in the cache.
- When the **high watermark** is hit, the cache cleaning processed is triggered: based on the implemented cache eviction policy, the files are evicted until the **low watermark** is reached.
- As expected, the performance is slightly worse than that of the original algorithm, but not significantly.
- **Advantage:** the cache cleanup is only run once in a while. Less load on CPU.

Comparing ML-based models with LRU



- When the high watermark is hit, prediction is run on all the files present in the cache. We then sort them based on the predicted reuse distance (or probability to be reused) in order to be evicted from the cache.
- Model with classification permofrms better, but still not able to beat LRU.
- When the cache size is 2-5 %, the difference between 1-2 days of reuse and 1-2 weeks becomes more crucial.

Evaluating ML model real performance



RMSE on the test data: **0.46**

Why the models perform poorly on the actual trace:

- ~60% of the files were used only once.
- The prediction was limited to a 15-day period.

Cache size	# of cache cleanups
100%	0
72%	4
36%	18
18%	53
9%	136
4%	319
2%	724

Conclusions

1. **The LRU replacement policy exhibits satisfactory performance on our trace.**
2. **There is scope for enhancements to the existing approach, at least theoretically.**
3. **Our research has demonstrated that surpassing LRU is challenging (supported by the feature importance distributions obtained from the ML models).**
4. **The challenge is not solely to predict future reads, but also to select the most appropriate way to integrate the predictive model into caching.** Possible alternative ways to do so:
    1. Expanding the training dataset.
    2. Different combinations of ML models can be explored.
    3. Optimizing the hyperparameters remains an area of opportunity.

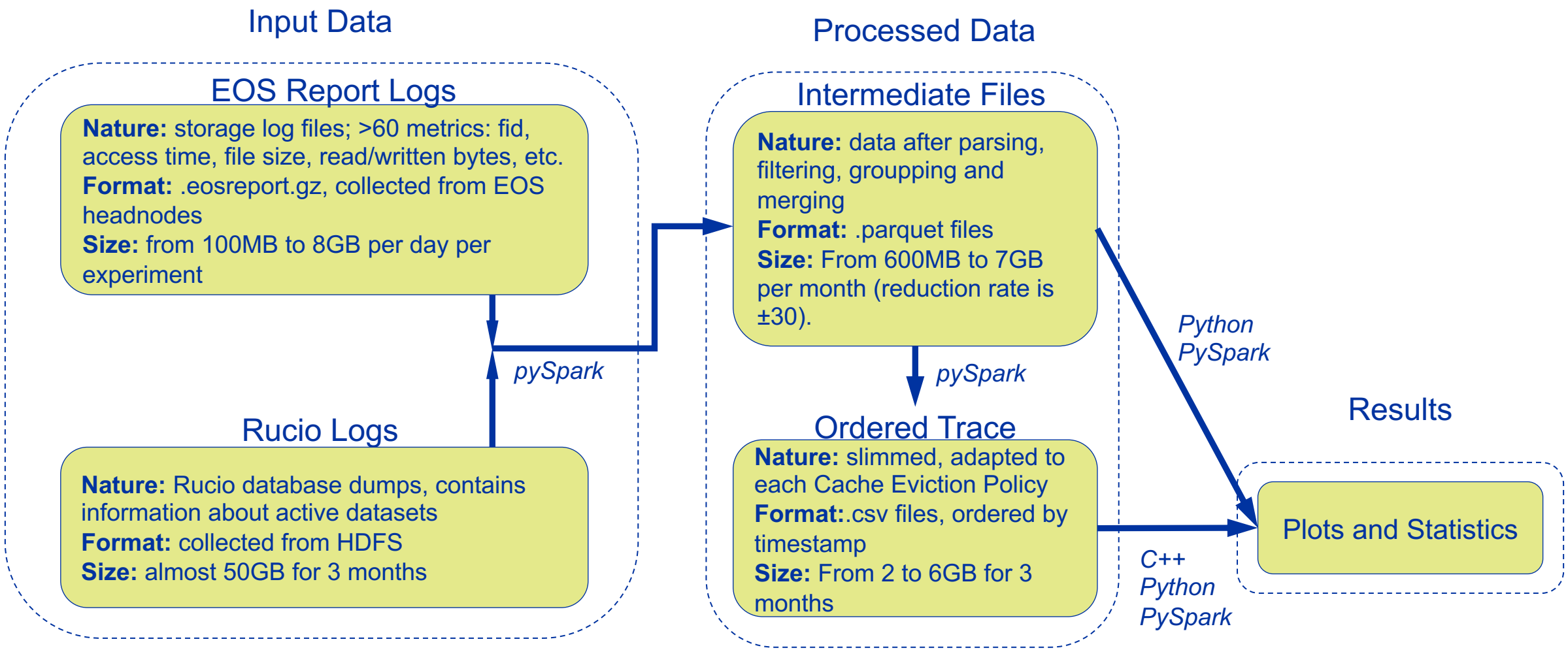
Thank you for your attention!

