Polytechnique Montreal

AUTOMATIC REDUCTION OF EXECUTION TRACE DATA VOLUME USING GRADIENT BOOSTING IN LARGE-SCALE MICROSERVICE SYSTEMS

AMIR HAGHSHENAS, NASER EZZATI-JIVAN, MICHEL DAGENAIS

Canadian AI 2024

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

AGENDA OVERVIEW

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

INTRO

Availability in Microservice architecture is VERY important

- Ensuring availability is essential using performance modeling.
- Tracing and logging are used for data collection.
- A question to answer: How much data is enough?
- Previous works are not suitable in this case
  - Does not consider existing trace data.
  - Not adaptable to mircoservice architecture

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

OUR CONTRIBUTION

The goal of this study is to use existing trace data to minimize the data required for performing efficient and accurate performance modeling.

01

CONSIDER EXISTING DATA

The first study to use trace data with the goal of reducing the number of features for accurate performance modeling.

02

SIGNIFICANT REDUCTION

Our approach reduced the trace data volume by about 69% without sacrificing model performance

03

COMPLEMENT TO EXISTING WORK

The outcome of our work can complement the existing models to update the tracing decision.

RELATED WORKS

Studies related to this work can be categorized into two different groups

01

ASSISTING TRACE/LOG DATA REDUCTION

Answering the question of
“Where to log?”
log2, log20, log4Perf

02

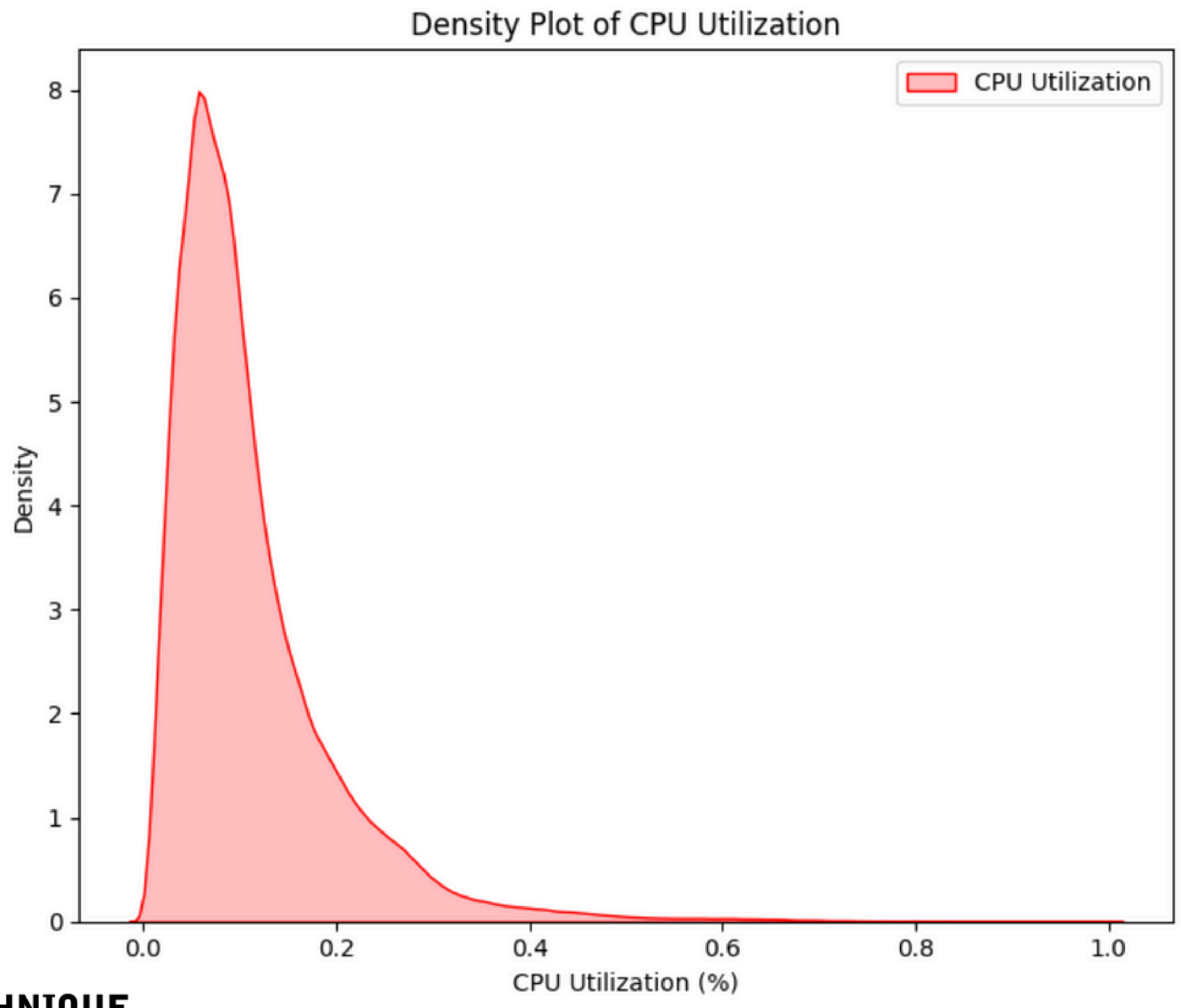
PERFORMANCE MODELING

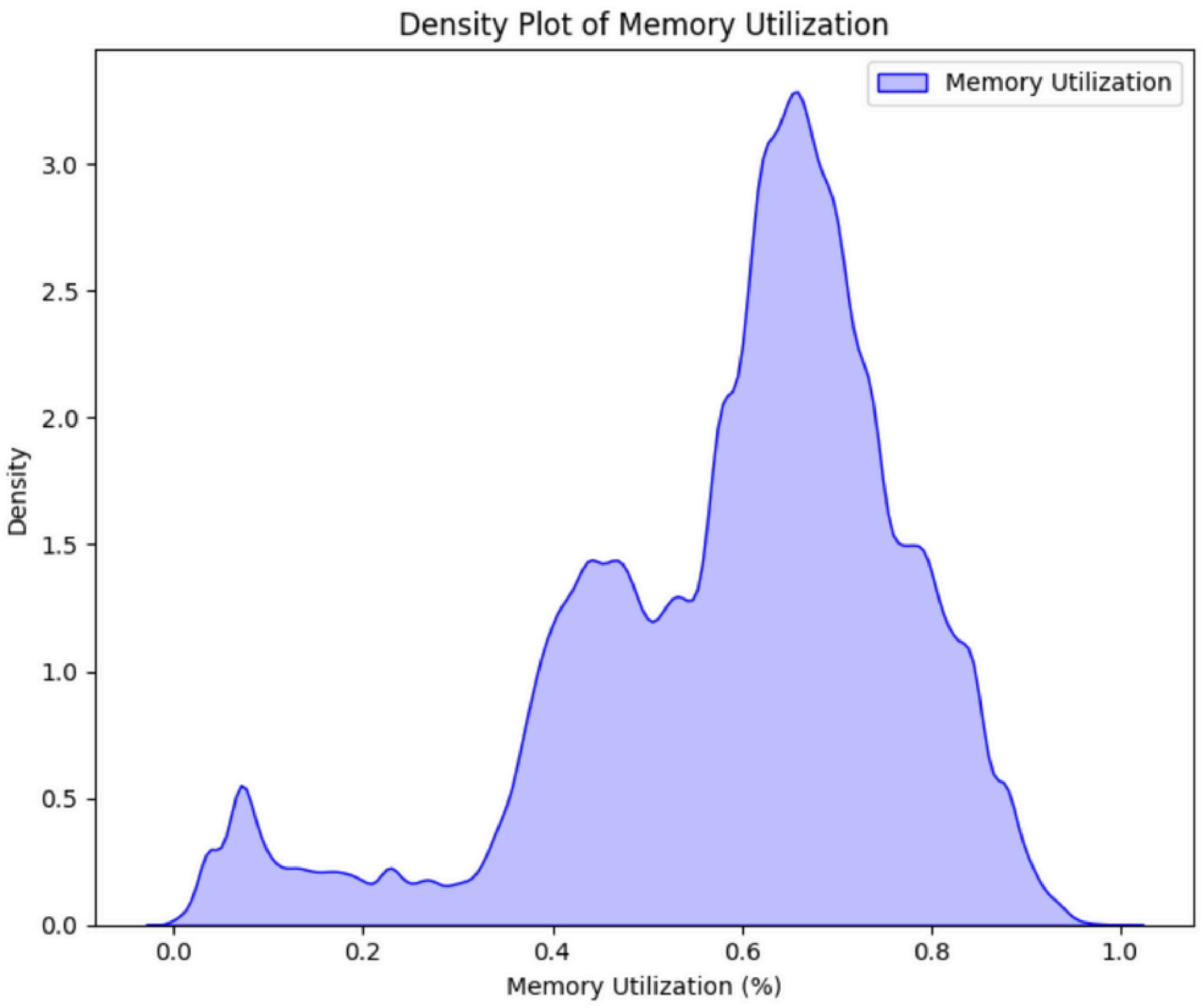
Using various techniques to model the performance of a software system for different objectives.

