



STAC Update: Machine Learning

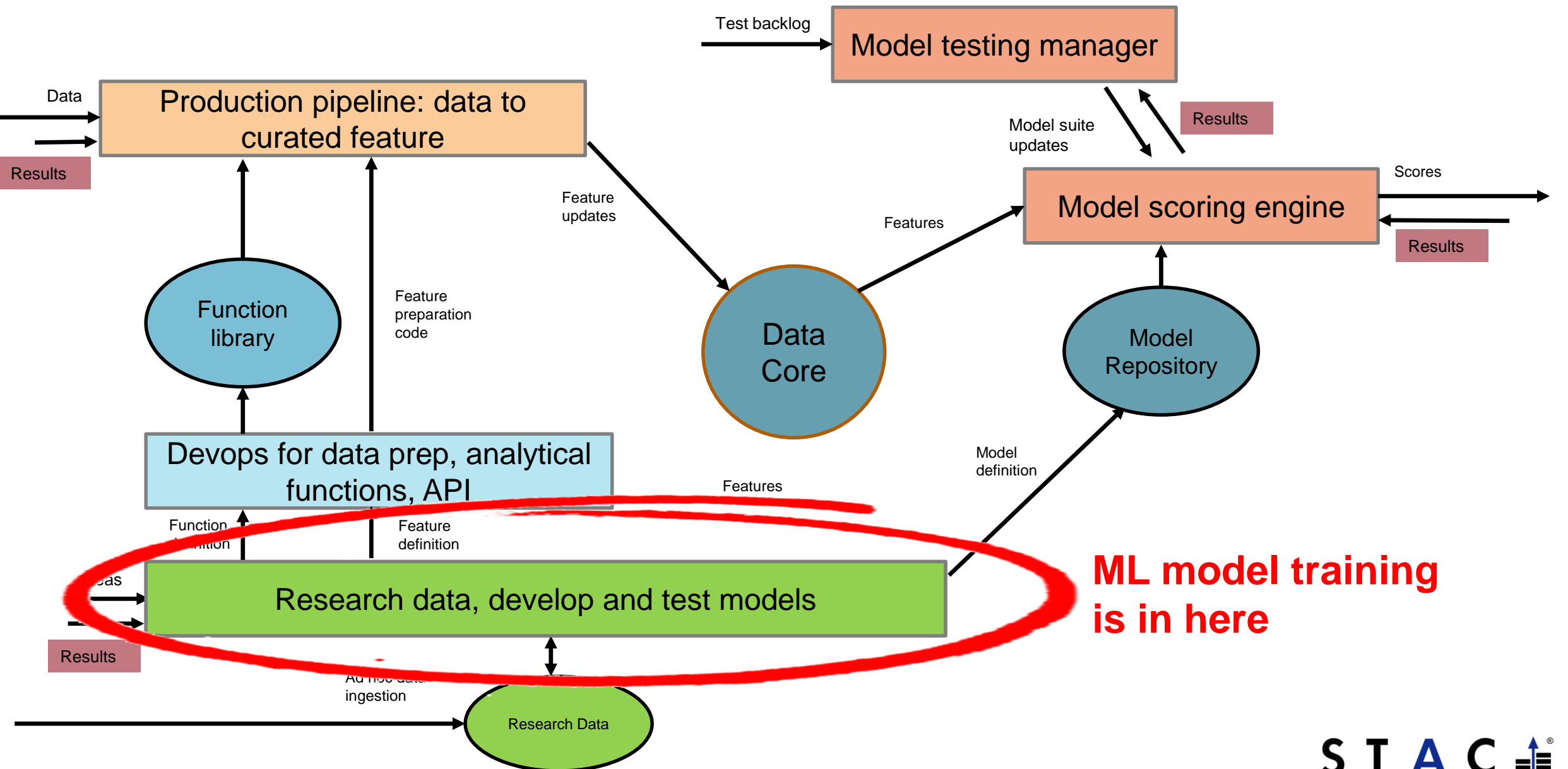
Michel Debiche,
Guest Analyst, STAC

michel.debiche@STACresearch.com

Background

- STAC is focused on helping with pain points in the data science process
- See QuantOps™ presentation from last STAC Summits
 - www.STACresearch.com/STAC-Summit-13-Jun-2018-debiche
- A key pain point: training of machine learning models

