PROCESSING BIG DATA FOR ANALYTICS -MODEL DEVELOPMENT & MONITORING

Liu Fan
isslf@nus.edu.sg
Jan 2023

Total Slides: 54

Objectives

At the end of this module, you should be able to:

- Understand Machine Learning Lifecycle
- Understand Different types of Machine Learning Deployment
- Understand Model Monitor & Data Drift

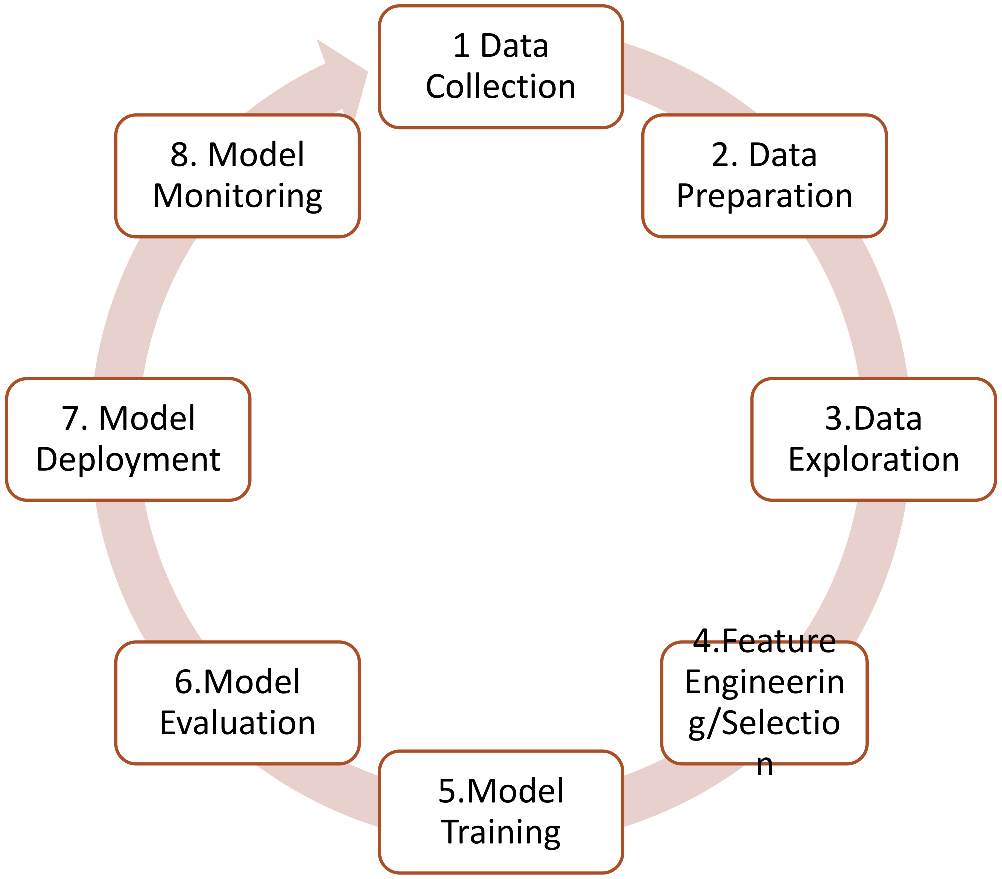
Agenda

Topics

- Model Development
  - Machine Learning Lifecycle
  - Machine Learning Pipeline
  - Different Types of Machine Learning Deployment
  - Big Data-driven Model Deployment
- Model Monitoring
  - The Concept of Model Monitoring
  - Motivations of Model Monitoring
  - Monitor Systems
  - Data Drift

Machine learning lifecycle

Machine Learning Lifecycle(recap)



A Typical Machine Learning Lifecycle(recap)

1. Data Collection
    - Data collection is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems. In this step, we need to identify the different data sources, as data can be collected from various sources such as files, database, internet, or mobile devices. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output.

2. Data preparation(Data cleaning/Data wrangling)
    - This may be the most tedious and time-consuming task in this cycle, which involves identifying various data quality issues. Data might comprehend misplaced entries, irregularities and semantic errors. Data scientists need to reformat and clean the data.

3. Data exploration
    - It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.

4. Feature Engineering/Selection
    - Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling. The goal of feature engineering and selection is to improve the performance of machine learning (ML) algorithms.

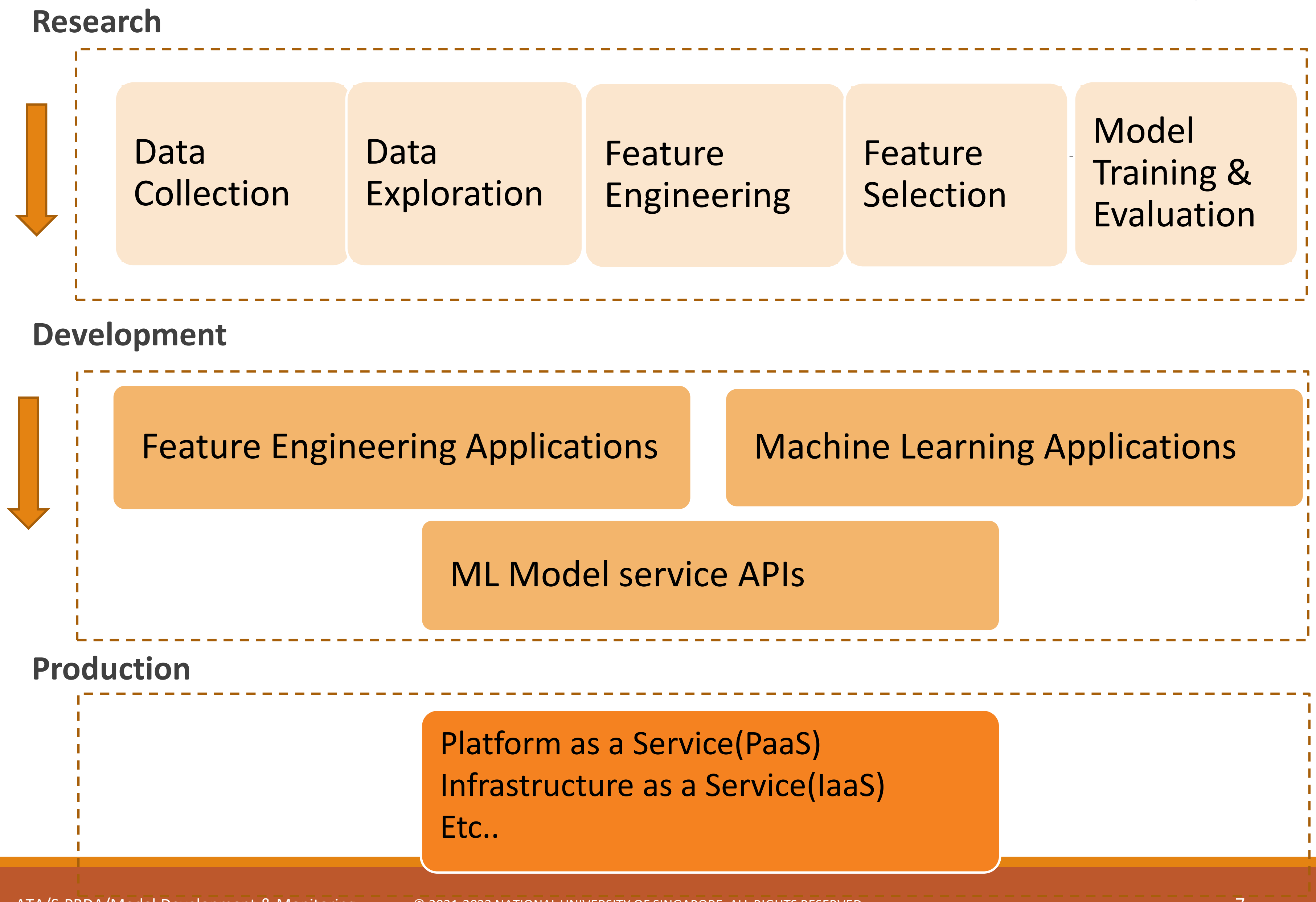
5. Model Training
    - We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