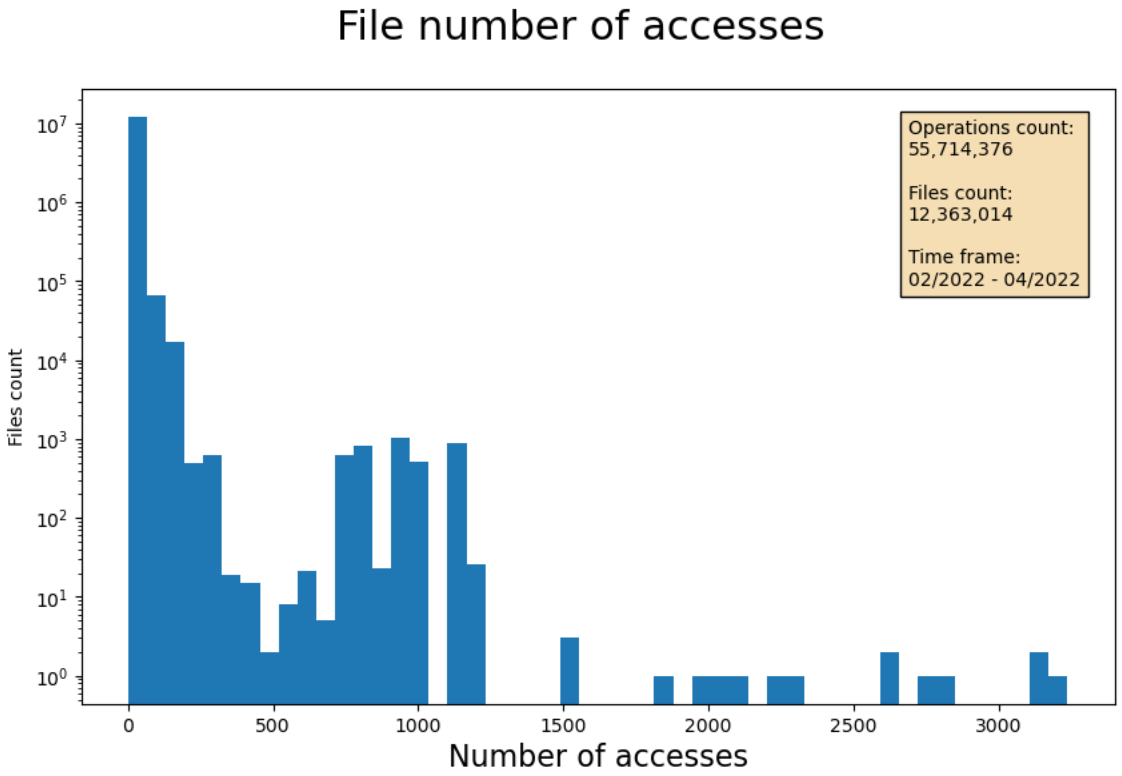
Any questions?

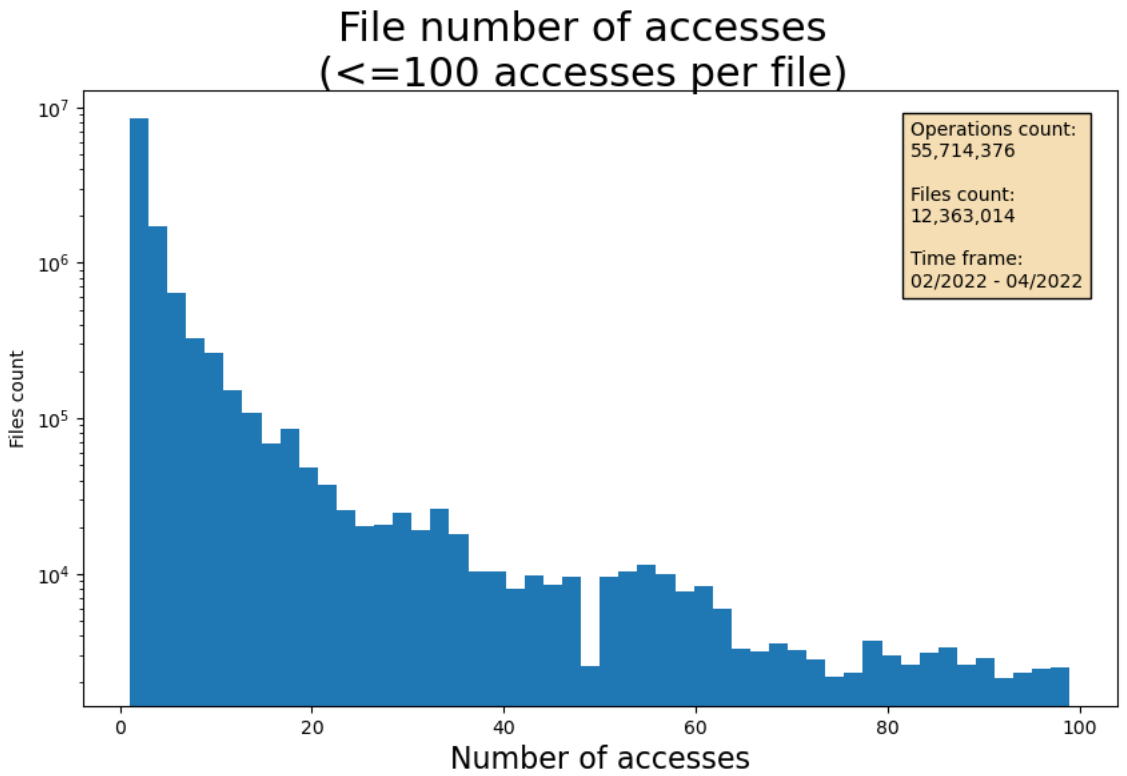
Olga Chuchuk
CERN, IT-GOV group
olga.chuchuk@cern.ch