ML model training in context

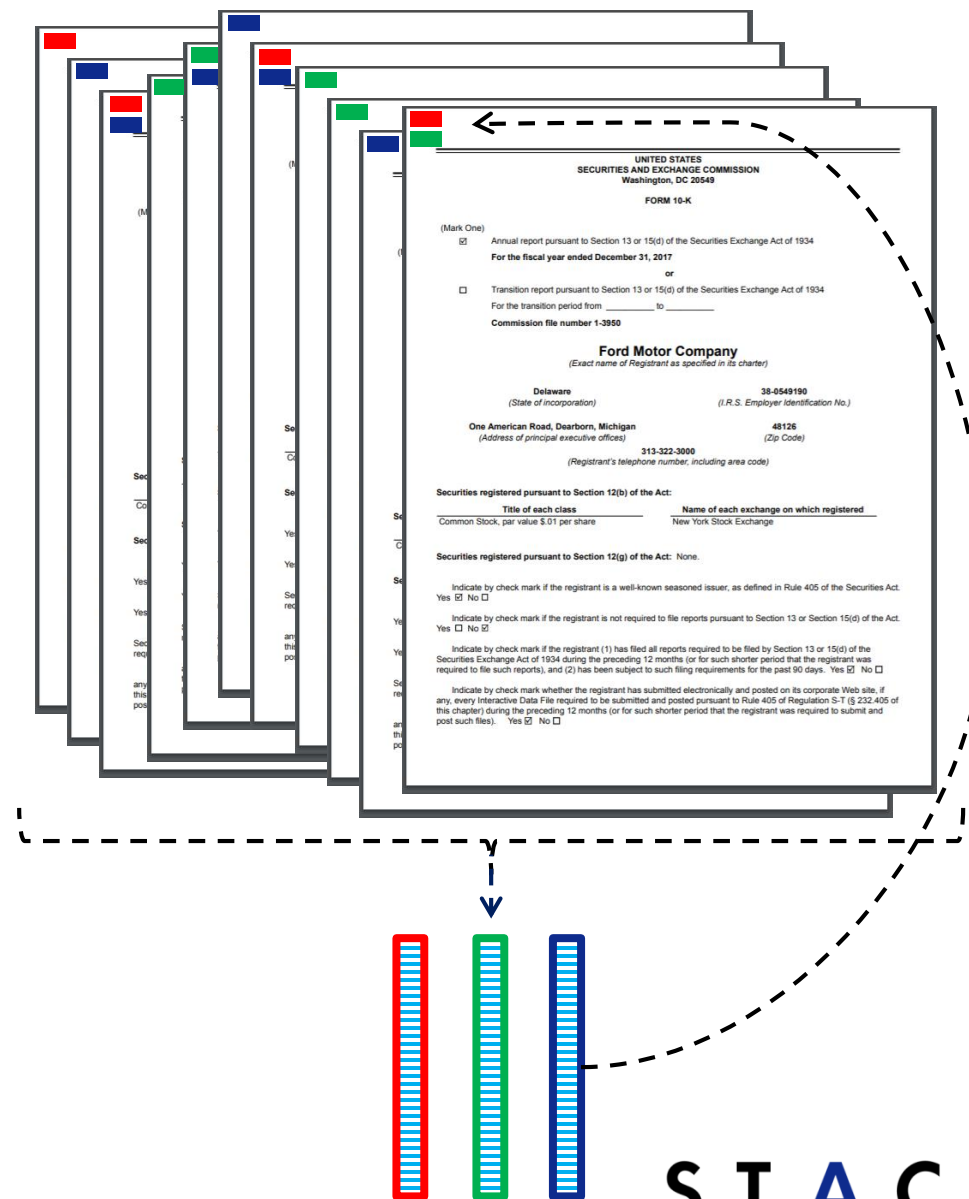


Why we built a strawman benchmark for ML

- Many Council members have asked for ML benchmarks
- Getting a group to go from a blank sheet to usable benchmarks is hard
 - Many, many degrees of freedom
- We decided to propose something for feedback
 - This is not an official Council-backed benchmark suite
- Based on discussions with user firms and projects we've been involved in

Use case

- Topic modeling of form 10-K documents (from EDGAR) using Latent Dirichlet Allocation (LDA)
- Topic models identify and quantify similarities among documents
- This is unsupervised learning
- The engineering challenge is the training phase



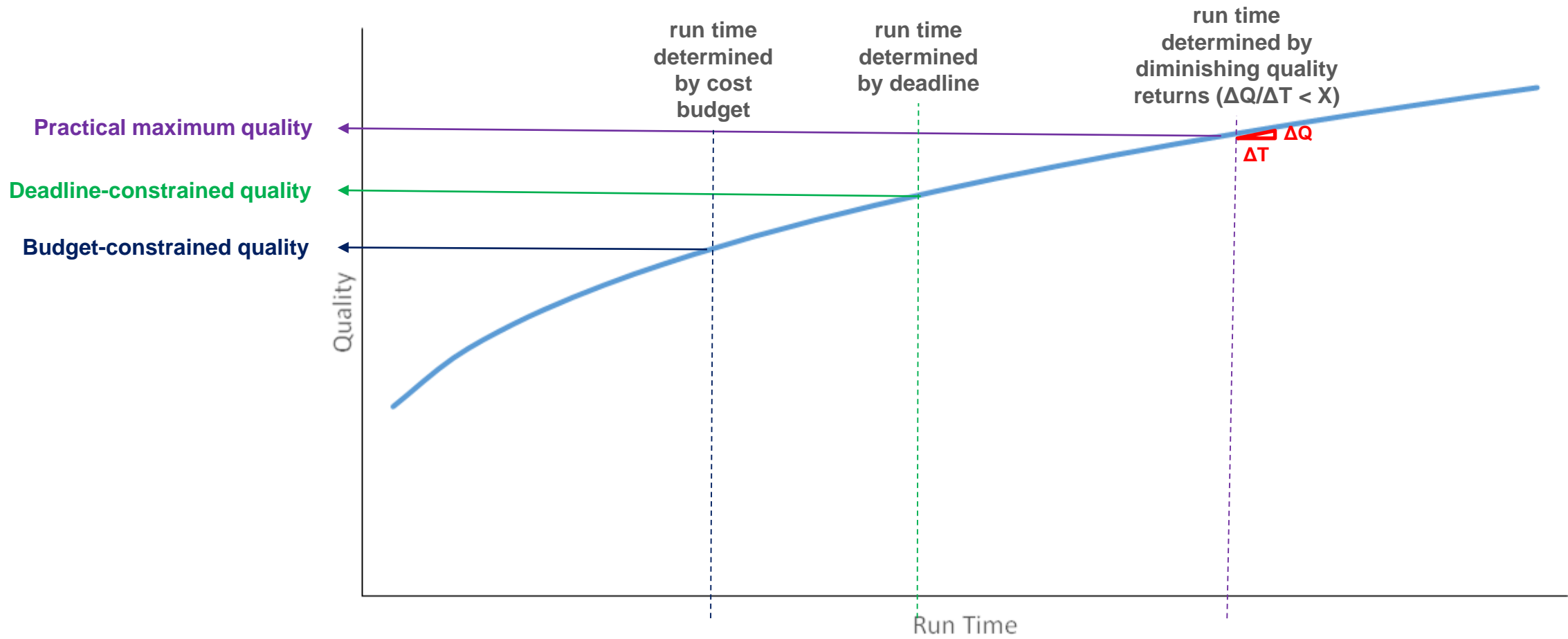
Why we chose this use case

- ML on unstructured text is a wide category of use cases
- It's an active area of academic & industrial research
- ML on 10-K forms is common in finance
- LDA is popular with “quantamental” firms thanks to explainability
- Not a toy problem, but simple enough to constrain benchmark costs
- Several implementations (e.g., Scikit-Learn, Gensim, MLLib)
- Can work on many platforms (processors, frameworks, MLaaS, etc.)
- May be relatable to other document-similarity algos (e.g., word2vec)

- Performance
 - How fast did the solution produce a model?
 - Cost
 - What did it cost for the solution to produce a model?
 - Quality
 - How good is the model the solution produced?
- If we allow all three to vary, how do we make comparisons?

Answer: Expose the tradeoffs of a each solution

Quality vs Run Time for a given Solution With example budget and model-generation deadline



Example problem

- Data scientist wants to find the combination of model parameters that yields the best quality
- Common approach is a grid search (like a parameter sweep)
- Before running a full grid search, data scientist runs subset grid searches to get a sense of parameter sensitivity
- The speed with which the system can run these small grid searches affects the data scientist's productivity
- What are the tradeoffs between cost and cycle time for different solutions?
- We ran some tests for particular cases

Initial tests

- Common to all stacks tested:
 - Scikit-Learn 0.20.0
 - SpaCy 2.0.12
 - RHEL 7
 - GCP
- Common to all workloads:
 - Quality (fixed the parameters to be searched)
- Varied:
 - Python distribution
 - Instance type

Findings: Python distributions

- Tested a single experiment on an n1-standard-16 instance
- First distro: Anaconda Python 3, MKL 2019
- Second distro: Intel Python 2018, Intel MKL 2018
- Non-Intel Python took 19% more time to complete
 - That is, it cost 19% more than Intel Python
- We standardized on Intel Python for the rest of the project

Findings: Threading

- *MKL_NUM_THREADS* parameter of MKL library faithfully utilizes nearly 100% of all cores on a platform
- Unfortunately, the overhead with this LDA implementation exceeded the benefit
- Best performance was a single thread per experiment
- So set *MKL_NUM_THREADS* environment *variable to 1*.

Findings: Multi-processing

- *n-jobs* parameter of Scikit-Learn LDA model uses multiple processes to parallelize the separable part of the algorithm
- Unfortunately, the overhead exceeded the benefit
- If combined with default MKL setting, contention for cores made performance even worse
- So, set *n_jobs* parameter to None (or leave default in place)

Findings: Scaling with parallel processes

- n1-standard-16: 26.22 minutes
 - ~ 25,000 vCPU seconds
- n1-standard-96: 8.75 minutes
 - ~ 50,000 vCPU seconds
- 3x speedup from 6x the vCPUs. 2x the cost.
- If your goal is to minimize IT operating cost, then n1-standard-16 wins
- But if your goal is to maximize data scientist productivity, then n1-standard-96 wins
 - <10 minutes generally thought short enough to allow interactive exploration

Key findings

- Intel Python seems to offer higher speed/lower cost with this workload
- Scikit-Learn LDA works best with one thread per experiment
- Process-level parallelism does not scale linearly with vCPUs
- Scaling up is still desirable if cycle time (hence data scientist productivity) is at a premium

Next steps

- Look at scale out vs scale up
- Write up initial project
 - Next few weeks
- Form working group
 - Provide feedback on this use case
 - Propose other key use cases
- Test more implementations, more platforms