University of Idaho

Department of Computer Science

CS 4622/5622 Applied Data Science with Python

Dr. Alex Vakanski

Lecture 25

Introduction to Data Science Operations

Lecture Overview

Introduction to Data Science Operations

25.1 Introduction to Data Science Operations (DSOps)

- ***Data Science Operations (DSOps)*** is a set of practices for deployment of Data Science projects in production environment
- Similar to the way Data Science is closely related and overlaps with Machine Learning, DSOps are related to ***Machine Learning Operations (MLOps)***
  - MLOps is a set of practices for deployment of Machine Learning models in production environment
- DSOps practices are developed in collaboration between data scientists and operations engineers (software engineers, deployment engineers)
- DSOps provide means for:
  - Simplifying and automating the deployment
  - Assuring the quality of deployed projects
  - Meeting regulatory requirements
  - Reducing costs by creating efficient workflows
  - Aligning deployed projects with the mission and objectives of the organization

Data Science Operations

25.1 Introduction to Data Science Operations (DSOps)

- The set of DSOps practices were initially developed for managing project deployment and the operations phase, however, in recent years, DSOps are increasingly applied for managing the entire life cycle of DS projects
- A depiction of the main phases in a DS project life cycle is shown below

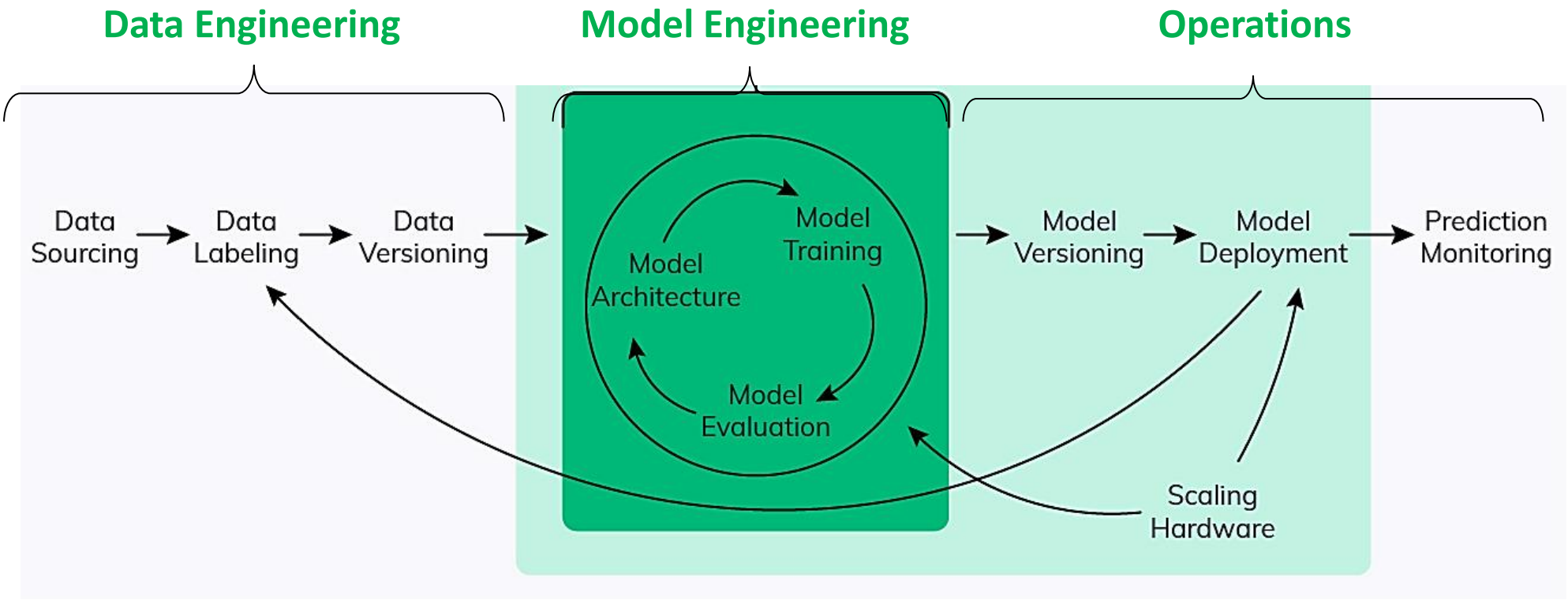

Figure from: https://neptune.ai/blog/mlops

DS Project Life Cycle

25.2 DS Project Life Cycle

- ***Data engineering phase*** (data preparation phase) in the life cycle of DS project can involve several steps, such as:
  - Data gathering / data sourcing
  - Data cleansing / labeling (if used for classification)
  - Data ingestion (preparing batches of data samples, or streams for real-time ingestion)
  - Data analysis and transformation (converting categorical variables, scaling numerical variables)
  - Data validation (missing values, duplicate features)
  - Feature engineering (selecting important features, dimensionality reduction)
  - Data splitting (train, test, validation datasets)

Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

DS Project Life Cycle

25.2 DS Project Life Cycle

- *Model engineering phase* (model creating phase) can involve:
  - Building a model
  - Training optimization (model search, hyperparameter tuning)
  - Model validation (evaluate on unseen test dataset)
  - Training at scale (train with new data, large datasets)
  - Select a final model to be deployed in production

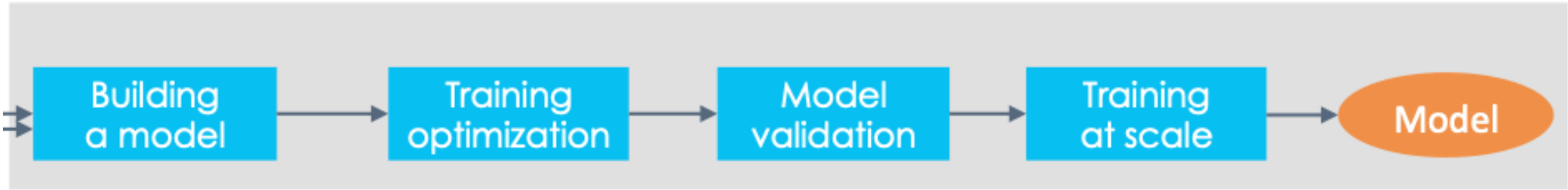

Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

DS Project Life Cycle

25.2 DS Project Life Cycle

- *Model deployment phase* (operations phase) can involve:
  - Deploying the model for access by end-users (e.g., to the cloud, edge)
  - Serving the model (consuming the model by end-users)
  - Monitoring the deployed model and logging the performance
    - Checking for performance degradation
    - Providing explainability of the decision-making process by the model
  - Finetuning the model and improving the performance
    - Deployed models need to be updated periodically
  - Model versioning and data versioning
    - Keep track of deployed models and data used with different models

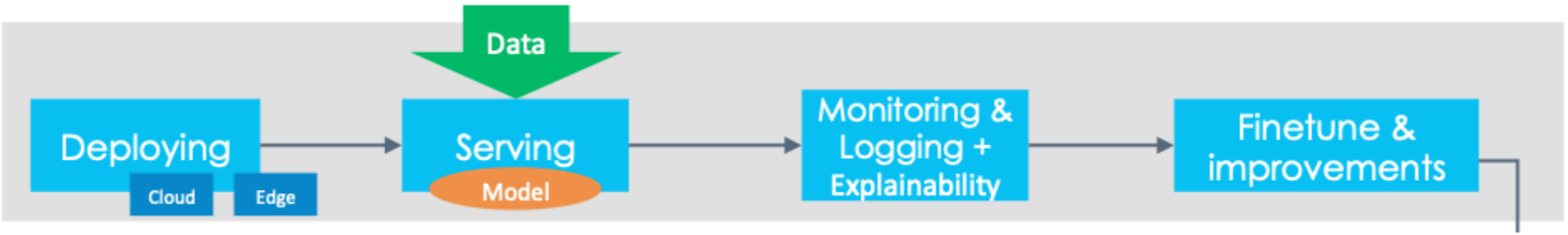

Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

Performance Degradation

25.2 DS Project Life Cycle

- Performance degradation in deployed ML models can be caused by:
  - Changes in the input data or labels between the trained and deployed model
  - Discrepancy between the data processing techniques in the training and serving pipelines
  - Using wrong objectives to optimize, that can result in collecting biased data for training the model
    - For instance, new biased data can be collected that is used for re-training the future versions of the model, resulting in even more biased model
- Changes in the data between the trained and deployed model can be caused by:
  - **Data drift**, the data distribution is changed
    - E.g., a bug in the data pipeline, launch the model in a new country, new users from different demographics, malicious users feeding the model bad data
  - **Concept drift**, the relationship between the data and the output is changed
    - E.g., users' behavior changes
  - **Domain shift**, the data distribution is not sampled adequately
    - E.g., bias in the data sampling process, outliers (rare examples that are not present in the training data occur)

DS Project Life Cycle

25.2 DS Project Life Cycle

- DSOps practices are frequently used to streamline the entire life cycle of DS projects, from data preparation and model creation, to model deployment and monitoring
  - Application of DSOps allows to run and deploy DS project consistently from end to end

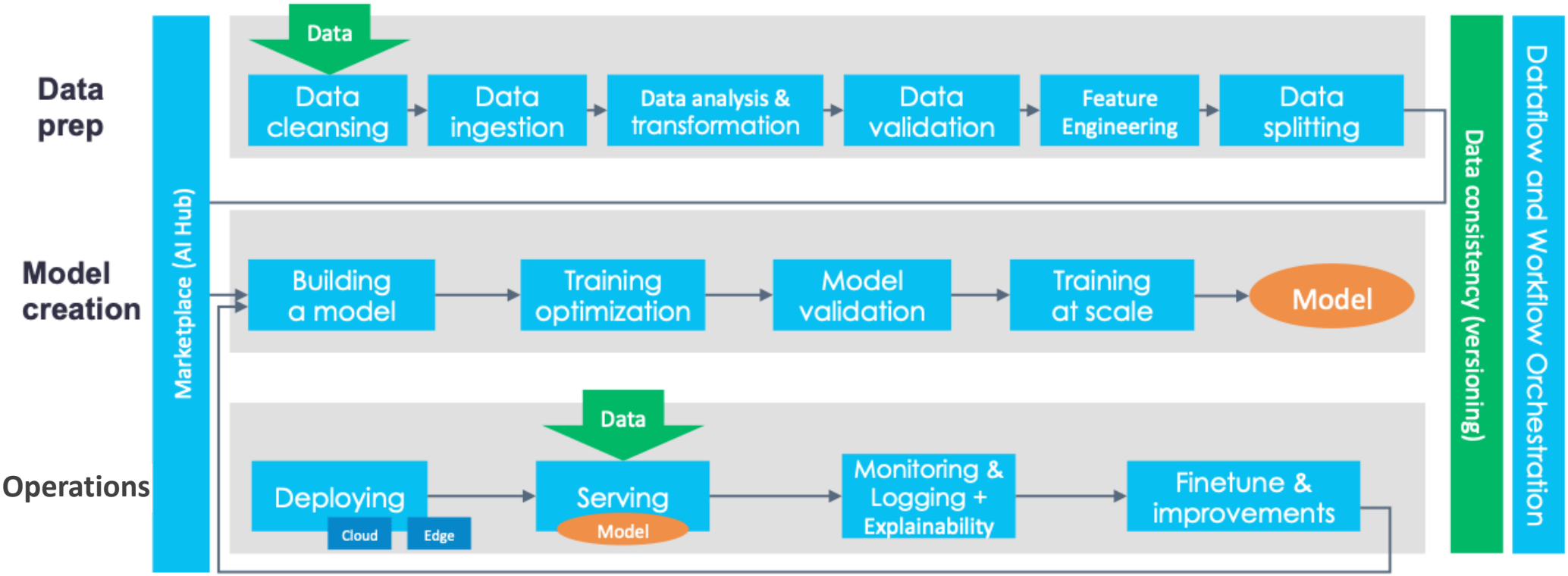

Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

DSOps vs DevOps

25.2 DS Project Life Cycle

- DSOps (and MLOps) were derived based on the *Development Operations (DevOps)* principles for software deployment and operations
  - DevOps were introduced to shorten software development life cycle via consistent practices for software testing, versioning, continuous delivery, monitoring
- The differences between DSOps and DevOps include:
  - DSOps is more experimental, since it involves training and validation of multiple models
  - DSOps typically requires hybrid teams that consist of data scientists, ML researchers, software engineers, deployment engineers, etc.
  - DS projects are characterized with performance degradation over time, due to evolving data
    - Requires monitoring models in production (e.g., adopt metrics to be monitored), and re-training and updating the deployed models
  - Continuous Integration (CI) involves testing and validating data and models, not only code
  - Continuous Delivery (CD) requires to establish model training pipeline and model deployment pipeline

DSOps Levels of Automation

25.3 DSOps Levels of Automation

- DSOps can be generally implemented at three levels of automation
  - DSOps Level 0 - manual process
  - DSOps Level 1 - ML pipeline automation
  - DSOps Level 2 - CI/CD pipeline automation

DSOps Level 0 – Manual Process

25.3 DSOps Levels of Automation

- ***DSOps Level 0*** – this level is typical for companies that are just starting with DS and ML
- Characteristics:
  - Each step of the life cycle is completed manually
  - Disconnect between model engineering and operations: data scientists create the model, and hand it over to operations engineers who deploy and serve the model
  - Infrequent model iterations: the data science team manages a few models that don't change frequently (e.g., update models a few times per year)
  - Absence of Continuous Integration (CI) or Continuous Delivery (CD): testing the models is part of the script or notebook used for training
  - Deployment is typically as a microservice with RESTful API
  - Lack of active performance monitoring

DSOps Level 1 – ML Pipeline Automation

25.3 DSOps Levels of Automation

- ***DSOps Level 1*** – the goal is to automate the ML model pipeline to achieve continuous delivery of prediction service
- Characteristics:
  - Rapid experimentation: experiments are orchestrated and done automatically
  - Continuous training (CT): the model is automatically re-trained in production, using fresh data
    - Requires automated validation of new data, and validation of updated models
    - Requires pipeline triggers to retrain models with new data (e.g., based on a schedule, availability of new data, model performance degradation, data distribution shift)
  - Modularized code for pipelines: code is designed to be reusable and shareable across the ML pipelines (e.g., using containers)
  - The model deployment step that serves the trained and validated model as a prediction service is automated
  - Pipeline deployment: differently from Level 0 where a trained model is deployed as a prediction service, Level 1 deploys an entire ML model pipeline that automatically runs to serve the trained model as the prediction service

DSOps Level 2 – CI/CD Pipeline Automation

25.3 DSOps Levels of Automation

- ***DSOps Level 2*** – the goal is to automate the CI/CD pipeline
- Level 2 is more suitable for tech-driven companies that need to:
  - Manage a large number of models
  - Retrain the models frequently (e.g., daily, or hourly)
  - Redeploy the models on many servers simultaneously
- Characteristics:
  - Development and experimentation: iteratively try new ML algorithms, and push the code to a source repository
  - CI pipeline: build source code and run various tests, output are pipeline components to be deployed in a later stage
  - CD pipeline: deploy the components from CI to the target environment, output is a deployed pipeline with an updated model
  - Automated triggering: the pipeline is automatically executed in production based on a schedule or in response to a trigger
  - Monitoring: the model performance is monitored based on live data

DSOps Level 2 – CI/CD Pipeline Automation

25.3 DSOps Levels of Automation

- DSOps Level 2 requires the following components
  - Source control: for versioning the code, data, and models
  - Test & build services: using CI tools for quality assurance, building packages and executables for pipelines
  - Deployment services: using CD tools for deploying pipelines to the target environment
  - Model registry: a registry for storing trained ML models
  - ML metadata store: tracking metadata of model training, such as model name, parameters, training data, test data, and metric results
  - Feature store: a repository to standardize the definition, storage, and access of features for training and serving
  - ML pipeline orchestrator: automating the steps of ML experiments

Model Deployment

25.4 Model Deployment

- ***Model deployment*** comprises activities for making the model accessible for making predictions (inference)
  - The model is run in a production environment, where it needs to provide reliable and fast prediction service to end-users
  - A report from 2021 shows that 41% of organizations with over 25,000 employees have more than 100 models in production
  - Some companies update their ML model every 10 minutes

Production Environment Considerations

25.4.1 Production Environment Considerations

- Most people learn DS and ML concepts from courses, academic papers, or from research projects (e.g., in graduate school)
- Deploying ML models in production has important differences in comparison to ML practice in courses or research projects
  - The tables lists some of the main differences

	Courses/Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have	Important
Interpretability	Good to have	Important

Table from: Chip Huyen, Designing Machine Learning Systems

Production Environment Considerations

25.4.1 Production Environment Considerations

- **Objectives**
  - Research: design an ML model that achieves state-of-the-art performance
  - Production: design an ML that satisfies the requirements of many stakeholders
    - E.g., managers, sales team, product managers, infrastructure engineers
- **Computational priority**
  - Latency versus throughput
    - Latency – time from receiving a query to returning a result
    - Throughput – number of processed query within a time period
  - Production environment prioritizes low latency
    - E.g., 53% phone users will leave a page that takes more than 3 seconds to load
- **Data**
  - Research: clean, static, mostly historical data
  - Production: messy, constantly shifting, historical + streaming data
- **Fairness**
  - The model is not biased against demographics groups
    - E.g., job applications are not rejected by a model because of the ZIP code
- **Interpretability**
  - The users can interpret the decision-making process by the model
  - Interpretability is very important for the end-users to thrust the model

Production Environment Considerations

25.4.1 Production Environment Considerations

- Pace of software deployment and ML model deployment is accelerating
  - Performers who have established DevOps deploy 973 times more frequently, and have 6570 times faster lead time to deployment