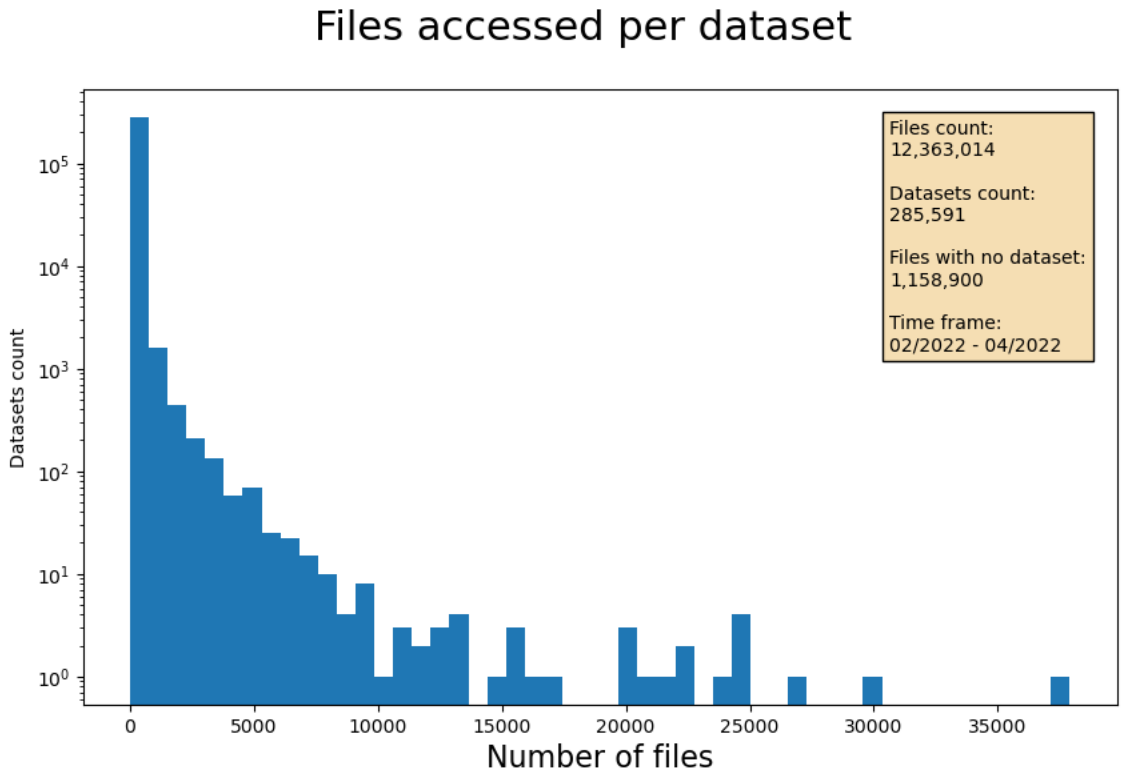
Backup Slides

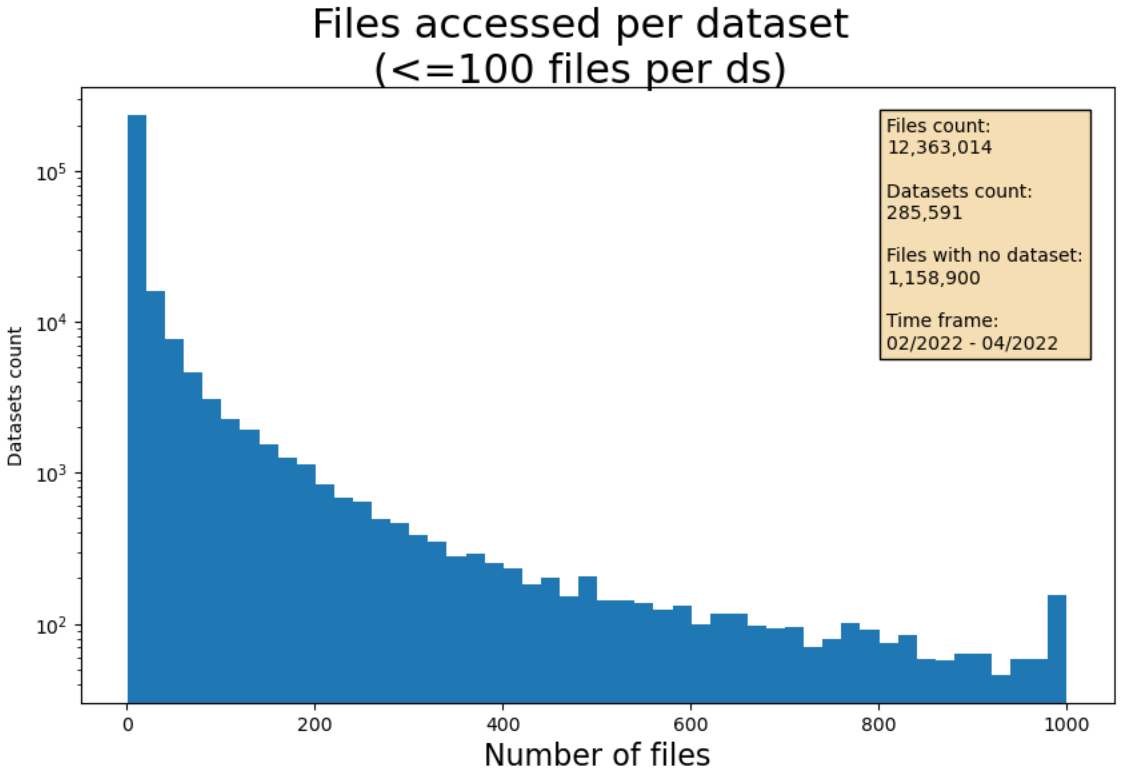
My data processing pipeline

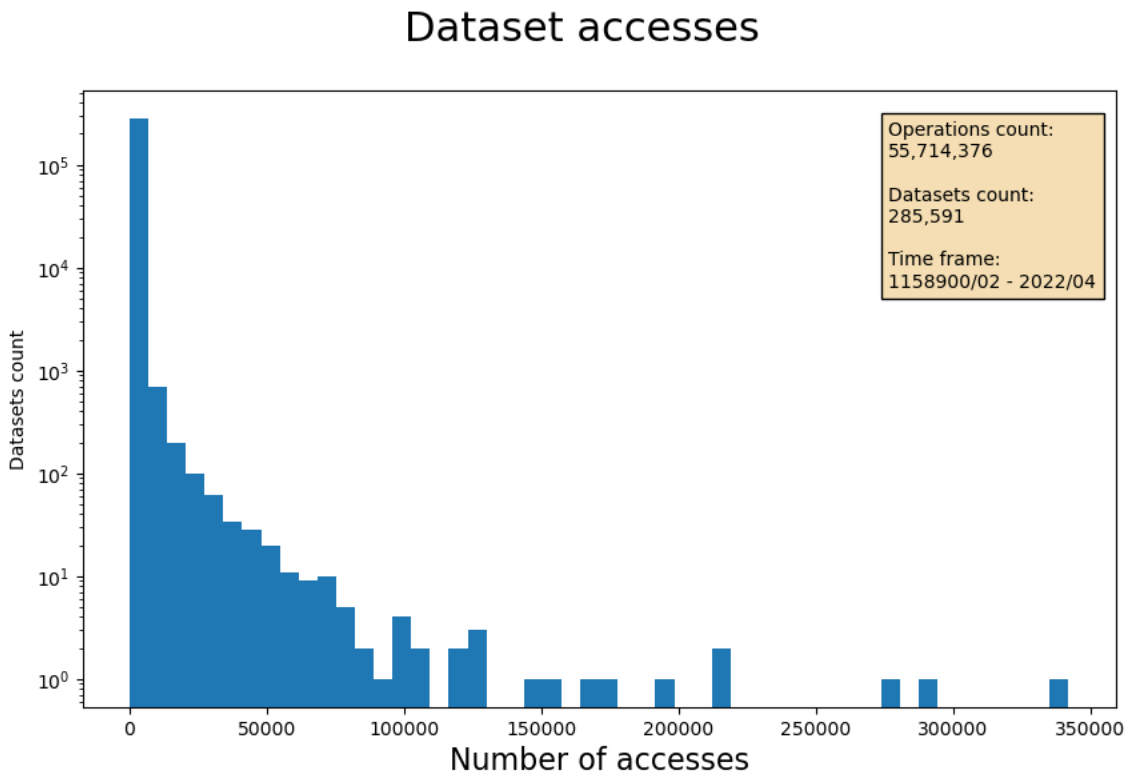


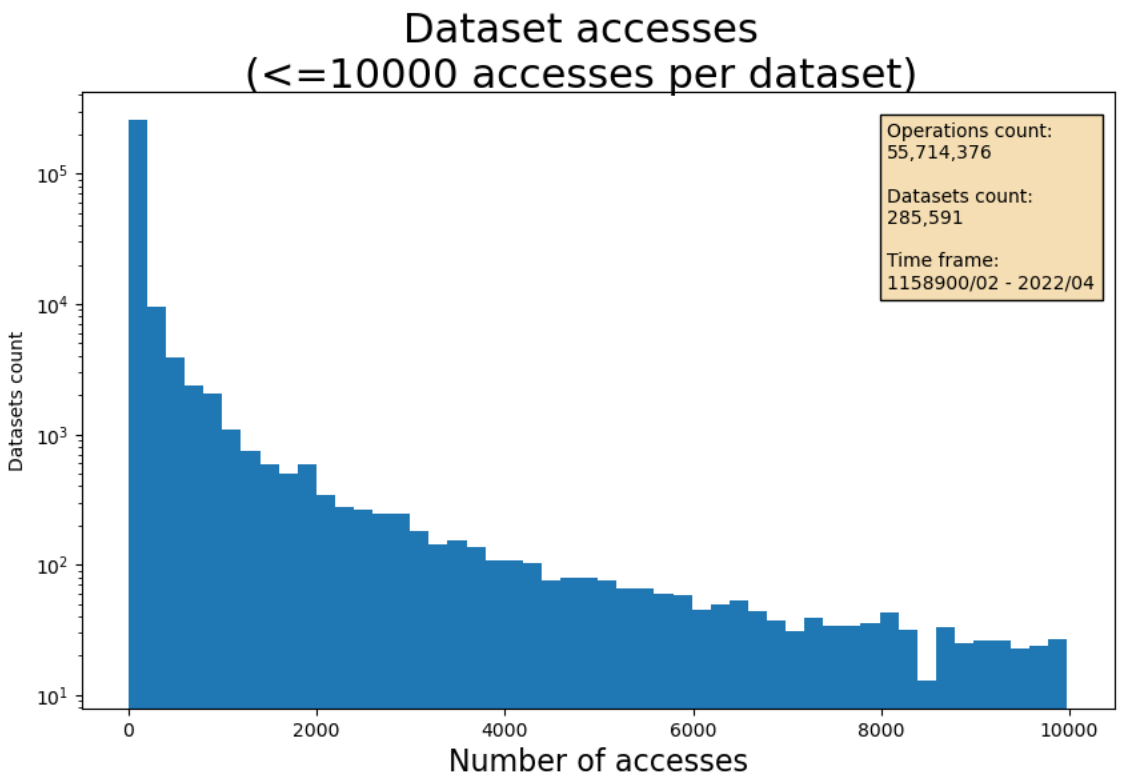

File number of accesses
Operations count:
55,714,376
Files count:
12,363,014
Time frame:
02/2022 - 04/2022
Files count
Number of accesses
0
500
1000
1500
2000
2500
3000


File number of accesses
