
MLOps
Accelerator

What is MLOps

MLOps (Machine Learning Operations) is a set of practices that combines deployment of machine learning prototypes and IT operations. Its purpose is to shorten the deployment life cycle and ensure machine learning models are production ready using standardized processes.

MLOps also provides for continuous training and monitoring of Machine Learning models

SDK

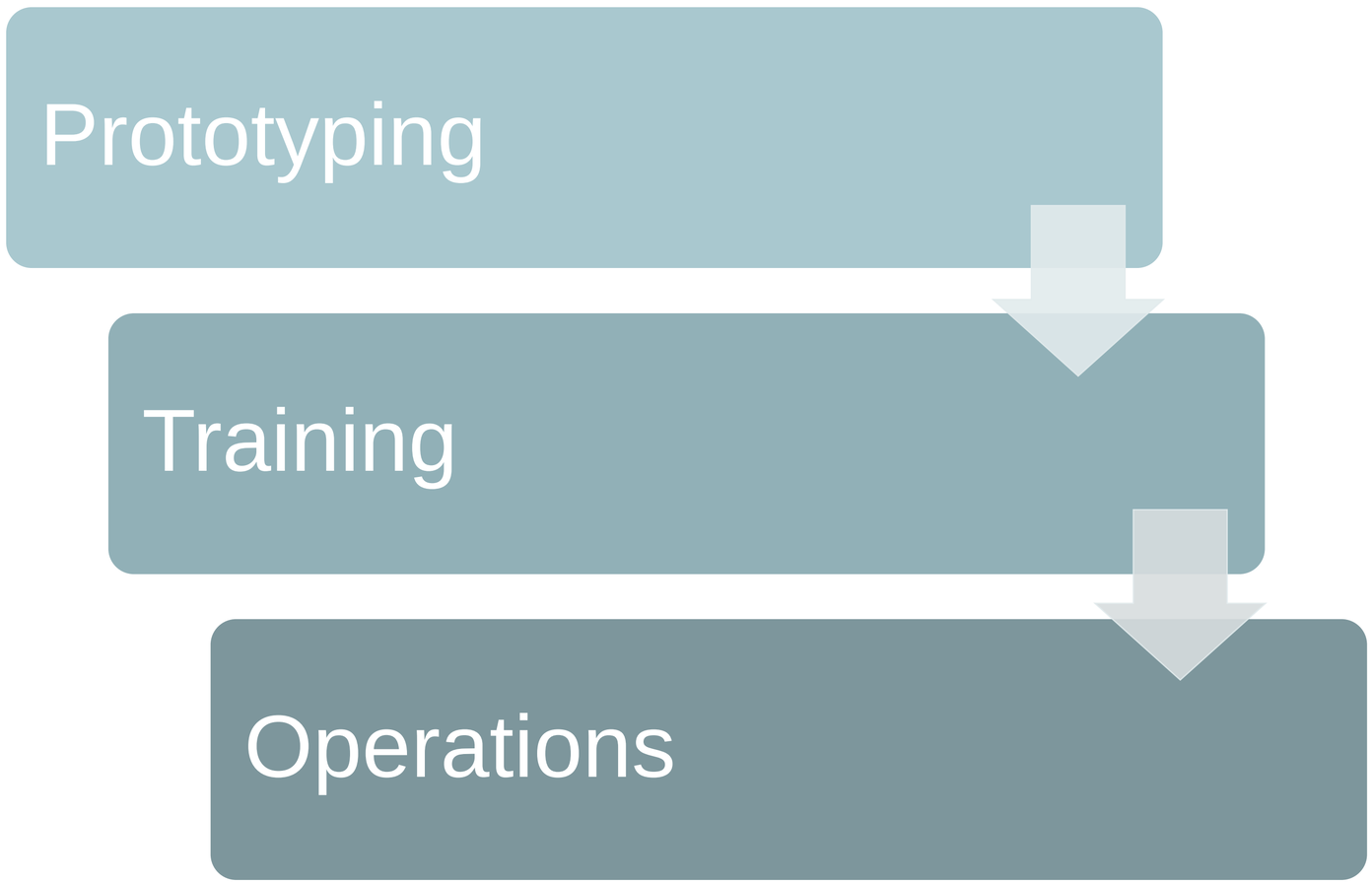
Key Challenges

- Converting the Models to run at Production volumes (including related ETL)
- Training the Models using Production volumes
- Storing Model Outputs in a usable and accessible format
- Evaluating, Tracking and Registering Production Models
- Deploying the model and related data transformations in a repeatable method
- Monitoring and re-training the models once in production
- Integrating the model into Business Processes