6. Model evaluation
    - After training the model on data, we evaluate the end result to see how well it performed or how reliable it is in real-life situations. Each performance metric has a distinct evaluation metrics.

7. Model deployment
    - The term deployment can be defined as an application of a model that makes the prediction by means of the new data.

8. Model monitoring
    - Model monitoring is the close tracking of the performance of ML models in production so that production and Data science teams can identify potential issues before they impact the business

Different Machine Learning Environments


Research
Data Collection
Data Exploration
Feature Engineering
Feature Selection
Model Training & Evaluation
Development
Feature Engineering Applications
Machine Learning Applications
ML Model service APIs
Production
Platform as a Service(PaaS)
Infrastructure as a Service(IaaS)
Etc..

Machine learning pipeline

Machine Learning Pipeline (Recap)

What is ML pipeline?

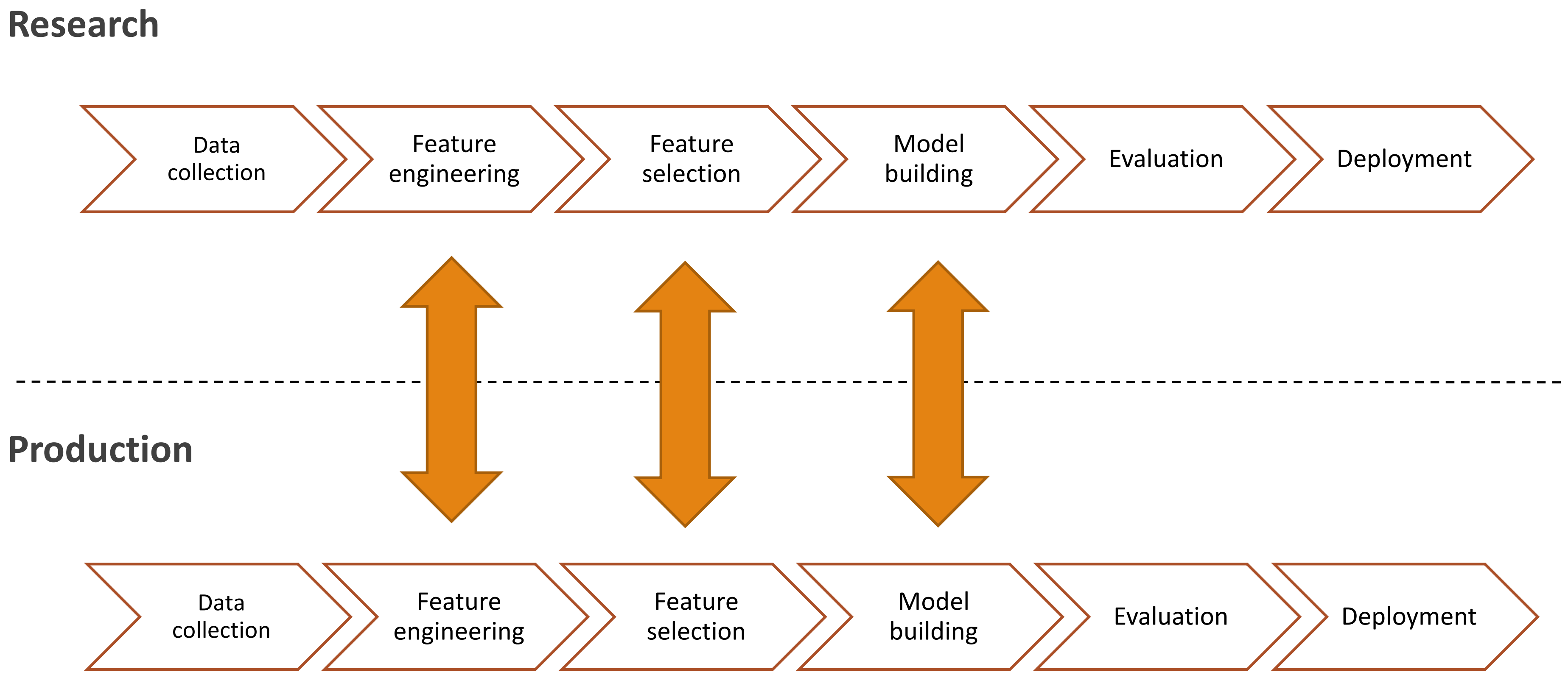
- ML pipeline is a means of **automating** the **machine learning workflow** by enabling data to be transformed and correlated into a **model** that can then be analyzed to **achieve outputs**.

Example

- A team might have spent several months building a model to detect fraudulent transactions. However, after that model is developed, it needs to actually be deployed. In this case, that means integrating it into existing processes so that it can actually start scoring transactions and returning the results.
- The machine learning model deployment actually refer to deployment of machine learning pipeline.
- The machine learning pipeline contains a variety of steps. we need to deploy the entire pipeline.

https://algorithmia.com/blog/ml-pipeline

Deployment of Machine Learning Pipeline



MODEL DEPLOYMENT

What is Model Deployment

- Deployment is the method by which you integrate a **machine learning model** into an **existing production environment** to make practical business decisions based on data. It is one of the most challenging stage.
- In order to start using a model for practical decision-making, it needs to be effectively deployed into **production**.
- It requires coordination between different parties such as **data scientists**, **IT teams**, **software developers**, and **business professionals** to ensure the model works reliably in the organization's production environment.

https://www.datarobot.com/wiki/machine-learning-model-deployment/

Challenges of Model Deployment (1)

Reliability

- The ability of the software to produce the intended result and the robustness to the errors

Reusability

- The ability of the software to be used across systems and projects

Maintainability

- The ease with which we can maintain the software

Flexibility

- The ability of adding or removing functionality from the software.