(<=100 accesses per file)
Operations count:
55,714,376
Files count:
12,363,014
Time frame:
02/2022 - 04/2022
Files count
Number of accesses
0
20
40
60
80
100

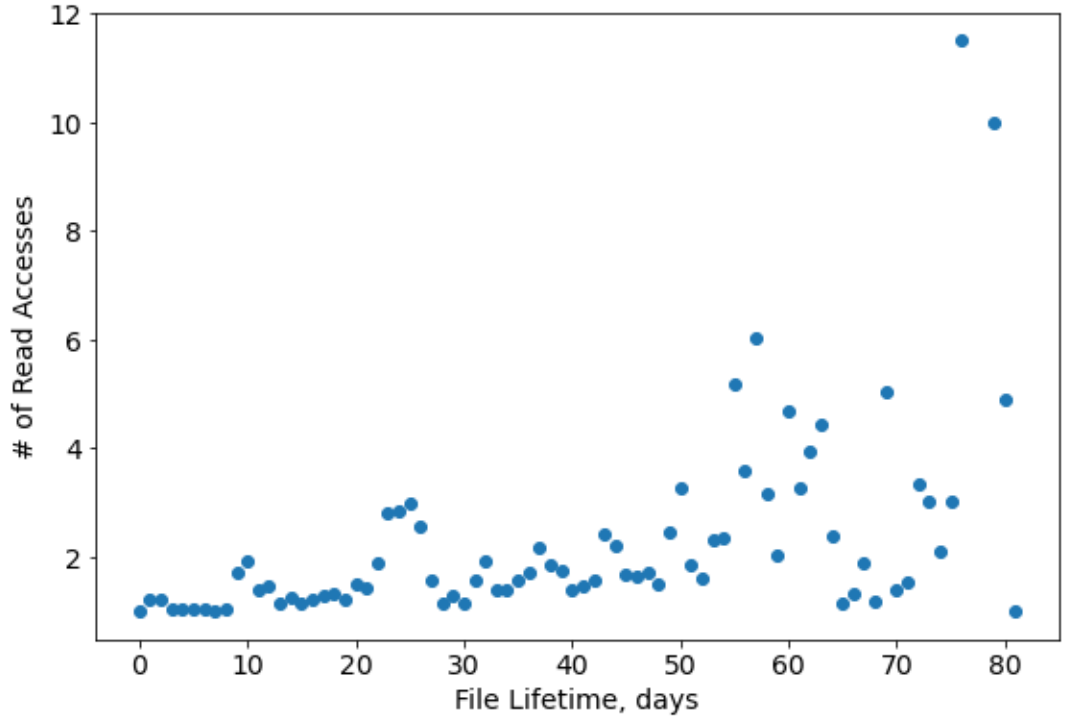

Files accessed per dataset
Files count:
12,363,014
Datasets count:
285,591
Files with no dataset:
1,158,900
Time frame:
02/2022 - 04/2022
Datasets count
Number of files
0
5000
10000
15000
20000
25000
30000
35000