High level Phases

Initial Implementation

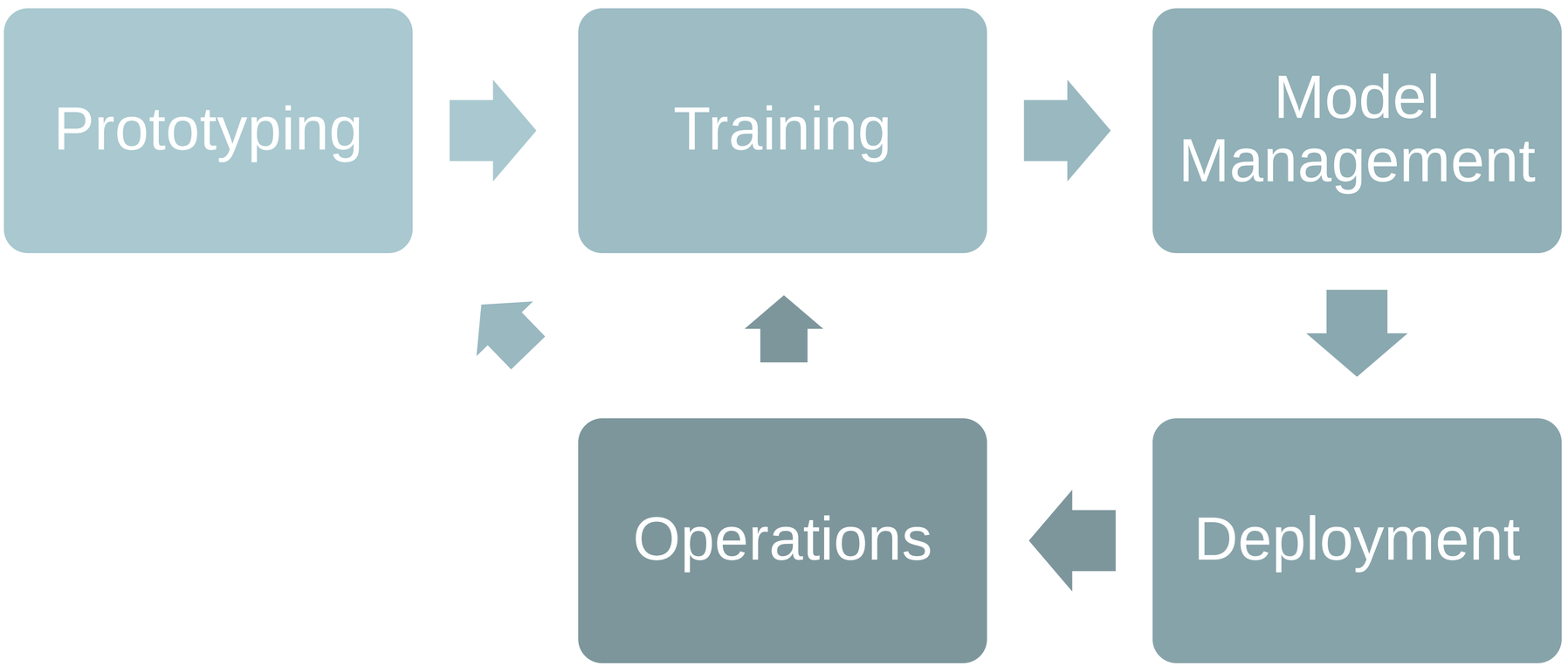
How most people envision a machine learning project



High level Phases

Actual Implementation

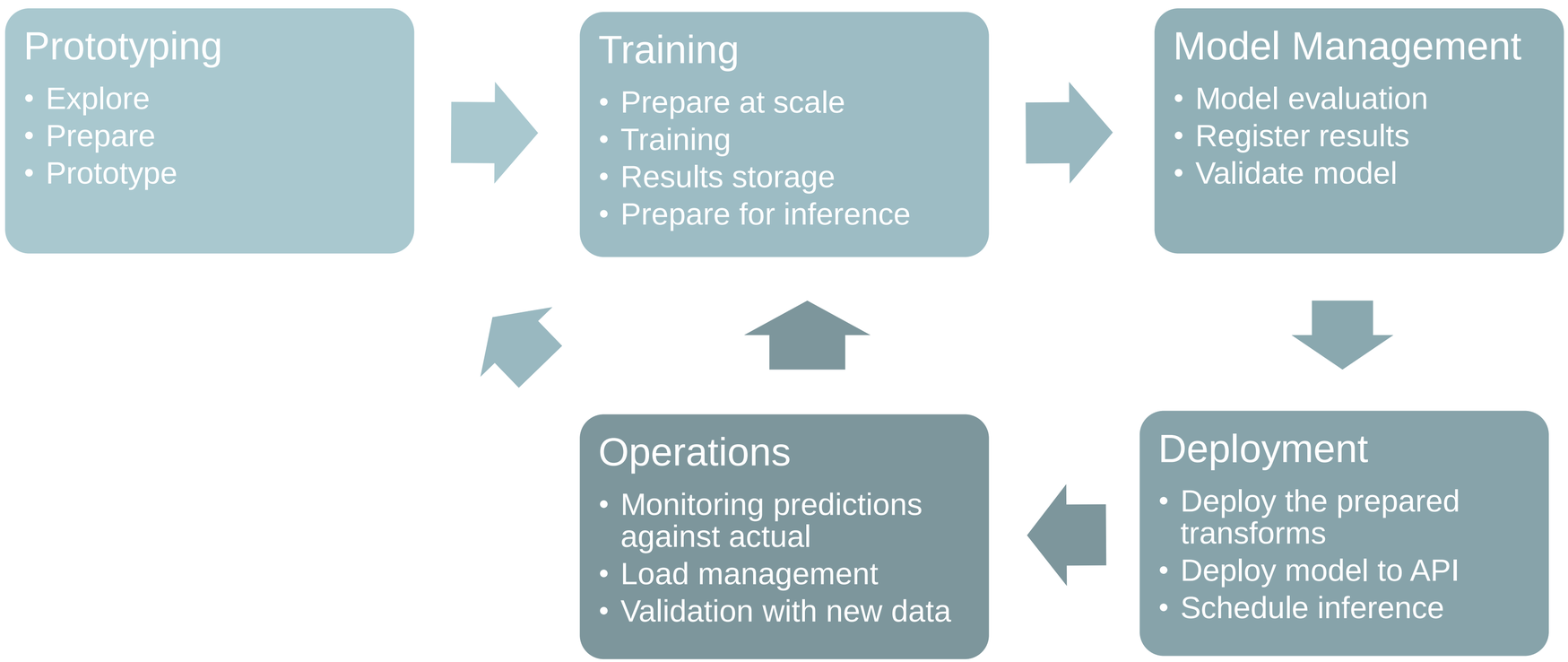
How most people **should** envision an ML project



Activities in each Phase

Actual Implementation

How most people **should** envision an ML project



Prototyping

Steps included

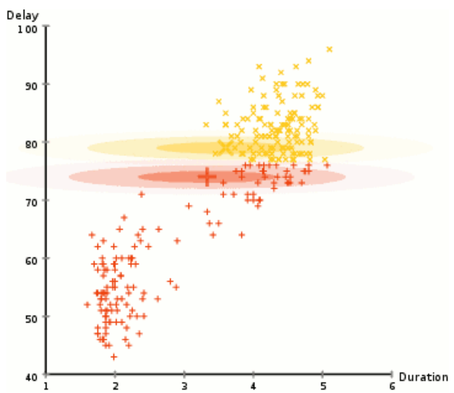
Prototyping
- Explore
- Prepare
- Prototype

Explore

- Understanding the problem
- Identifying data source and useful datapoints

Prepare

- Initial features
  - Identification of useful datapoints for training a model
  - Removal of outliers and faulty data that could skew the model
- Derived features
  - Transform available data to a feature to make ready for training



Prototype

- Running through different algorithms (highly iterative -> many combinations)
- Identifying the hyperparameters range
- Identify a strong candidate to move to training phase

Prototyping

Common problems

- Creating derived features creates a lot of artifacts to track
  - It can get complicated!
- Generating features need to be repeatable and reusable
  - Will be reused in the rest of the phases,
    - E.g. Prepare at scale, prepare for inference, validation and deployment
  - Other developers need to review the model
  - Other developers could reuse the same artifacts
  - E.g. If you run a PCA in training, and find weights for linear combination of initial features to construct the derived features, you need to apply the same weights to create the derived features in the test and validation datasets
- Keep in mind, the initial training process code could potentially be very different from the code that's used to deploy and utilize the model in production.

Training

Steps included

Training
- Prepare at scale
- Training
- Results storage
- Prepare for inference

Prepare at scale

- Productionalize the transformations using the artifacts created in the prototyping stage
  - Done by a data engineering team and included in the “normal” ELT runs
- Ensure the deployed code (training, maintenance, and inference pipelines) perform at high volumes of data

Training

- Using a larger data set and the hyperparameter range, do a deep dive to determine the best model
- Track and record experiments

Result storage

- Build pipeline to store the results in the data lake

Preparation for inference

- In the preparation for scale there may have been decisions like “exclude outliers” but in inference stage all data will go through the model.

Training

Common Problems

Prepare at scale and for inference

- This is often passed off to another team with competing priorities (data engineering)
- If it is not, the data scientist may not have the expertise to productionize properly
  - Automated deployment and testing are not always considered
- Seems to be the major blocker for many AI projects

Model Management

Steps included

Model Management
- Model evaluation
- Register results
- Validate model

Model evaluation

- Picking which one is the best via the error measurement analysis

Register the results

- Want to know which model was chosen and track it
- Used to compare against in the future as the model will need updating
- Creates a container image
  - Tracking packages and their versions are critical

Validate model

- Using a new set of data, confirm the performance of the model
- Need to have ability to revalidate with same data, or in the future adjust to new data (configurable)
- Can need an approval workflow

Model Management

Common problems - Reliable evaluation & comparison of models

- Before deploying a new model to production, performance is evaluated
  - Compare new model performance with already existing models
- Avoiding Biases in testing & validation:
  - When datasets are divided into train and test set, the division can not be biased
  - Multiple validation datasets are generated with different strategies to assess the performance of the model in an as closely as possible unbiased environment
  - It is like an automated unit test
- When data has natural grouping
  - When features are separated into groups, those groups may have different patterns and behaviours. E.g. geographical regions, seasons, COVID vs. Non-COVID years
  - Different validation strategies are required to make sure the model is doing well enough for all groups
- To identify & confirm model drift
  - The performance of a productionalized model needs to be periodically evaluated to make sure it is performing as well as expected. To identify and investigate possible drifts, different validation datasets are generated and need to be tracked appropriately

Deployment

Steps included

Deployment

- Deploy the prepared transforms
- Deploy model to API
- Schedule inference

Deploy the prepared transforms

- The same training transformations on raw data needs to be applied in maintenance and inference pipeline before the data is fed to the model

Deploy the container to an API

- Using Azure Container Instance or Azure Kubernetes, a web service is created for the model that can be reached with an API

Schedule inference

- The inference pipeline will be triggered on a schedule determined by business users
- Azure ML Pipelines, Azure Data Factory Pipelines, Databricks Jobs, or Azure DevOps Pipelines can facilitate this task, depending on choice of execution orchestration tool

Deployment

Common problems

Package management & environment replication

- Types and versions of the packages that data scientists use in their initial investigations may vary and can have significant effect of the output of the models
- Team members often need to replicate each other’s work for knowledge transfer or code review

Operations

Steps included

Operations
- Monitoring predictions against actual
- Load management
- Validation with new data

Monitor predictions

- Periodically, the performance of the live models will need to be assessed

Load management

- Given the data load, compute resources need to be adjusted to minimize the cost incurred by the client while achieving the performance standards
- Initial investigations, training pipelines, maintenance and inference compute resources are completely different

Validation with new data

- If a new flavor of data is received (such as data from a new region), the model is re-validation to ensure it is performing as well as expected

Operations

Common Problems

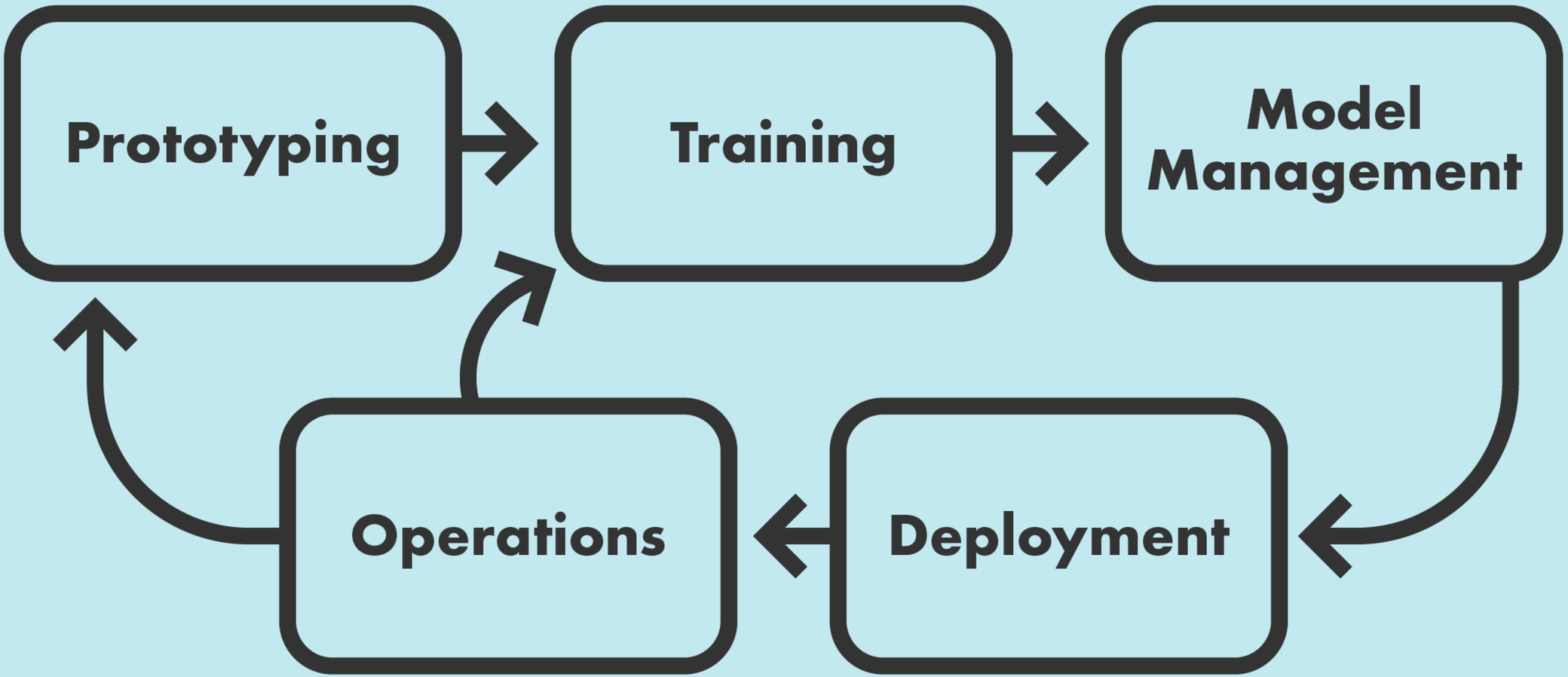
Model drift is caught late

- Normally when downstream business users identify the model's drift, it is too late
- Although retuning hyperparameters usually helps, model drift can have various causes, so investigations may be required to fix the problem. Investigations are not possible without proper historical records and enough information.

Mismatch between computation resources & data volume

- Leads to overhead costs and delayed reports
- Managing infrastructure for different stages of ML life cycles must be done with minimal effort from data science team

MLOps Lifecycle



Why Invest?

Early Identification of Model Drift

- Pro-active monitoring and re-training ensures models adapt to new data patterns

Optimize Data Scientists

- Repeated and automated processes allow Data Scientists to focus on the science

Realize ML Benefits Sooner

- Standardized processes move models through the lifecycle more efficiently

Cost Reduction

- Highly performant code and more efficient allocation of resources all drive lower development and support costs

Thank you

Please visit SDK for more Information:

Appendix A - Assumptions

Using Databricks or Azure ML as the development environment

Although Databricks or Azure ML is not required, our current iteration assumes it is. This removes the necessity to manage the infrastructure or containers if an IDE or VM development environment is used in the preparation and training phase.

Models may already be trained

The solution will work with a variety of models, or if partial MLOps is in place. The solutions we are proposing are technology agnostic in that they are Microsoft and Databricks based.

Batch inference is target

Current iteration is focused on batch. This is important as the Data Preparation API management and deployment has not yet been included.

Appendix B - Common Terms

- Algorithm
  - Set of code outlining steps that learns from the data to generate an ML artifact
- Initial feature
  - Data from exploratory analysis of raw data that were found to be useful in model building E.g. Temperature
- Derived feature
  - Engineered or transforming of initial features that can be understood better by the algorithm
- Hyperparameter
  - A parameter in a model that influences the way the machine learns and how the model performs. It is essentially a dynamic parameter in algorithm's code
- Inference
  - When the model is in production running predictions

Appendix B - Common Terms

- Overfit
  - When a model memorizes data instead of learning from it and identifying patterns in it. It can happen when the model has too many features, or tuned specifically for the training set where negatively impacts performance when model is applied to a new dataset.
- Drift
  - A model's performance deteriorates over time after being deployed to production. The most common reason is that the more recently received data have different patterns and behaviors than the data that was used to train the model. The most common solution is to retune hyperparameters
- Bias
  - When a model is associating a higher influence to a feature than the feature actually has.