Reproducibility

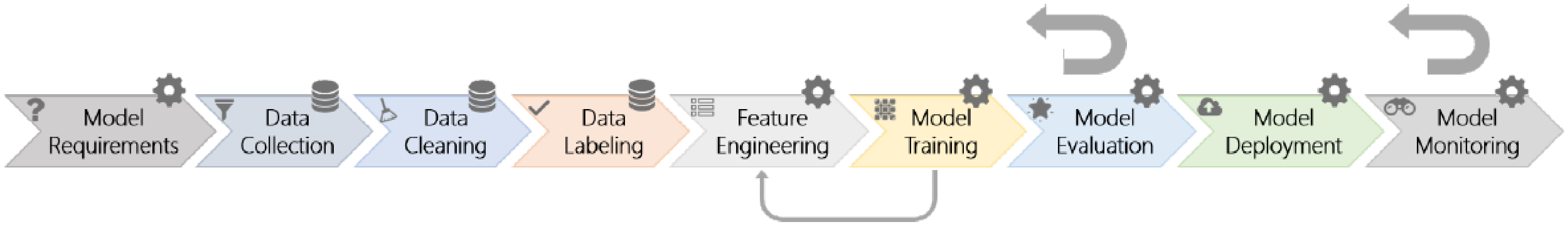
- The ability of the software to product the same result, given the same data across systems

Challenges of Model Deployment (2)

- ML systems are complex
  - The ML model is a small fraction of real-world ML systems. The required surrounding infrastructure is vast and complex. The complexity of many machine learning systems adds to the challenges of deployments.
- Data Dependencies
  - Data may from many different sources.
  - Internal data can be removed or updated and external data can change suddenly
- Configuration issues
  - Model hyperparameters, model versions, requirements
- Data preparation and Feature engineering
  - Data preparation and feature engineering code can be very complex and comprised of many steps.
- Testing
  - Traditional tests often do not detect errors in ML systems. Exception won't be thrown by a model that is simply underperforming or returning incorrect prediction. Alternative ways of capturing and detecting those mistakes need to be designed.
- Collaboration in different teams
  - Such as business team software engineering team, data science team, IT team and operation team.

How to address the challenges?

- Automation of all stages of the ML workflow



- Eliminating manual steps to get a model to production environment.
- Such as adding data processing and feature engineering steps to production environment and utilize integration and deployment tooling to automatically version and deploy our models.

How to address the challenges?

- Reproducibility
  - Training is reproducible
  - There should be a way to undo deployment so the previous model version is restored.
- Testing
  - Test the entire ML pipeline
    - Train on subset of the data
    - Use simpler model for training
  - Feature engineering and pre-processing code is tested
  - Model configuration is tested

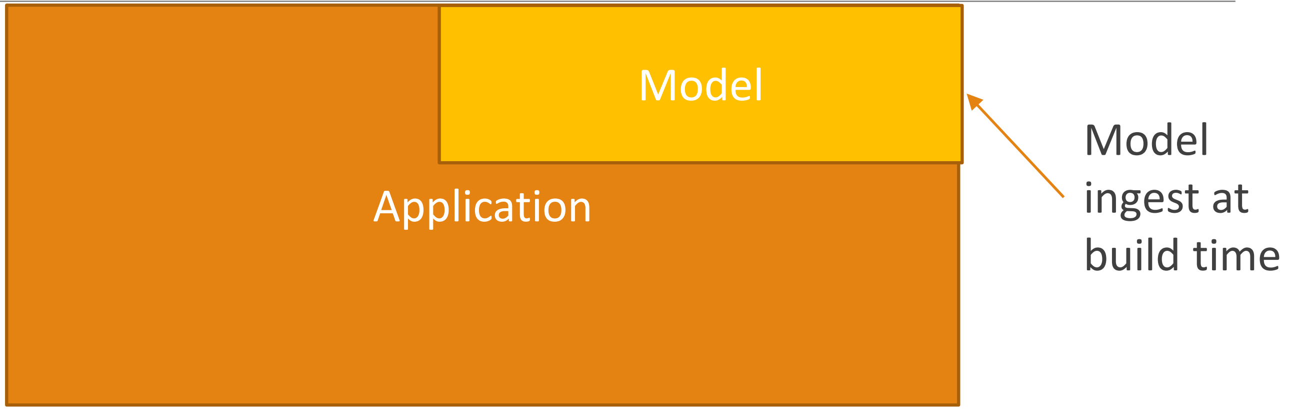
Deployment reproducible Machine Learning Pipeline

- Data collection
  - Training data cannot be reproduced –> save a snapshot
  - Databases are constantly updated and overwritten –> design databases with accurate timestamp
- Feature Engineering & Selection
  - Replace missing data with random values –> set a seed
  - Removing labels based on percentages –> code on how a feature is generated should be tracked under version control.
- Model training
  - Machine learning models relay on some random selected number –> set a seed, record the order of the features, record the hyper-parameters
- Model deployment
  - Software versions should be the same
  - Use a container and track software specifications
  - Develop, deploy utilise the same language

Different types of Machine Learning Deployment Systems

- Model Embedded in Application
- Served via a dedicated service
- Model published as data
- Batch prediction (offline process)