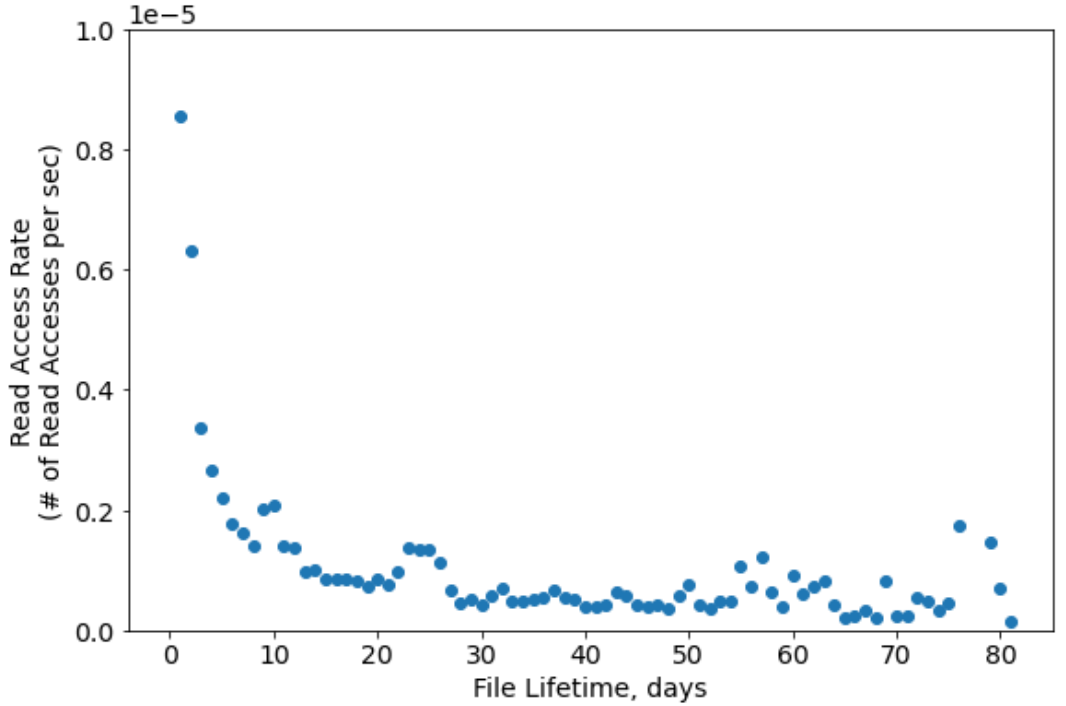

Files accessed per dataset
(<=100 files per ds)
Files count:
12,363,014
Datasets count:
285,591
Files with no dataset:
1,158,900
Time frame:
02/2022 - 04/2022
Datasets count
Number of files
0
200
400
600
800
1000


Dataset accesses
Operations count:
55,714,376
Datasets count:
285,591
Time frame:
1158900/02 - 2022/04
Datasets count
Number of accesses
0
50000
100000
150000
200000
250000
300000
350000

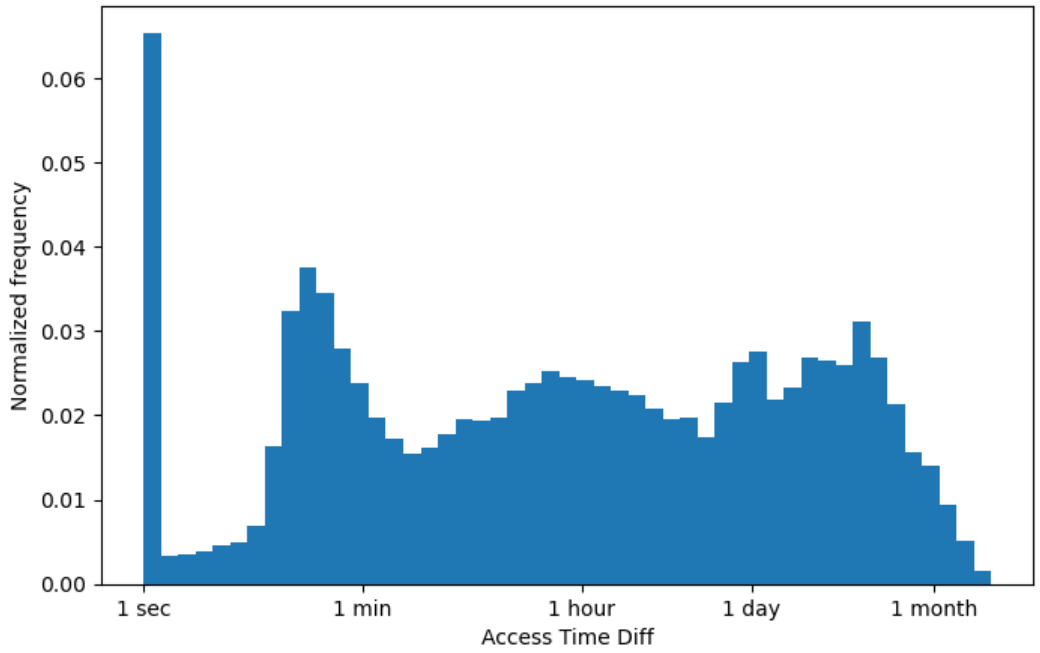

Dataset accesses
(<=10000 accesses per dataset)
Operations count:
55,714,376
Datasets count:
285,591
Time frame:
1158900/02 - 2022/04
Datasets count
Number of accesses
0
2000
4000
6000
8000
10000

Dependency between the file lifetme and the number of accesses

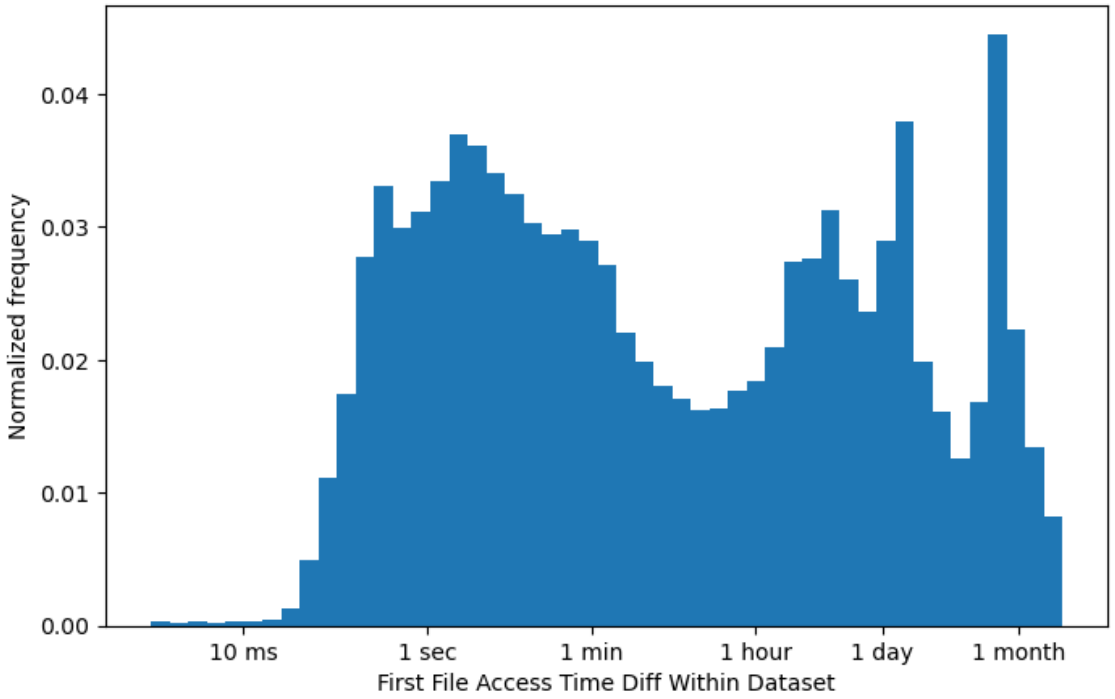




Time between consecutive reads



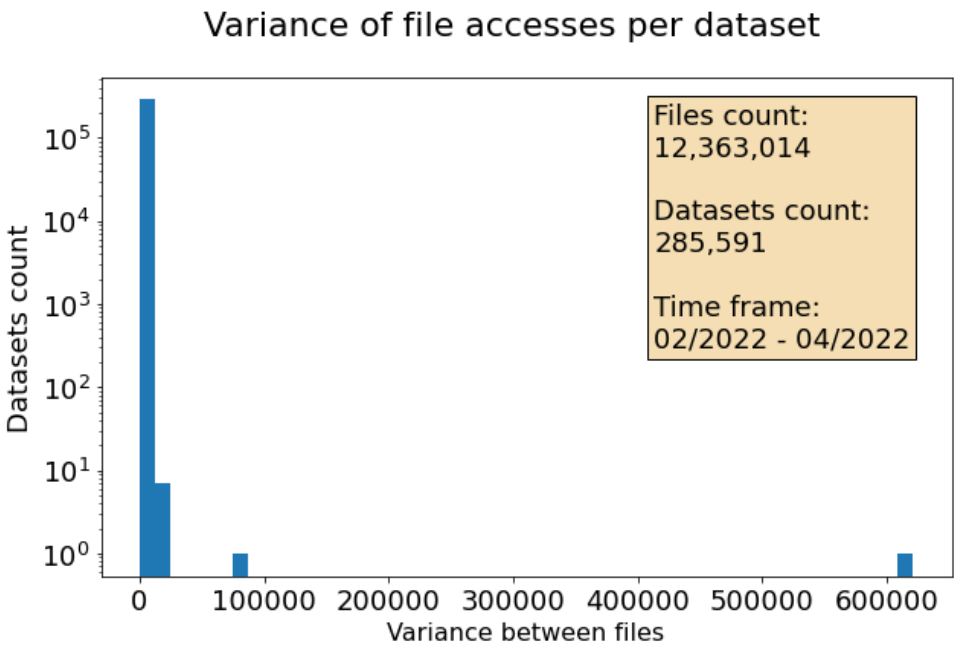
Time between the first file accesses within a dataset

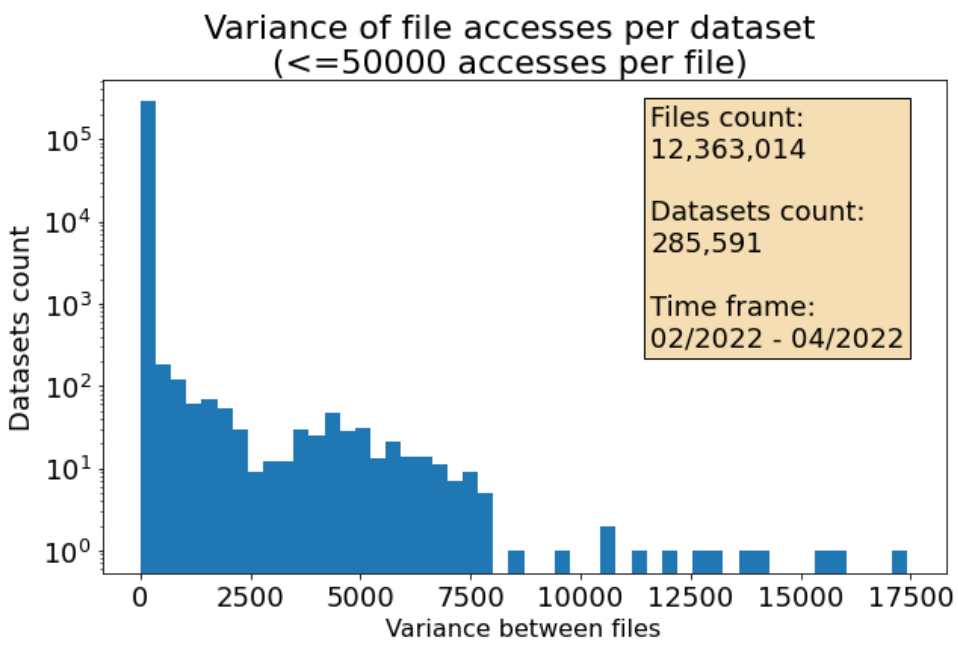


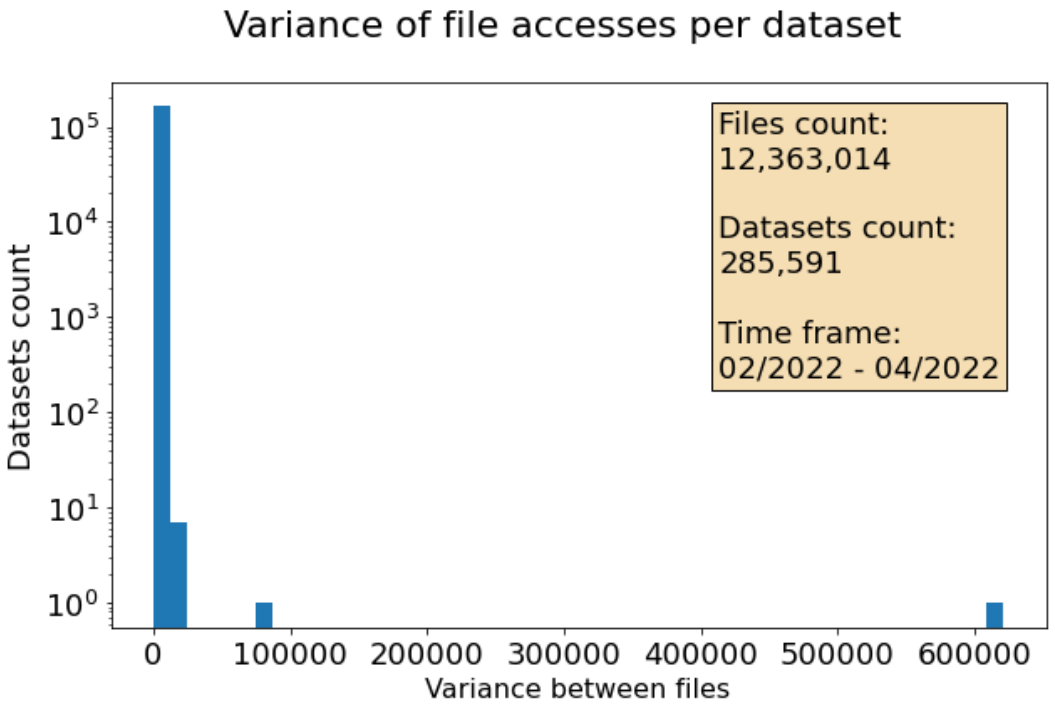
Total dataset count:
285,591

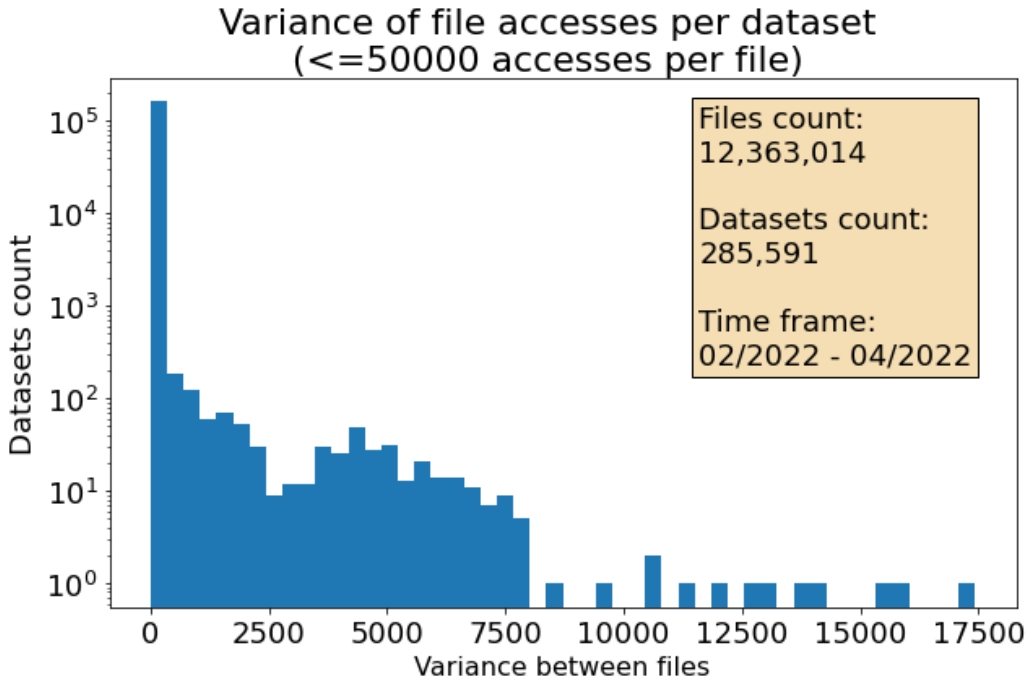
Number of dataset entries:
(excluded datasets with 1 file only)
162,825

Time frame:
02/2022 - 04/2022

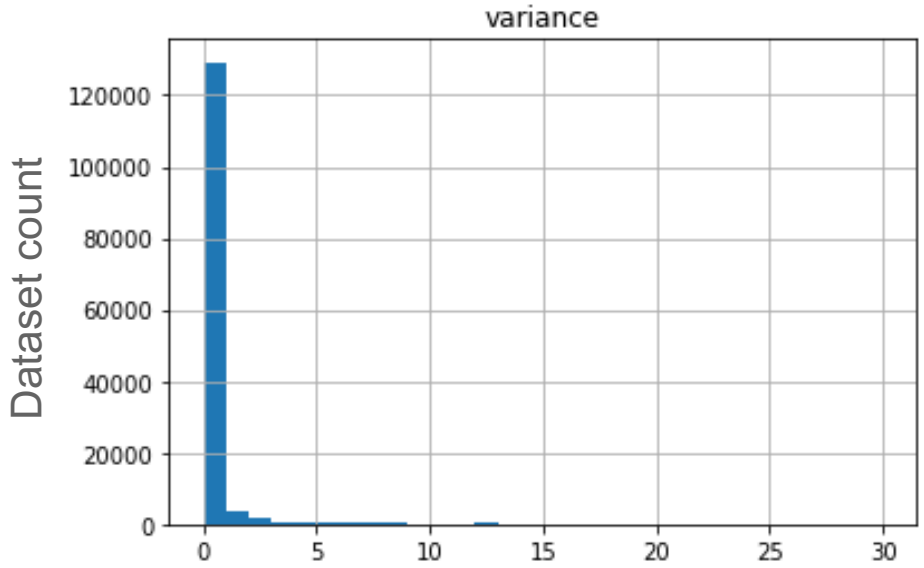

Variance of file accesses per dataset
Files count:
12,363,014
Datasets count:
285,591
Time frame:
02/2022 - 04/2022
Datasets count
10^5
10^4
10^3
10^2
10^1
10^0
0
100000
200000
300000
400000
500000
600000
Variance between files


Variance of file accesses per dataset
(<=50000 accesses per file)
Files count:
12,363,014
Datasets count:
285,591
Time frame:
02/2022 - 04/2022
Datasets count
10^5
10^4
10^3
10^2
10^1
10^0
0
2500
5000
7500
10000
12500
15000
17500
Variance between files


Variance of file accesses per dataset
Files count:
12,363,014
Datasets count:
285,591
Time frame:
02/2022 - 04/2022
Datasets count
10^0
10^1
10^2
10^3
10^4
10^5
0
100000
200000
300000
400000
500000
600000
Variance between files

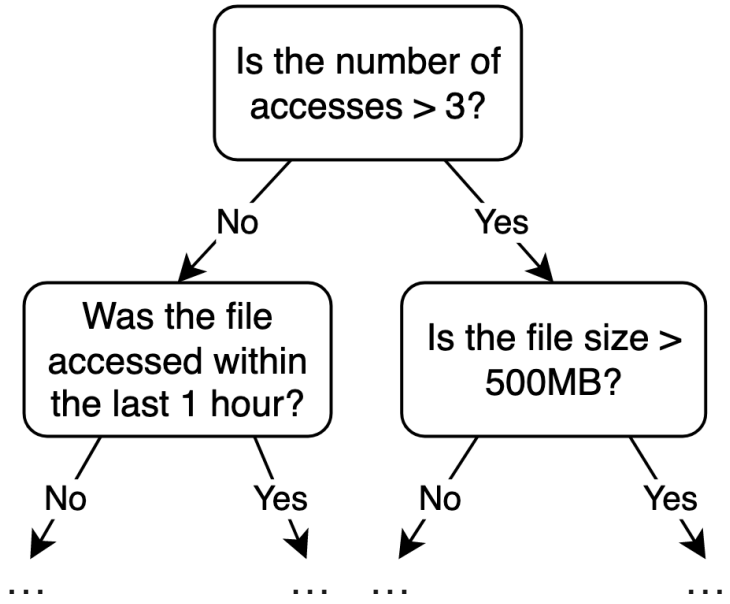

Variance of file accesses per dataset
(<=50000 accesses per file)
Files count:
12,363,014
Datasets count:
285,591
Time frame:
02/2022 - 04/2022
Datasets count
10^0
10^1
10^2
10^3
10^4
10^5
0
2500
5000
7500
10000
12500
15000
17500
Variance between files

Variance of the number of reads per dataset



Decision Trees (DT) and Random Forests (RF)

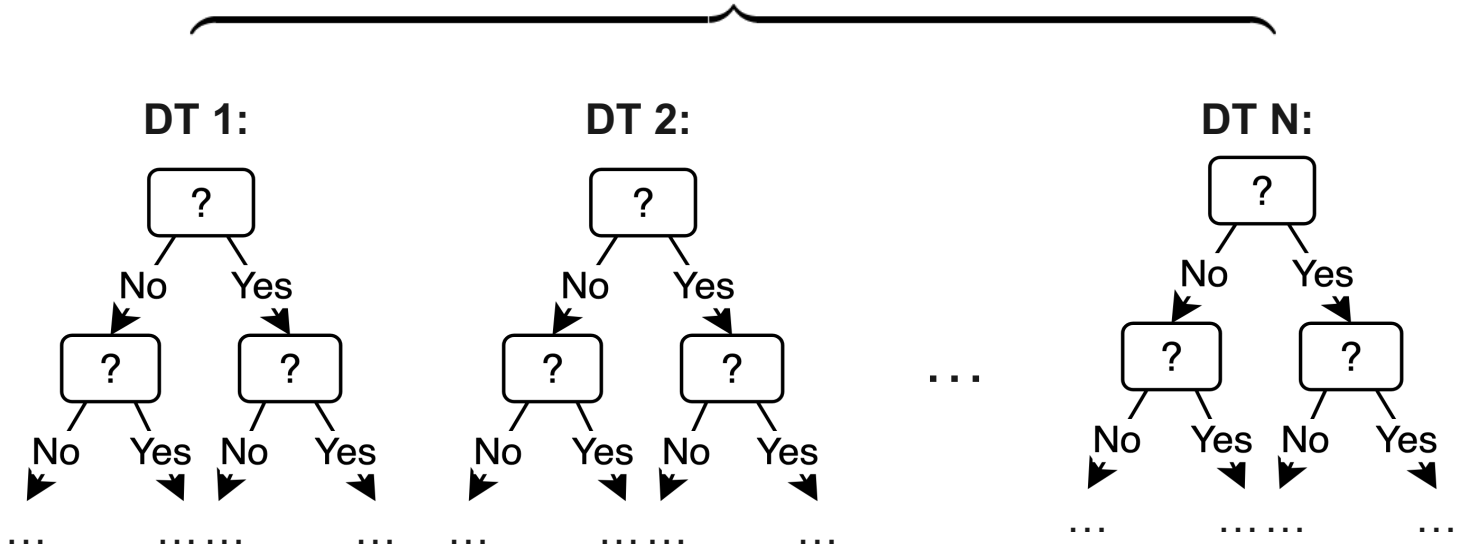
Decision Tree:



+ Classification and regression
+ Easy to interpret
- Prone to overfitting

+ Classification and regression
+ Can be parallelized
+ Solve the overfitting issue
+ Able to handle high-dimensional data and noise
- More complex to interpret

Random Forest:



Feature importance