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

DATASET

We used a publicly available dataset containing run-time information for the Alibaba Production cluster.

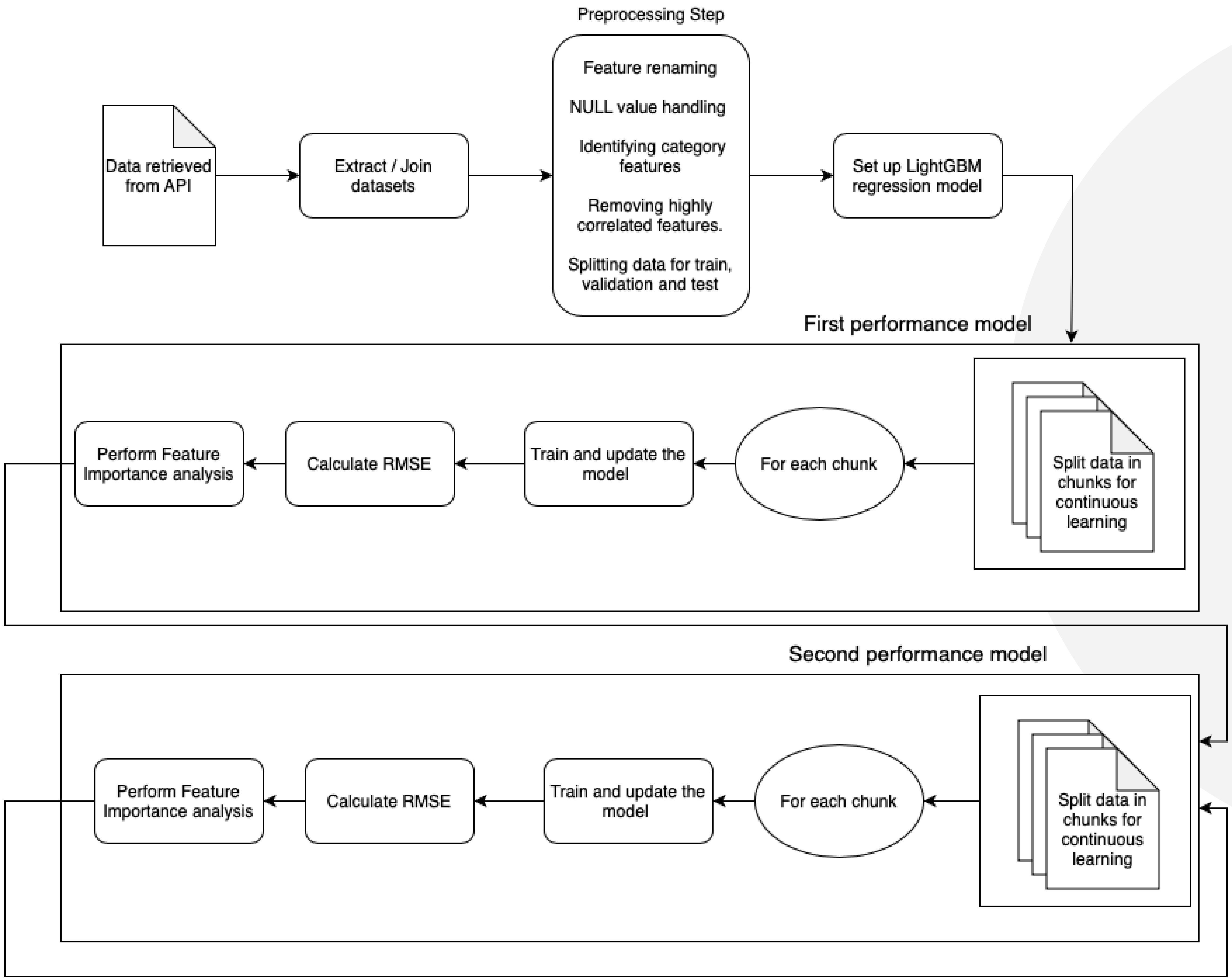




DATASET DESCRIPTION

- Includes run-time data for over 40,000 bare-metal nodes, 470,000 containers and 28,000+ micro-services in Alibaba Production Cluster.
- Gathered across 13 days in one-hour intervals.
- Contains four main sections:
  - MSResource: microservice run-time information
  - MSRTMCR: microservice call rate and response time
  - MSCallGraph: call graph interactions among microservices
  - Node: node run-time information.
- We used two hours intervals for analysis due to very large data volume.

WORKFLOW OVERVIEW



FIRST PERFORMANCE MODEL

- Due to large data volume, we used pandas capability to analyze the data in different chunks.
- Each chunk is separated into training, test and validation sets.
- We used early stopping mechanism to halt training when no improvement is observed.
- We calculated RMSE to asses model performance.
- We also calculated feature ranking based on Gain importance.

- Set objective to be Regression.
- Set learning rate to 0.1 for generalization.
- Set number of boost rounds to 5,000 for comprehensive learning.
- Set max depth to 7 and number of leaves to default to to balance complexity and performance.
- set lambda l2 regularization to 0.1 to further avoid overfitting.

SECOND PERFORMANCE MODEL

- We used the feature ranking of previous step, we ran the same model using different subsets from different interval of the dataset.
- We used the top 9, top 5 and top 3 features from the feature ranking.
- Objective is to determine the minimum number of features without losing performance.

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

RESULTS

Number of Features	RMSE CPU	RMSE Memory	Data Reduction (%)
All 29	0.08	0.14	0 (full data)
Top 9	0.02	0.13	69 %
Top 5	0.14	0.21	83 %