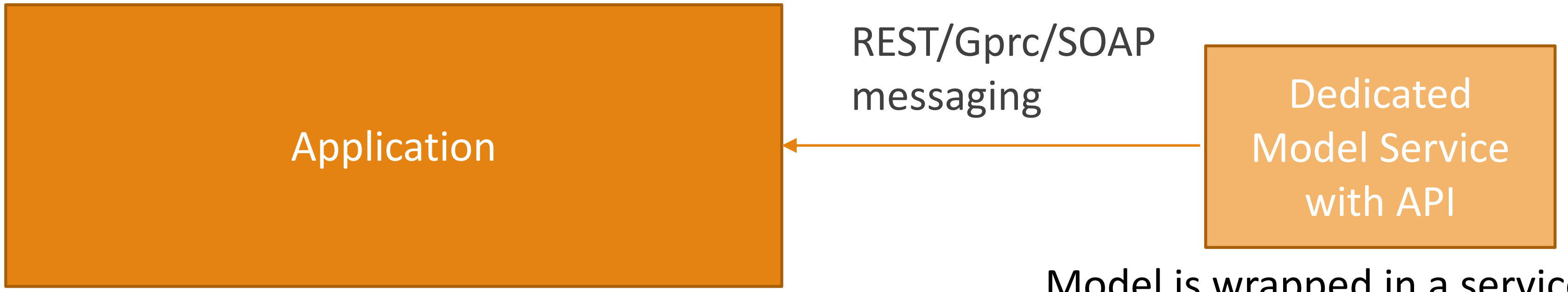
1. Embedded Model



- Pre-Trained: Yes
- On-the-fly prediction: Yes
- Variations: Embedded on mobile device (Core ML), running in the browser (Tensorflow.js)
- In this model, the trained model is embedded in the application as a dependency.
- An example of this instance is if we had a flask application that we used to predict the value of a property. Our Flask application would serve an HTML page to collect information about a property. The Flask application would take those details as inputs, forward them to the model to make a prediction then return them to the client.
- Due to privacy concert, the data is not sent to the backend, the prediction is done directly on the web browser.

https://towardsdatascience.com/4-machine-learning-system-architectures-e65e33481970

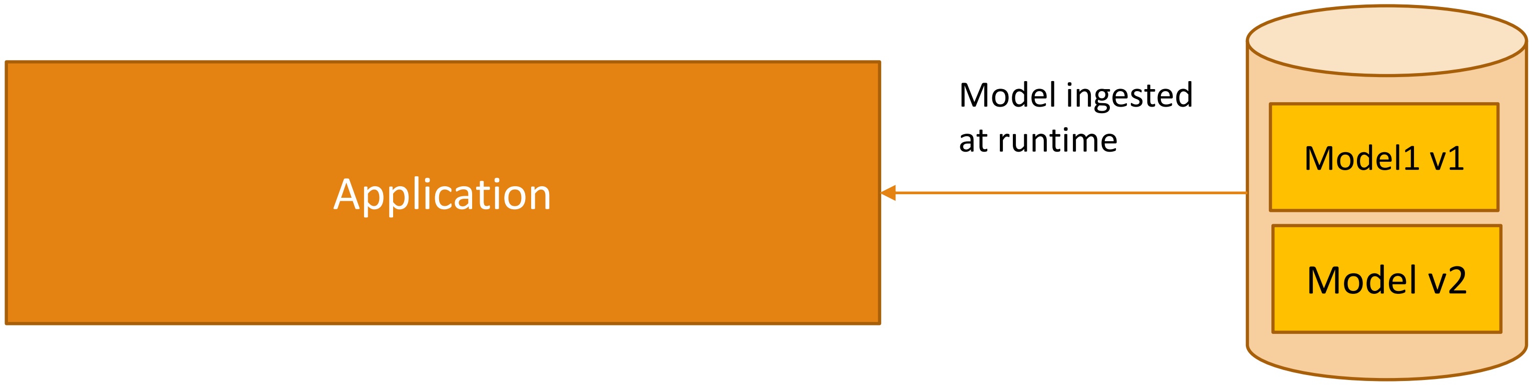
2. Dedicated Model API



Model is wrapped in a service that can be deployed independently

- Pre-Trained: Yes
- On-the-fly prediction: Yes
- In this architecture, the trained machine learning model becomes a dependency of a separate Machine Learning API service. Extending from the Flask application to predict the value of a property example above, when the form is submitted to the Flask application server, that server makes another call — possibly using REST, gRPC, SOAP, or Messaging (i.e. RabbitMQ) — to a separate microservice that has been dedicated to machine learning and is exclusively responsible for returning the prediction.

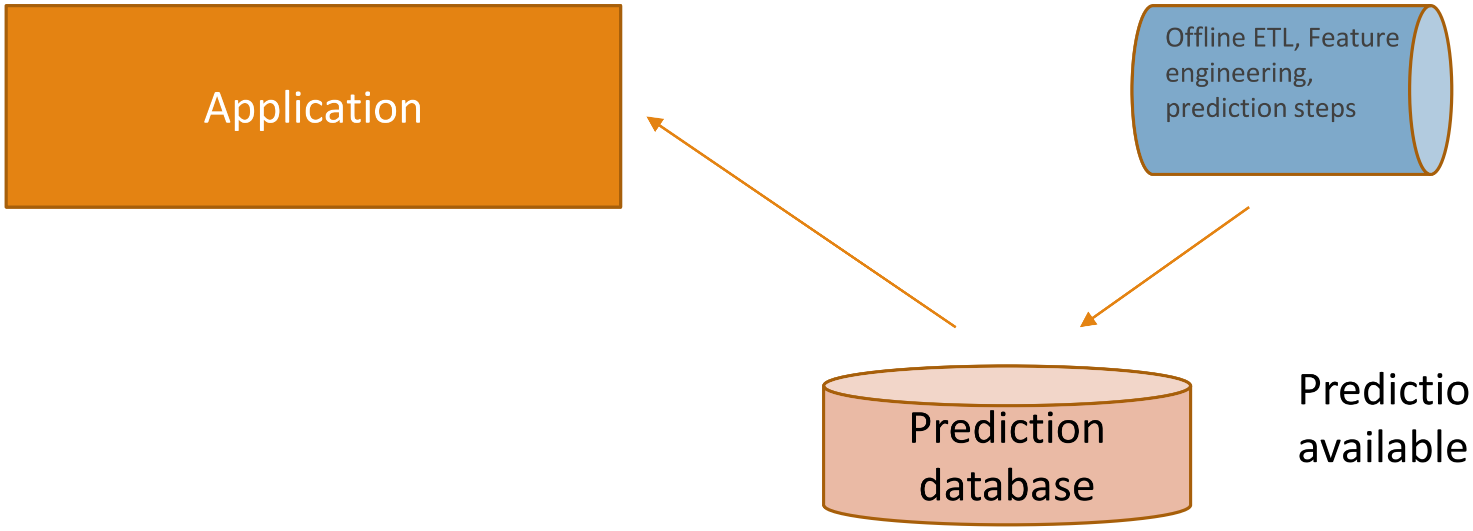
3. Model Published as Data



Streaming platform(e.g. Kafka)
Application subscribes to events and ingests new models in memory

- Pre-Trained: Yes
- On-the-fly prediction: Yes
- In this architecture, our training process publishes a trained model to a streaming platform (i.e. Apache Kafka) which will be consumed at runtime by the application, instead of build time — eligible to subscribe for any model updates.