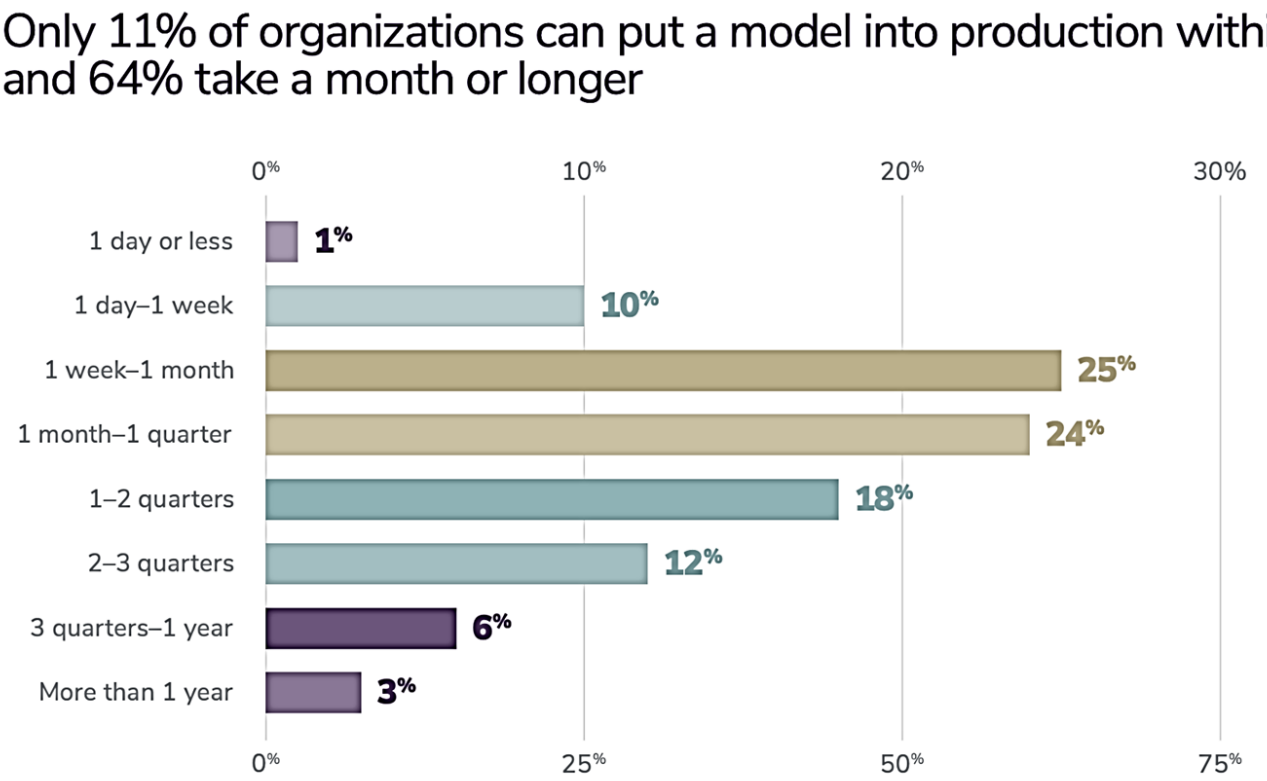

Figure from: Chip Huyen, Designing Machine Learning Systems

Online Prediction vs. Batch Prediction

25.4.2 Online Prediction vs. Batch Prediction

- During deployment, it is important to decide whether the predictions will be produced to the end-users as online prediction or batch prediction
- *Online prediction*
  - Generate predictions as soon as requests from end-users arrive
  - Predictions are returned as responses immediately to the end-users
  - It is also known as *synchronous prediction* (or *on-demand prediction*) since the predictions are generated synchronously with requests
- *Batch prediction*
  - Generate predictions periodically before requests arrive, store them (e.g., in SQL tables, or CSV files), and retrieve them when needed
    - E.g., movie recommendations are generated every 4 hours, and are shown to users when they log onto the website
  - It is also known as *asynchronous prediction*, since the predictions are generated asynchronously with requests

Batch Prediction

25.4.2 Online Prediction vs. Batch Prediction

- The figure depicts *Batch Prediction*, where a Data Warehouse is used to store batch predictions by Prediction Service
  - Using Batch Features (step 1), the Prediction Service periodically supplies predictions to the Data Warehouse (step 2)
  - When the App requests a prediction (step 3), Precomputed Predictions are fetched from the Data Warehouse (step 4)
- Batch features are computed from historical data, and stored in data warehouses or databases
  - Data warehouses are repositories that integrate data from one or more sources, and store current and historical data in one single place

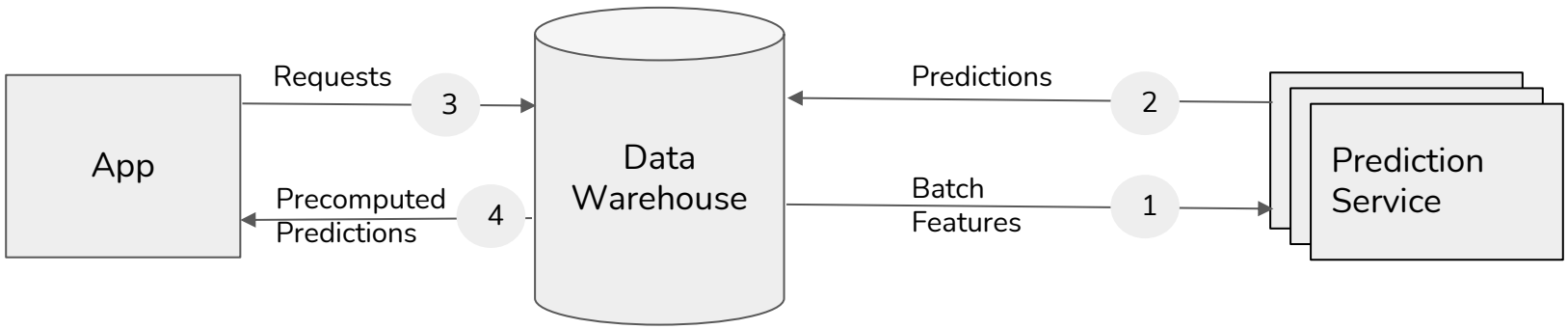

Figure from: Chip Huyen, Designing Machine Learning Systems

Online Prediction

25.4.2 Online Prediction vs. Batch Prediction

- In *Online Prediction*, when the App sends a request (step 1), the Prediction Service uses data features (step 2) to generate Predictions for the App (step 3)
- Online prediction can use both batch features (from historical data in a Data Warehouse) or streaming features (from Real-time Transport data)
  - E.g., to estimate delivery time for an order on Doordash, ML model can use batch features (e.g., average preparation time of this restaurant in the past) and streaming features (number of orders in real-time, number of delivery persons, etc.)