Top 3	0.28	0.35	90 %

FEATURE RANKING

Feature	Description
consumermq_rt	Return time of fetching message from queue
dminstanceid	Container ID of an upstream microservice(MS)
writedb_rt	Return time of writing on database
readdb_rt	Return time of reading from database
interface	Interface of call from upstream to downstream MS
http_mcr	Rate of HTTP calls
http_rt	Return time of HTTP calls
readmc_rt	Return time of reading Memcached
providermq_rt	Return time of writing message to queue
providerrpc_rt	Return time of RPC calls for provider
consumerrpc_rt	Return time of RPC calls for consumer
readmc_mcr	Rate of reading Memcached
rpc_id	Unique ID of RPC call
providerrpc_mcr	Rate of RPC calls for provider
writemc_rt	Return time of writing to Memcached
readdb_mcr	Rate of reading from Database

DISCUSSION

- Our proposed method is potentially generalizable to other applications and domains. (Special implementation is needed)
- We tried to counter potential overfitting but the results depend on accuracy of LightGBM.
- LightGBM outperforms the other algorithms.
  - Principal Component Analysis (PCA)
  - Recursive Feature Elimination (RFE)
  - Genetic Algorithms (GA)
- We used a real-world dataset by Alibaba but more testing is needed to fully evaluated the effectiveness of this approach.
  - Using the outcome on a production system.
  - Using similar datasets from other domains (not easy to obtain)

SUMMARY

WHAT?

Reducing required data for accurate and efficient performance modeling.

WHY?

State of the art work is not suitable in microservice architecture. More studies are needed.

HOW?

Performing two phase regression using LightGBM and selecting top ranked features.

RESULT?

About 69% reduction in data size with slight increase in the performance of the model and focusing on essential aspects of microservice.

THANK YOU

Amir Haghshenas
amir.haghshenas@polymlt.ca

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY