Machine Learning at Scale

Srijith Rajamohan, Ph.D.

Sr. Developer Advocate (Data Science/ Machine Learning)

databricks

The Databricks Ecosystem for ML

DELTA LAKE

APACHE Spark™

Databricks Notebooks

Feature Store

mlflow™

Delta Share

Databricks AutoML

Questions for Scalable ML

- Track the provenance and reason for model creation
- What training data was used, if any?
  - Proprietary data, sensitive data, storage, data retention period?
  - Real-time or batch?
- How are the models being used and who is using it?
  - Exploratory analysis and production environment?
- Is model performance being measured regularly and is the model being updated?
- Is the model well documented to ensure reuse?
- Is the model deployment process being automated?
- Institutional adoption and support

The ML Team

- Subject Matter Experts (SMEs)
  - Deep understanding of business problems
  - Need to be integrated into the ML lifecycle
- Data Scientist
  - Needs modeling skills
  - More importantly need communication skills!
- Data Engineer
  - Manages the efficient flow of data
- ML Engineer (can also be split between ML Architects/ DevOps)
  - Design, manage and scale the ML infrastructure (ML Architect)
  - Deploy and operationalize the models, ensure availability (DevOps)

Framing the problem

Operationalize the model

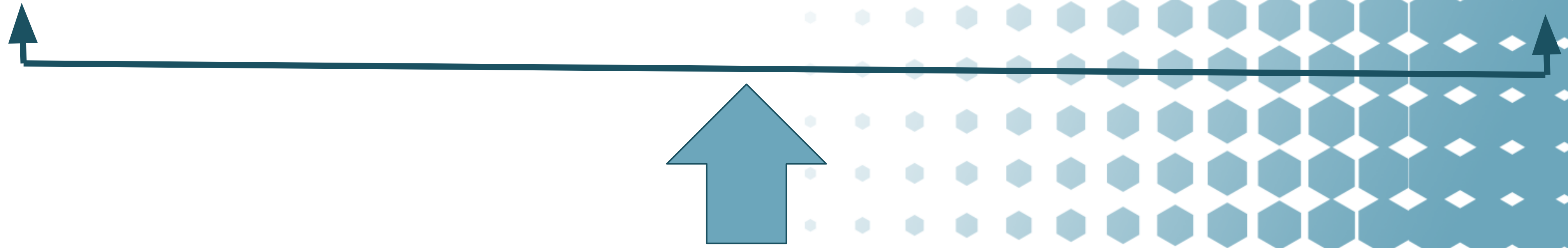
Best Practices for ML

- Software engineering practices
  - Code quality best practices
- Validate your data
  - Ensure proper data types and format are fed to your model (Schema validation)
  - Ensure no data drift, can render a supervised model ineffective
- Version and track your experiments like code!
  - Changing hyperparameters, inputs, code etc.
- Monitor predictive performance over time
  - Ensure model performance does not degrade over time
  - Ensure model fairness across different classes of data (bias)

What is MLOps?

MLOps = ML + DataOps + DevOps

Build -> Test -> Deploy -> Monitor -> Feedback -> Build



Model management

The Model Build

What should happen at this stage?

- Build and verify the model with test data
    - Ensure you are solving the right problem
    - Estimate the required performance metric (accuracy/F1 etc.) and choose the model with the lowest complexity
    - Address fairness and reproducibility at this stage of model design
- Make sure functional and unit tests pass for all modules
- All artifacts are version controlled and saved
    - An artifact is all the code and associated model, config or data files, environment and documentation
    - Use tools such as MLflow for saving these artifacts

Data

- Is data available?
  - Provenance - Openly available or proprietary data?
  - Static data or continuously updated data?
    - If not static - is it processed as batch or in a streaming fashion?
    - How is the data going to be stored and versioned?
- Storage
  - Where is the data stored?
  - How do you move it to the training stage?
- Data quality - Is it reliable or is denoising required ?
- Does the data have to be labeled?

Data

- What features are important in the predictive process?
  - Consider using a feature store
    - Feature discoverability and reuse
    - Track lineage of data used for the predictive modeling process
- Compliance
  - Removing protected features and identifiable information
  - Account for bias
  - Complying with GDPR/CCPA regulations
- Data access
  - Access control for the data

Training

- Infrastructure (CPU vs GPU) for online or batch training
- Online training
  - Incrementally updated with new data
  - Useful in a streaming data scenario
  - Not stateless
- Need for retraining
  - Model can get outdated, need to be retrained with new data for accurate predictions or inferences
- Cost of retraining
  - Frequency of retraining needs to be balanced with the cost of retraining

Model Training and Management

- Systematic training/experimentation (MLflow)
  - Track variables, hyperparameters, artifacts (MLflow tracking)
  - Version control models (MLflow model registry)
  - Enables reproducibility
- Hyperparameter tuning
  - Tune the hyperparameters to identify the best performing model
    - Not necessarily the model with the best metric
    - Generalizability and model inference time
- AutoML - Identify the best class of modeling techniques, along with their hyperparameters

Distributed Training

Data parallelism

- Data is divided among the different nodes
  - Entire model is copied to all the nodes
- Gradients are communicated back to all other nodes to update the model
  - Synchronous or asynchronous updates
- Model size is a concern

Model parallelism

- Model is divided among all the nodes
- Only works if you can take advantage of task parallelism in the model
- Model size is less of a concern

Deep Learning Synchronization

Model parameter server

- Central servers hold all shared parameters
- Workers receive updates from the central server
- Harder to scale
  - Speedup now depends on the overhead of communication with the central server

All-reduce

- All the machines store the shared parameters
- No central server
- Several architectures for this
  - Ring All-reduce
  - Tree All-reduce

databricks

Other Topics in Training

- Quantization-aware training
  - Lower-precision training to minimize memory/compute requirements
- Federated learning
  - Decentralized learning with the Federated Averaging algorithm (Google)
  - Keep data on device
  - Model is updated with data on device and updates sent back to central server
  - Updates from all devices are averaged
- Privacy-preserving learning
  - Learn from data that is encrypted or with minimal exposure to the data

Reproducible ML

- Document assumptions about data, the problems that are being solved
- Algorithms need to be reproducible
  - Apart from the obvious reasons, needed for governance/compliance
  - Almost a necessity for debugging and identifying issues with the model
- Data versioning
  - Use frameworks such as Delta Lake for data versioning
  - Use a feature store to track and save features
- Log the environment
  - Source code
  - Libraries
  - Infrastructure
- Design and log the output metrics for reproducibility

Explainable AI (xAI)

- Why do we have a certain output Y for an input X? (No black boxes)
- What features caused this result?
  - Shapley values
  - LIME
  - Permutation Feature Importance
- Compliance
  - In certain domains such as the Financial industry, Neural Networks cannot be used
    - Results not explainable, ML algorithms more interpretable compared to DL
  - GDPR : What logic is used? What process is used? What is the impact of the decision made by the algorithm?

Model Testing and Preparation

The Handoff

- Data Scientist - > Data Engineer/ML Engineer
  - Minimize the amount of work for the Data/ML Engineer
- Convert the model to be production ready
  - The correct deployment format
  - Reduce the model footprint
    - Use model quantization, pruning and distillation as appropriate
  - Account for power considerations, i.e. model used in mobile devices
  - Account for latency considerations, i.e. real-time models
- Ideally, all of these should be taken into account during the model design stage

Validation

- Make sure the production-ready model performance is sufficient on test data
  - Test data should be representative of real-world distributions
  - It should also cover extreme events that are unlikely to occur
  - It should also be tested on a sample of current production data
- Ensure that computational performance is also tested
  - Compute and memory usage
  - Solution time
- Establish well-defined checks for model performance with assertions
- Assess model fairness (Governance)

Portability

- Portability and reproducibility go hand-in-hand
- Formats such as ONNX, POJO, MLflow projects etc.
- Use reproducible environment files
  - conda or virtualenv environments for Python
- Containers
  - Docker, Singularity etc.
  - Docker containers can be deployed to Kubernetes clusters
    - Easily and automatically take care of scaling and load management (Elastic systems)
- Ansible/Terraform/Puppet/Chef playbooks for reproducible VMs

Some Critical Issues in ML...

databricks

Governance

- Financial, Legal and Ethical governance
  - Industry-specific regulation regarding use of models (Process & Data governance)
    - Account for bias, lineage, data integrity (ALCOA), explainability and reproducibility
- All data privacy regulations such as GDPR and CCPA apply to ML
  - Lawfulness
  - The purpose of data collection
  - Data minimization
  - Accurate representation of data
  - Storage limitation
  - Confidentiality
  - Accountability

Data Governance

- What is covered under Data Governance?
  - Availability of data at every point in the ML lifecycle
    - Encourages data sharing/ reuse, eliminate duplicate work
  - Is the data usable?
  - Data consistency: Point-in-time, transaction and application consistency
  - Data integrity: Increase confidence in decision-making
  - Data access and security: Compliance
  - Accountability: Who is responsible for the data quality?
- Data Governance for defense or offense
  - Defensive activities are designed to minimize risk (E.g. compliance)
  - Offensive activities are revenue-generating (E.g. modeling for churn)

What is Bias?

- Bias is discrimination based on an (legally-protected) attribute or class
  - Race, age etc.
  - Could be the result of the bias in data (improper sampling), or improperly framed problem statements (not accounting for bias when there is limited data), or using features that have protected-class status
- What is fairness?
  - Not necessarily just legal regulation
  - Public opinion can define what is fair
  - The notion of fairness evolves over time
- Intentional misuse of ML or misplaced use of ML
  - E.g. Cambridge Analytica (Fairness) and recidivism assessment (Bias)

Responsible ML

- Transparent, accountable, fair
- Should be able to explain how various features affect the predictions
  - Shapley values, Sensitivity analyses etc.
- Should be representative and not adversely affect a particular class
  - Needed for regulatory purposes
  - Can be unlawful to use protected features
- Make sure worst-case scenarios are assessed (adversarial attacks)
- Understand risks regarding safety when making critical decisions
- **Understanding the limitations of the model and interpreting the results correctly**

What is Trustworthy AI/ML?