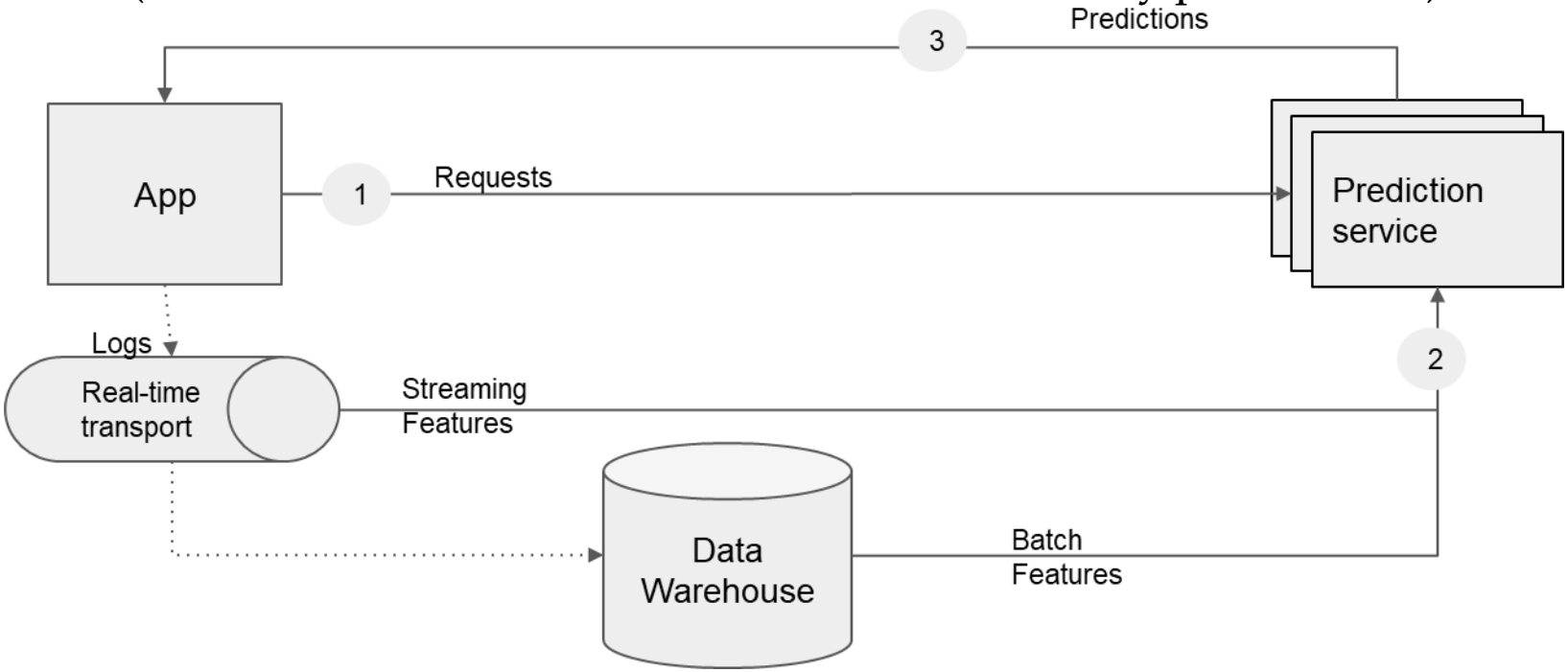

Figure from: Chip Huyen, Designing Machine Learning Systems

Online Prediction vs. Batch Prediction

25.4.2 Online Prediction vs. Batch Prediction

- Comparison between Online Prediction and Batch Prediction

	Online prediction (synchronous)	**Batch prediction (asynchronous)**
Frequency	As soon as requests come	Periodical (e.g., every 4 hours)
Useful for	When predictions are needed as soon as data sample is generated (e.g. fraud detection)	Processing accumulated data when you don't need immediate results (e.g. recommendation systems)
Optimized	Low latency	High throughput
Examples	• Google Assistant speech recognition • Twitter feed	• TripAdvisor hotel ranking • Netflix recommendations

Table from: Chip Huyen, Designing Machine Learning Systems

Deploying to the Cloud vs. Edge

25.4.3 Cloud Computing vs Edge Computing

- *Cloud computing* is performed on web servers
  - Examples:
    - Queries to Alexa, Siri, Google Assistant
    - Queries to Google Translate
  - Cloud providers offer many tools that facilitate model deployment
  - It is a preferred way for model deployment by many companies
- *Edge computing* is performed on edge devices, such as cell phones, tablets, laptops, embedded devices, smart watches, cars, etc.
  - Examples:
    - Unlocking devices with fingerprints, face recognition
  - Running models on edge devices is an appealing option, and many companies have been recently trying to push their models to edge devices

Benefits of Cloud Computing

25.4.3 Cloud Computing vs Edge Computing

- Benefits of cloud computing:
  - Access to structured environments (libraries, tools) for managing DS projects
  - Access to latest computational resources (no need to purchase hardware or software)
  - Pay for what you need only
  - Ability to quickly scale projects
  - Improved efficiency by relying on infrastructure hosted and managed by the cloud provider, and ML tools developed by the cloud provider
- Cloud computing companies have invested hundreds of billions of dollars in infrastructure and management
  - Spending in 2023 reached $599 billion, compared to $491B in 2022 and $178B in 2021
  - Amazon Web Services (AWS), Microsoft, and Google Cloud accounted for 65% of the spending in 2023
- AWS Sagemaker, Microsoft Azure ML, Google Cloud AI provide many DSOps tools to facilitate the life cycle of DS projects

Benefits of Edge Computing

25.4.3 Cloud Computing vs Edge Computing

- Can work without **Internet connection**, or with unreliable connections (rural areas)
  - Caveat: in some cases, apps may need external real-time information to work
- Do not need to worry about **network latency**
  - I.e., latency in sending data to the model on the Cloud, and afterward sending prediction back to the users
  - Large network latency can make some tasks impossible (e.g., text autocomplete)
- Fewer **privacy concerns**
  - There is no need to send user data over networks (which can be intercepted)
    - Cloud database breaches can affect many people
  - Easier to comply with regulations (e.g., GDPR by European Union)
- **Cheaper**
  - Training ML models can take long time and can results in high Cloud service costs
  - A mistake in estimating Cloud computing costs can bankrupt your startup

Disadvantages of Edge Computing

25.4.3 Cloud Computing vs Edge Computing

- The main disadvange of edge computing is that the devices may not be powerful enough to run models (especially high-performing DL models)
  - Computational power constraint
    - Large deep learning models require GPU
  - Memory constraint
    - To store the model and load it into memory
  - Energy constraint
    - Battery power to run an app for long periods of time
- Potential solutions include:
  - Hybrid cloud-edge computing, where predictions for frequent inputs are precomputed on the Cloud and stored on the Edge device
  - Improve the capabilities of edge devices, such as manufacture more powerful hardware
  - Implement smaller or optimized models

Continuous Deployment

25.5 Continuous Deployment

- ***Continuous deployment*** in DSOps requires to develop automated CI/CD pipelines and related infrastructure so that deployed models can uninterruptedly learn from new data
- Continuous Integration (CI) provides guidelines for automatically developing, testing, and merging code changes in Data Science projects in a structured manner
  - CI allows Data Science teams to confidently and quickly apply changes to the projects, since the CI pipeline provides for proper integration of made changes, where any errors in the code can be easily detected and resolved
- Continuous Delivery (CD) provides guidelines for delivering the code and models from CI to production environment
  - CD allows to serve the end-users with the latest updated version of the model

Continuous Deployment

25.5 Continuous Deployment

- **Use cases** for continuous deployment
  - It is especially important for model performance in rare events
    - E.g., Black Friday, Prime Day shopping, Christmas
    - Such events can significantly degrade performance of models trained on regular historical data
    - Solution: apply continuous deployment using fresh data throughout the day/the season
  - Continuous cold start
    - It means that the model needs to make predictions for a new user without any historical data
    - Other examples include cases when existing users haven't logged in for a very long period of time, or existing users who rarely log in (a user books a hotel a few times a year)
    - Continuous deployment is used to adapt to the user's preference during the current session
      - For example, Tik Tok can make recommendations for new users within several minutes
- **Disadvantages** of continuous deployment
  - Higher cost – requires infrastructure, developed pipelines, compute resources for training and evaluating multiple models, storage capacity
  - Risk of catastrophic forgetting – refers to the tendency of a neural network to completely forget previously learned information when training with new data

Stateless Retraining vs Stateful Training

25.5.1 Stateless Retraining vs Stateful Training

- Updating the model can be achieved via stateless retraining or stateful training
- ***Stateless retraining***, refers to training the model from scratch with new volume of data
  - I.e., the current state of the model is not preserved when updating the model
- ***Stateful training*** refers to finetuning an existing model on new data
  - This training mode is also called incremental learning
  - It allows to update the model with less data
  - The model converges faster, requires less compute resources than stateless retraining
  - Stateful training is the preferred approach in continuous deployment
    - However, there are cases when stateless retraining is required
      - E.g., when new input features are added to an existing model
      - E.g., when a new model architecture is applied

Stateless Retraining vs Stateful Training

25.5.1 Stateless Retraining vs Stateful Training

- Model updating with stateless retraining creates new versions of the model v1, v2, v3, by training from scratch when new data become available
- Model updating with stateful training uses an initial model v1 and creates fine-tuned versions of the model v1.1., v1.2, v1.3 as new data become available

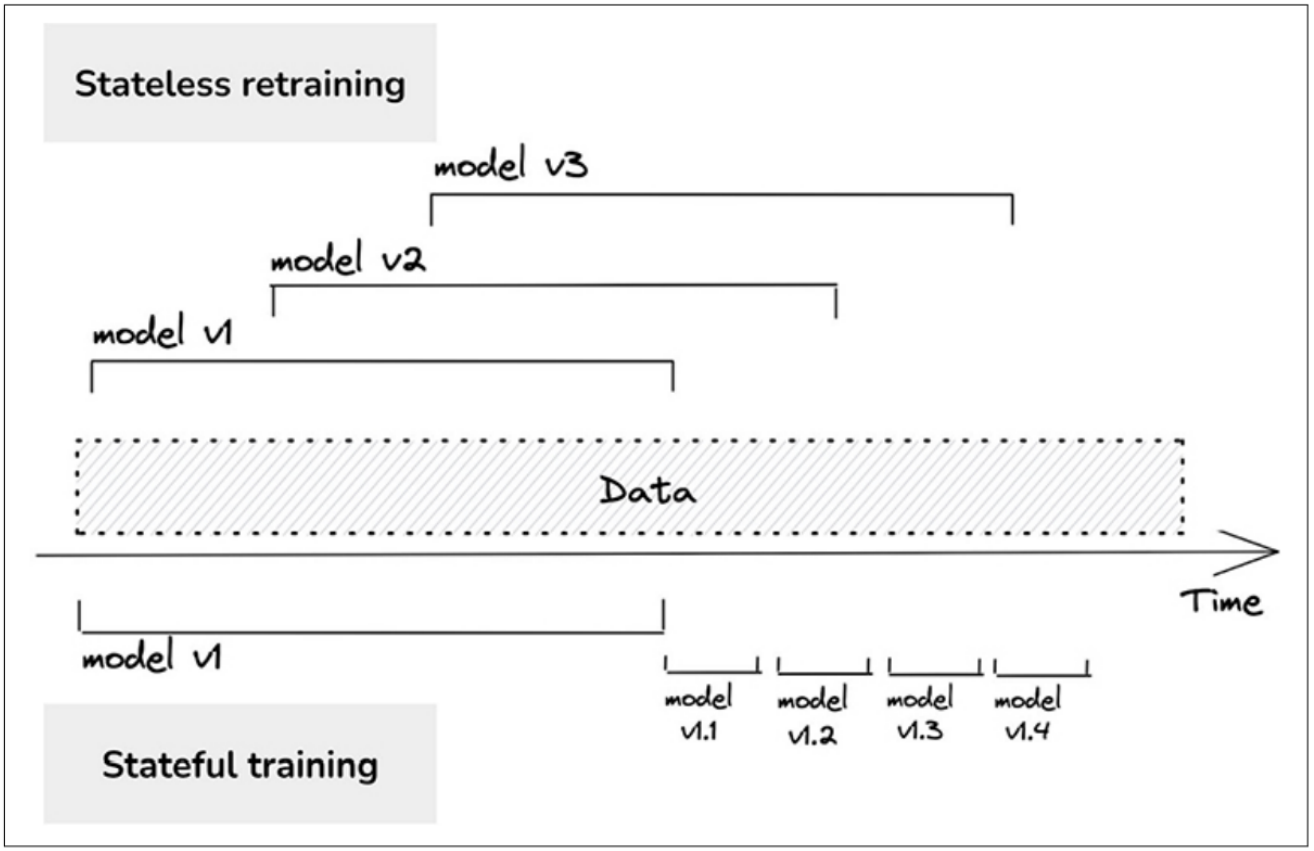

Figure from: Chip Huyen, Designing Machine Learning Systems

Triggering Model Updates

25.5.2 Considerations for Model Updates

- Deciding when to update the model requires to define a trigger mechanism
  - Time-based trigger
    - E.g., every 5 minutes
  - Performance-based trigger
    - E.g., the accuracy drops below 90%
  - Data volume-based trigger
    - E.g., the total amount of available new data is increased by 5%
  - Drift-based trigger
    - E.g., a major data distribution shift is detected

Frequency of Model Updates

25.5.2 Considerations for Model Updates

- When the updates are time-based, an important decision for continuous deployment is the ***frequency of model updates***
- Considerations for this decision are based on:
  - Quantifying the value of data freshness
    - E.g., what is the gain in model's performance when switching from retraining monthly to weekly to daily?
    - To calculate the gain, train models on historical data using different time windows, and evaluate the models on most recent data to see how the performance changes
      - Adopting a smaller window for updating usually can increase business performance (such as higher click rate, lower prices for rides)
  - Deciding on model iteration vs. data iteration
    - Model iteration: change the model architecture, employ new algorithm
    - Data iteration: train the same model using new data
    - It is recommended to periodically perform model iteration and data iteration
      - E.g., introducing a new model architecture can significantly improve the performance

Model Updating

25.5.3 Model Updating

- Model updating can be based on:
  - Offline evaluation – the model is evaluated using historical data
  - Online evaluation – the model is evaluated using fresh production data
    - Pros: the online data used for model evaluation has the same distribution as the incoming production data
    - Cons: online evaluation in production is risky
- Several approaches for safe ***online model evaluation*** are employed
  - Shadow deployment
  - A/B testing
  - Canary release
  - Interleaved experiments

Shadow Deployment

25.5.3 Model Updating

- ***Shadow deployment***
  1. Deploy the new model in parallel with the existing model
  2. Route each incoming request to both models to make predictions, but only serve the predictions by the existing model to the users
  3. When the results by the new model are satisfactory, switch to the new model

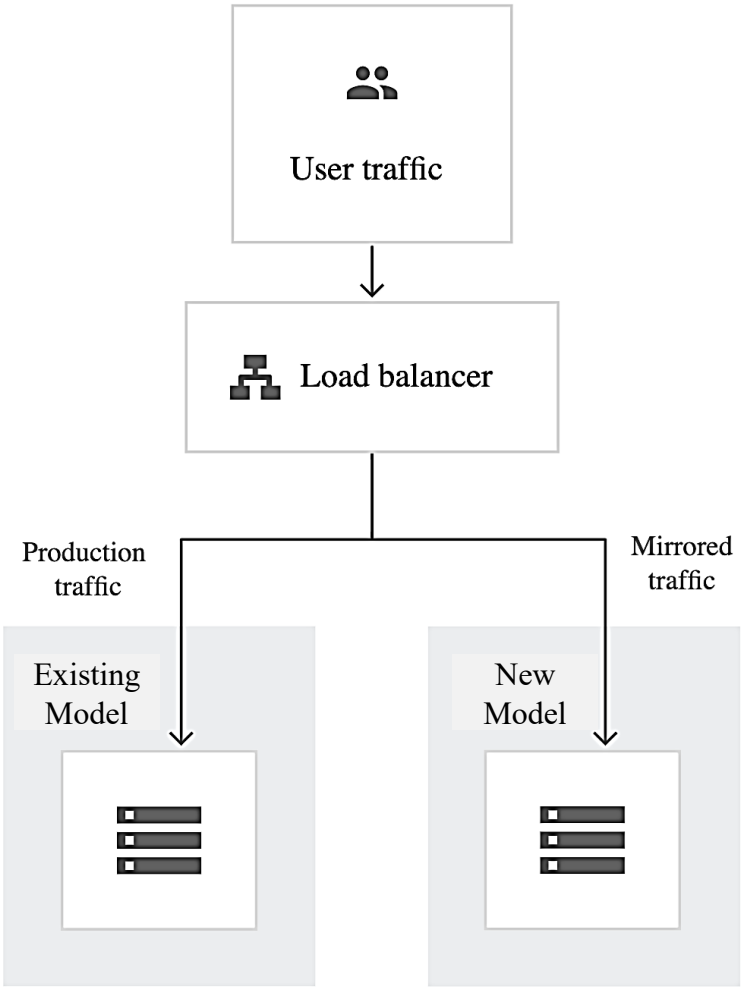

Figure from: Chip Huyen, Designing Machine Learning Systems

A/B Testing

25.5.3 Model Updating

- ***A/B testing***
  1. Deploy the new model alongside the existing model (e.g., model A and model B)
  2. A random portion of the traffic is routed to the new model for prediction and serving, based on routing rules
     - The rest of the traffic is routed to the existing model for prediction and serving
  3. Monitor the predictions by both models and collect feedback from end-users, to determine if there is a statistically significant difference in the performance based on two-sample tests
  4. Select the better-performing model to keep
- Notes:
  - A/B test should be run on sufficient number of samples, to ensure confidence in the performance
  - It is possible to evaluate not only one, but multiple candidate models (e.g., A/B/C testing)

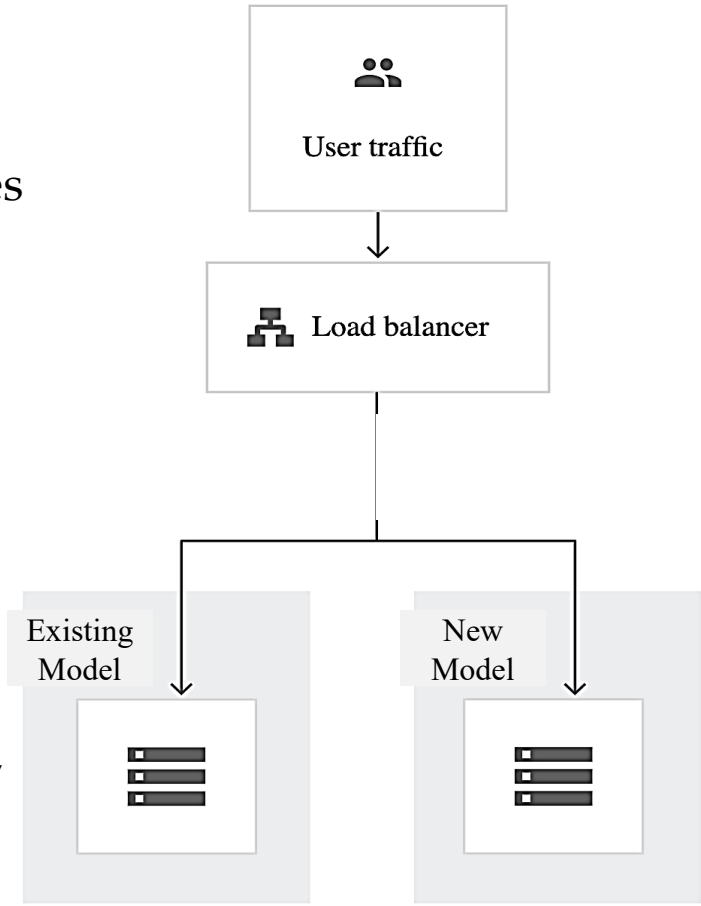

Figure from: Chip Huyen, Designing Machine Learning Systems

Canary Release

25.5.3 Model Updating

- ***Canary release***
  1. Deploy the new model (called canary) alongside the existing model
  2. Portion of the traffic is routed to the new model
  3. If the performance is satisfactory, gradually increase the traffic to new model until it fully replaces the existing model
     - E.g., roll out to USA first, then Europe, Asia, and the rest of the world
     - If the performance is poor, keep the existing model