4. Offline Predictions



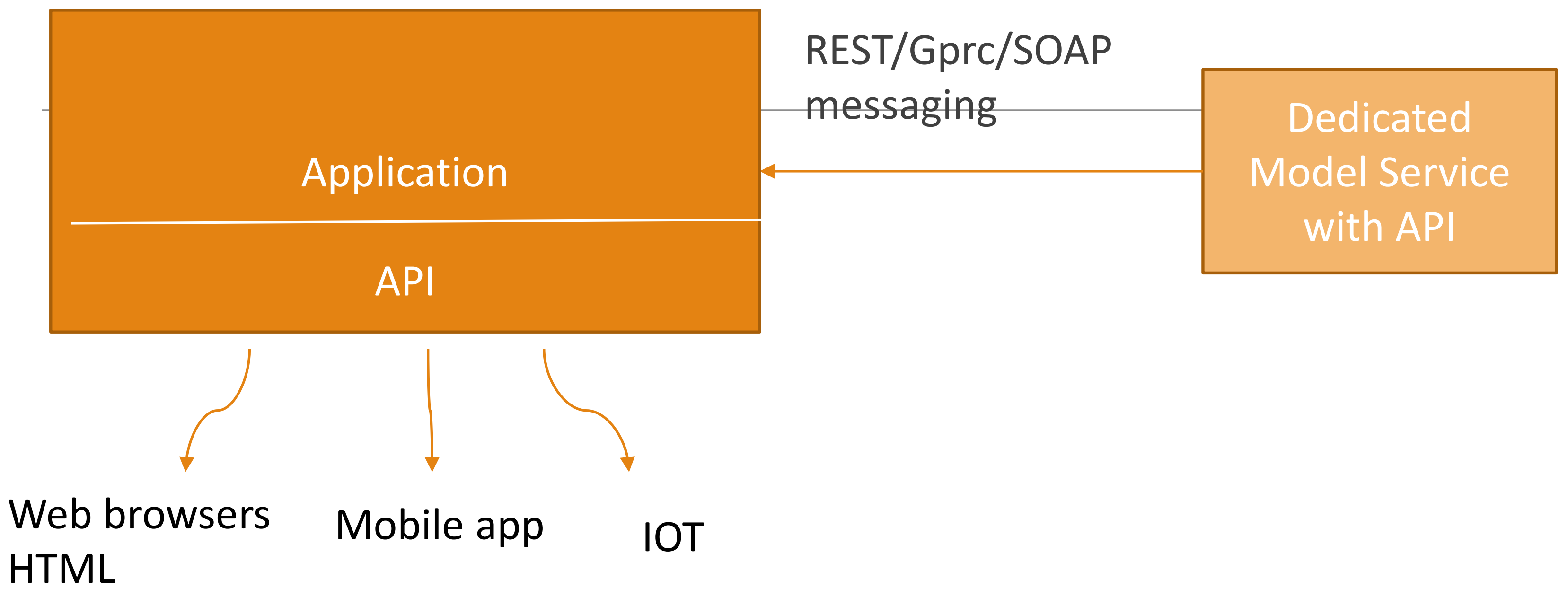
Prediction is not available on the fly

- Pre-Trained: Yes
- On-the-fly prediction: No
- Variations: dashboards, CSV files.
- Predictions are triggered and run asynchronously either by the application or as a scheduled job.

Comparisons

	Embedded	API	Streaming	Offline
Prediction	On the fly	On the fly	On the fly	Batch offline
Result delivery	Within app process	Via API	Streaming via message Queue	Shared DB, API, file
Latency on prediction	Low	Moderate	Depends	Hours/Days
System management difficulty	So so	Easy	Very hard	So so
Model update requires re-deployment	Yes	Yes	No	Yes

Dedicated ML API and Microservices



- In an complicate system, it has multiple microservices.
- it make sense to separate out your application and your ML API.
- In this case, our application might be concern with user registration, authentication, payments and other typical operations activities, if our ML API were embedded in the application and we want to update the user registration code, we need to release both the application and the ML model. In this case, it’s better to choose dedicated ML API.

Python web Frameworks

- Python web frameworks are a collection of modules or packages which help developers in writing a **web application** in the Python programming language.
- Types of Python Frameworks
  - Full-stack Framework
  - Microframework

https://www.monocubed.com/top-python-frameworks/

Full Stack Framework

- Full stack are one of the best Python web application frameworks, known as **one-stop-solution** for fulfilling all kinds of app building requirements.
- This approach has a lot of databases and components that are commonly included in the full-stack framework such as – form validation, form generators, and template layouts.
- Here are some examples of full stack frameworks:
  - Django
  - Web2Py
  - Pylons Framework
  - Pyramid

Micro-framework

- These kinds of web frameworks are known as the **lightweight framework (non full stack framework)** because they do not offer additional patterns and functionalities compared to a full stack framework, such as multi-threaded database abstraction layer, form validation, specific tools, and libraries.
- Some of the best Python web frameworks of this type are:
  - CherryPy
  - Falcon
  - **Flask**
  - Pycnic

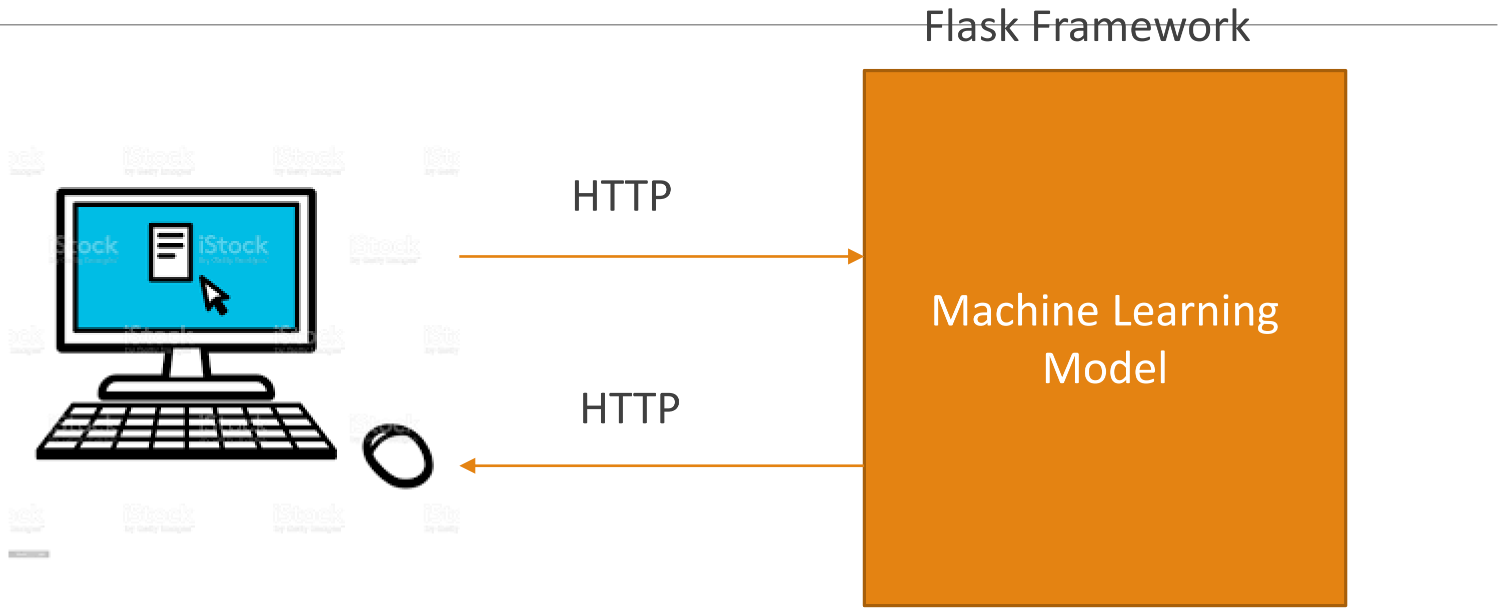
Which framework to choose?

- It needs to look at the size and complexity of the coding strategy.
  - If you are planning to develop a large structure filled with lots of requirements and features, a full-stack framework will be the right choice.
  - On the other hand, if it is planning a small and simple app product, you should think about microframeworks.

Flask

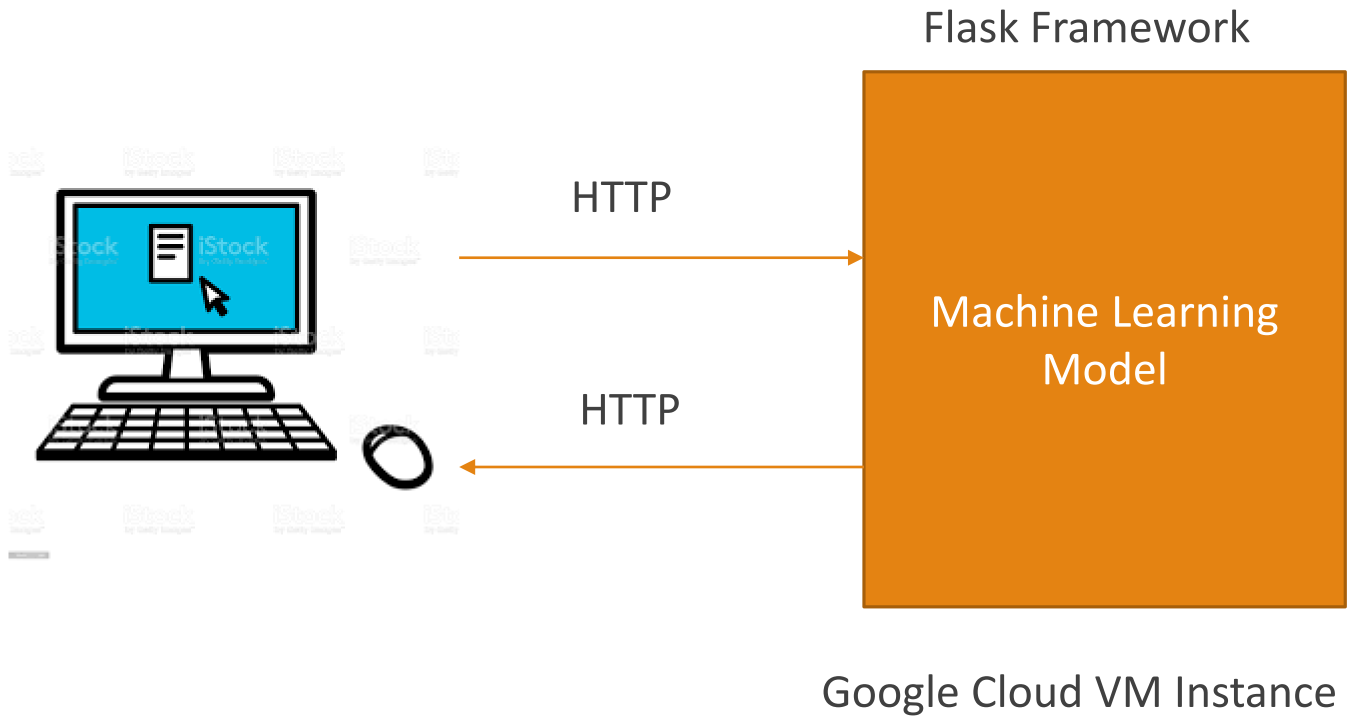
- Flask is a well-known and best Python framework that comes under the Microframework category and comes with the BSD license.
- Flask features:
  - Routing, templating language, session management, etc.
  - Complete support for unit-testing
  - Inbuilt development web server
  - Error logging and ticketing role based access control to administer errors
  - A minimum, pluggable model
  - First-class REST support and coordination
  - Flexible web application section
  - Built-in fast debugger
  - Unit testing support
  - Error tracking mechanism
  - RESTful request dispatching
  - Etc..

Flask REST API



- REST(REpresentative State Transfer) is a popular way of exchanging data, you can build an application using Java, Python. Scala, .net or any other programming language.
- Any client who wants to access your application, they would send a request over HTTP protocol using REST and get a response back, data is typically exchanged in XML or JSON format over HTTP protocol.
- Using flask framework can easily build a REST API for your python application.

Host the ML REST API on the GCP

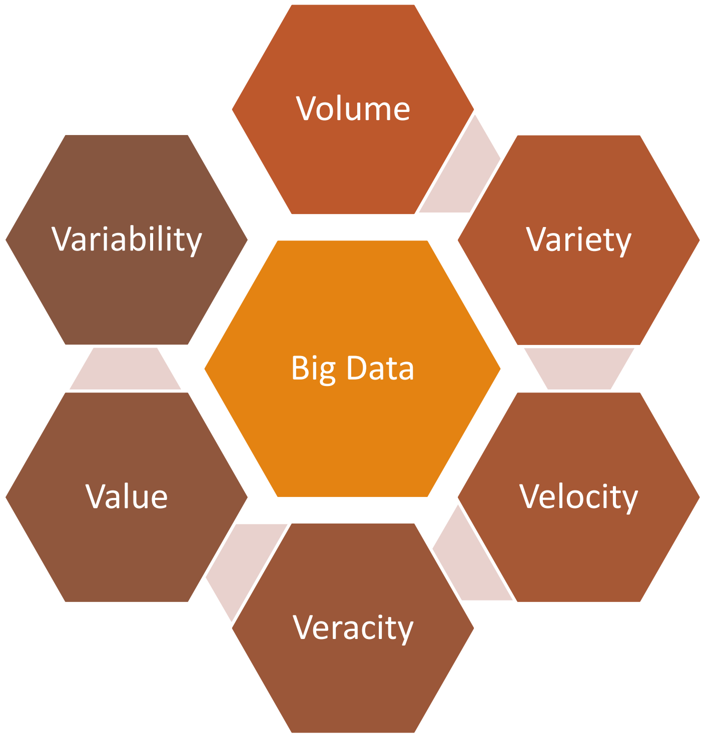


Deploy model on GCP so anybody who has access to the IP address of the VM can access the model.

Big Data-driven Model Deployment

What is Big Data? (Recap)

- Volume – The amount of data
- Variety – The types of data
- Velocity – The speed at which big data is generated
- Veracity – the degree to which big data can be trusted
- Value – the business value of the data collected
- Variability – the ways in which the big data can be used and formatted



Challenges of Big Data Model Deployment

- Model training takes up a lot of compute power
- Normally rely on GPUs
- GPUs can be expensive

Types of Cloud Computing

- **Public Cloud**: Computing infrastructure is hosted at the vendor’s premises.
- **Private Cloud**: Computing architecture is dedicated to the customer and is not shared with other organizations.
- **Hybrid Cloud**: Organizations host some critical, secure applications in private clouds. The not so critical applications are hosted in the public cloud

Classification of Cloud Computing based on Service Provided

- Infrastructure as a service (IaaS)
  - Offering hardware related services using the principles of cloud computing. These could include storage services (database or disk storage) or virtual servers.
  - Amazon EC2, Amazon S3.
- Platform as a Service (PaaS)
  - Offering a development platform on the cloud.
  - Google’s Application Engine, Microsofts Azure
- Software as a service (SaaS)
  - Including a complete software offering on the cloud. Users can access a software application hosted by the cloud vendor on payper-use basis. This is a well-established sector.

Solutions

- **Azure Machine Learning** to build, train, and validate reproducible ML pipelines;
- **Azure Pipelines** to automate ML deployments;
- **Azure Monitor** to track and analyze metrics;
- **Azure Kubernetes** Services and other additional tools.

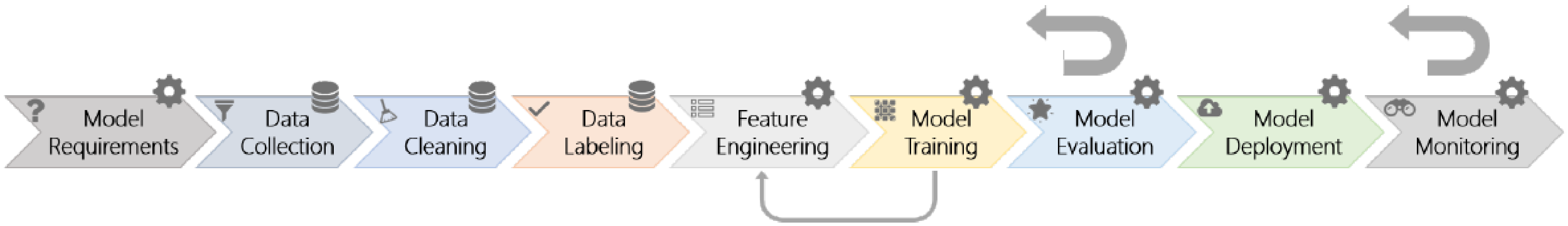
•**Dataflow** to extract, validate, and transform data as well as to evaluate models;
•**AI Platform Notebook** to develop and train models;
•**Cloud Build** to build and test machine learning pipelines;
•**Tensorflow Extended** to deploy ML pipelines;
•**Kubeflow Pipelines** to arrange ML deployments on top of Google Kubernetes Engine.

- **Amazon SageMaker**

A suite of tools to build, train, deploy, and monitor machine learning models.

Model monitoring

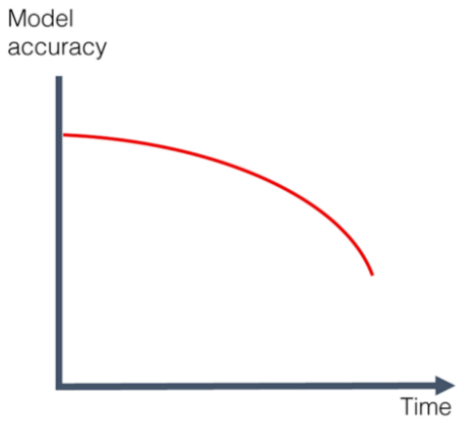
What is Monitoring?



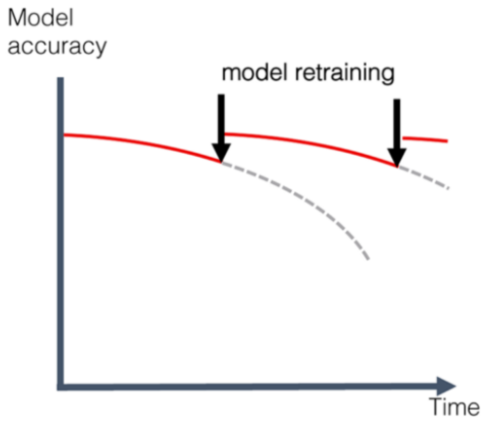
- Monitoring is the last step of Machine Learning lifecycle
- Monitor inputs, outputs and performance
- We should continue monitoring our models throughout their lifetime

Model Decay

- **Model decay**(also known as model drift) are concepts that describe the process during which the **performance** of a model deployed to production **degrades** on new, unseen data or the underlying assumptions about the data change. Model decay is a term that literally says that the model got worse.
- When data quality is fine, there are two usual suspects: **data drift** or **concept drift**. Or both at the same time.



Model decay over time

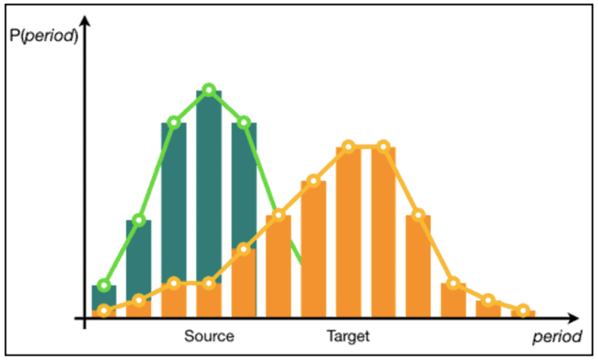


Regularly updated model

https://towardsdatascience.com/machine-learning-in-production-why-you-should-care-about-data-and-concept-drift-d96d0bc907fb

Data Drift

- There is a **shift** that occurs at the data level, as the distribution of the data used to train an ML model, called the source distribution, is different from the distribution of the newly available data, the target distribution.
- Change in the input features:
  - Data may change the distributions of the variables
  - Feature definition change
  - Features become unavailable
  - New features - It happens when some previously infrequent or even unseen feature vectors become more frequent, and vice versa.
- Example: Temperature readings changing from in degree Fahrenheit to degree Celsius.