- Organizations worldwide have resorted to providing guidance or drafting regulations for ML systems
- According to the EU, any system or model using ML can be considered trustworthy if
  - It has continuous human oversight
  - It is a robust algorithm and implementation
  - It complies with all data privacy regulations
  - It accounts for fairness, mitigates bias and is explainable
  - It works towards the betterment of the target audience
  - There are measures for accountability such as data lineage tracking, data integrity and model reproducibility

Governance Measures

- The measures should be proportional to the
  - Level of risk
  - Impact of the outcomes
  - Extent of the impacted audience
- Extremely sensitive decision making pipelines require a thorough governance process
  - Every component of the modeling pipeline needs to be examined
  - Training algorithms, reproducibility, explainability, documentation, testing coverage, infrastructure fault-tolerance, model monitoring strategies
- Data Stewards ensure controls for the organization
  - Metadata, access control, rules, storage, data models

Model Deployment

CI/CD Pipelines

- Pipeline complexity depends on a variety of factors
  - Startup or enterprise?
  - Number of team members?
  - Infrastructure budget?
  - Regulatory needs?
  - Criticality of the decisions made by the models
- Frequency of deployment
  - Deploy frequently in smaller increments
- Deployment strategies
  - Blue/green - deploy the new model along with the old one, when the old new model is verified shut down the old one
  - Canary deployment - a portion of the data is redirected to the new one, both models are live now

Infrastructure

- Where is the model deployed?
- Self-serve or managed?
  - High-engineering costs for some self-serve options
- Kubernetes - scalable enterprise-grade frameworks
  - Can be self-serve or managed (e.g. Seldon)
- Model serving using the ML framework serving mechanisms
  - E.g. Tensorflow serving, Pytorch serve
- Lightweight open-source serving frameworks
  - MLflow, BentoML etc.
- Data Science platforms
  - Managed MLflow on Databricks

Model Deployment and Serving

- Assess the scale of the problem
  - One-off, short-lived and light weight
    - Use frameworks like Flask or Dash
  - Mission-critical/Robust
    - Ensure deployment standards are met
    - Test the deployment in a production-like environment
    - Test the model accuracy
    - Check for governance guidelines
    - Stress-test the application

Model Monitoring

What is Model Monitoring?

- Ensure that the models are performing as intended
  - ML models are probabilistic models unlike in software engineering
  - Harder to come up with tests that verify ‘as intended’
- Critical to long-term viability of ML models
- Need to have the infrastructure and policies in place to do this

Model Monitoring

- DevOps
  - Assess the model performance in terms of
    - Speed
    - Memory
    - Compute resource usage
- Data Science metrics
  - Model degradation from model drift, i.e. incorrect/unexplainable/biased results
    - Assumptions about the data distribution are no longer valid
    - Assumptions about the relationship between the input data and the output labels are no longer valid
- Business concerns
  - Is the model delivering value?
  - Is the value delivered by the model outweighing the costs

Model drift

Covariate drift (Data drift)

- Change in the distribution of data
  - This can change the distribution of the features derived from the data
- In anomaly detection
  - The distribution of credit card spending might have increased over time
  - Can no longer use the same threshold for fraud detection
  - The same threshold will result in false positives

Concept drift

- Change in the relationship between the input and output
- In anomaly detection
  - If the data distribution has not changed
  - Fraud is committed at charges 'x' < T but in several transactions 'n' for a total of 'nx'
  - The definition of fraud is now different, or the concept now has to redefined

Detecting Data Drift

- Drift can be the result of
  - Non-stationary phenomena
  - Improper sampling of the data for training the model
- How do we identify data drift? (Identify distribution changes)
  - Kolmogorov-Smirnov tests
    - Non-parametric tests
    - Tests if the samples are from the same distribution
  - Chi-square tests
    - Tests for categorical features
    - Frequency of test features should match the frequency of the train features
  - p-values in Frequentist statistics or Bayesian p-value
- Concept drift is harder and requires domain expertise

Model Monitoring for Updated Models

- Shadow testing
  - New model deployed into production alongside the old model
  - The new model also predicts the results for an input, but these results are not returned to the user
  - The results of the old and new models are compared
- A/B testing
  - The input is split between the new and old models
  - Ons set of users get Model A, the others get Model B
  - The results of either model are returned to the user

Level of sophistication in MLOps

Manual

- Manual EDA, training and validation
- Commit code to repository
- Integrate code into an existing ecosystem
- Deploy and monitor

Continuous Model Delivery

- Automatic model training and tuning
- Data statistics automatically generated
- Features from data stored in Feature store
- Model registry to store models
- Modularized code and standardized data

Continuous Integration/ Continuous Delivery

- In addition to automated model building, automated pipeline builds
- Pipeline components are automatically built and tested
- Pipeline package store holds all the verified pipelines.

Wrapping Up

- MLOps is an evolving field
  - Covered only a small subset of this vast field
- Modeling is only a small part of ML in practice
- The needs of an enterprise are vastly different from that of a small research group
  - Assess your needs carefully

databricks