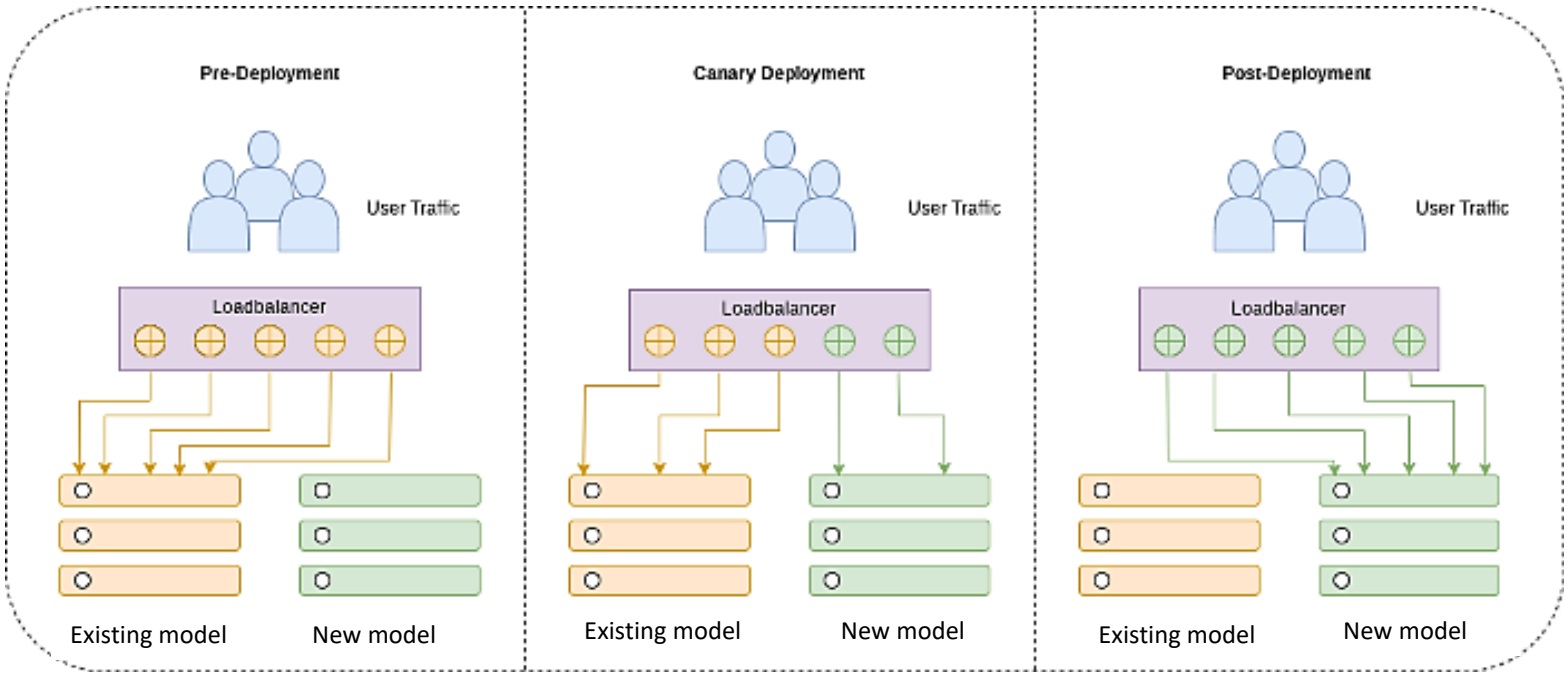

Figure from: https://blog.getambassador.io/kubernetes-canary-testing-and-release-with-istio-4cbdedcc9914

Interleaved Experiments

25.5.3 Model Updating

- ***Interleaved experiments***
  1. Deploy the new model alongside the existing model (e.g., model A and model B)
  2. Mix (interleave) the predictions by both models A and B together, and show them to the end-users
  3. Analyze end-users' preferences (e.g., click rates) to decide which model to keep
- This approach is especially useful for recommender systems and search ranking
- Comparison to A/B testing
  - In A/B testing, one group of users is served the predictions by model A, and the rest of the users are served the predictions by model B
  - In interleaved experiments, all users are served predictions by both models A and B

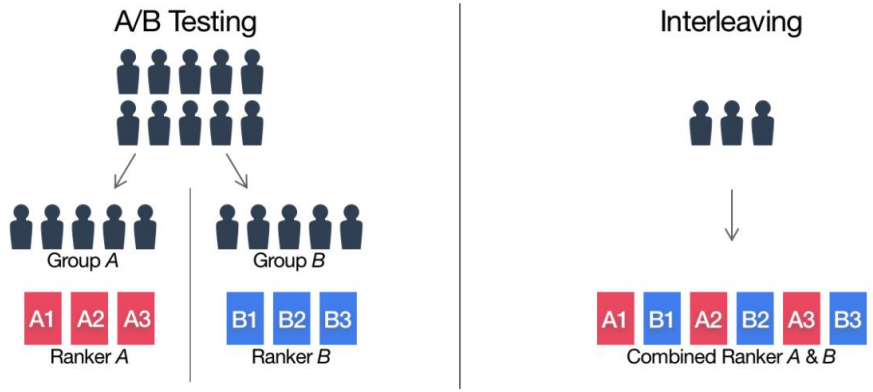

Figure from: Chip Huyen, Designing Machine Learning Systems

References

1. Chip Huyen, Designing Machine Learning Systems, O'Reilly Media, 2022.
2. MLOps: What It Is, Why It Matters, and How to Implement It, by Prince Canuma, available at: https://neptune.ai/blog/mlops