Changes in the distribution of the feature "period" over time, yields to feature shift

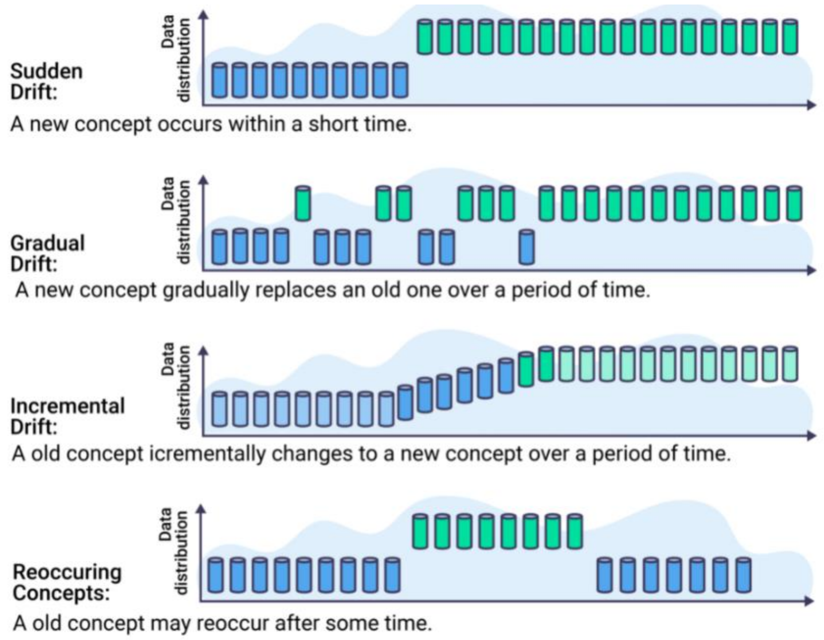
Concept Drift

- Concept Drift - It is the phenomenon where the statistical properties of the class variable — in other words, the target we want to predict — change over time.
- In other domains, this change maybe called “covariate shift,” “dataset shift,” or “nonstationary.”
- Concept drift refers to a change in relationships between input and output.

https://analyticsinsight.b-cdn.net/wp-content/uploads/2020/10/AR2.png

Four Types of Concept Drift

Sudden drift – A new concept occurs in a short period of time.
Gradual drift – A new concept gradually replaces an old one over an extended period of time.
Incremental drift – an old concept incrementally changes to a new concept over some period of time.
Recurring concepts – an old concept may reoccur after some time Note: in gradual drift, the two concepts go back and forth until the new concept finally stabilizes; in an incremental drift, the concept gradually changes to the new one.


Sudden Drift:
Data distribution
A new concept occurs within a short time.
Gradual Drift:
Data distribution
A new concept gradually replaces an old one over a period of time.
Incremental Drift:
Data distribution
A old concept icrementally changes to a new concept over a period of time.
Reoccuring Concepts:
Data distribution
A old concept may reoccur after some time.

- Sudden drift -e.g. at the beginning of pandemic, stock prices suddenly changed
- Gradual drift - you witness fewer new oil companies and more new tech companies
- Incremental drift - a stock price gradually and steadily increases
- Recurring concept - changes in food delivery volumes on the weekends vs weekdays

Approaches to detect drifts

- Monitoring data statistical properties
- The model's predictions
- Python scikit-multiflow library
  - ADWIN: the scikit-multiflow package can detect data drift using an algorithm known as adaptive windowing (ADWIN) that detects data drift over a stream of data. ADWIN works by keeping track of several statistical properties of data within an adaptive window that automatically grows and shrinks. Work well on gradual drift.
  - Drift Detection Method (DDM) – works well on sudden drift detection
  - Early Drift Detection Method (EDDM) – a modified version of DDM

Handling of drifts

- Once the drift is detected, the model needs to be updated. The details will depend on the application but in general:
  - If we diagnose a concept drift, the affected *old data* needs to re-labeled as well and model re-trained.
  - If we diagnose a data drift, enough of the *new data* needs to be labelled to introduce new classes and the model re-trained
  - A combination of the above when we find that both data and concept have drifted

Model and input checks

- Model input monitoring
  - Range of variables. Percentage of missing values, etc.
- Distribution monitoring
  - Statistical test and different statistical parameter to exam the real distribution
- Model performance monitoring
  - Extend model performance test through the model lifetime.
- We compare the inputs and the performance between training data with live data.

Monitoring Tests

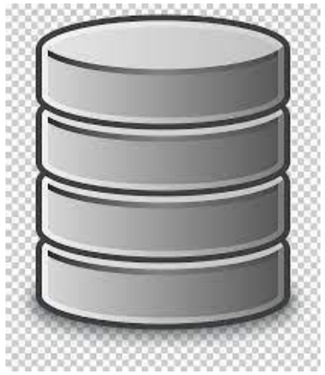
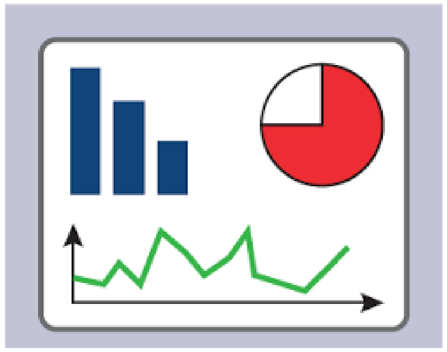
- A change in the data also has an impact on the model, so it is necessary to monitor these changes.
- Monitor that there are no differences between the training data and the data on which the model is being executed.
- Check that we do not have models that have not been trained for a long time, as there could be problems in retraining them (data or training flows not available, the model may have been losing effectiveness, etc.).
- Monitor model performance, not only in effectiveness but also in training times, execution, bandwidth, etc.

https://anllogui.medium.com/machine-learning-model-testing-and-monitoring-strategy-8eff78d08794

Pillars of Monitoring

- Metrics
  - A numeric representation of data measured over interval of time
  - Metrics can harness the power of mathematical modelling and prediction to derive knowledge of the behaviour of a system over intervals of time in the present and future.
- Logs
  - An immutable, timestamped record of events that happened over time

Monitoring System

- Processing & storage
  - Accept and store incoming and historical data
  - The monitoring system should be able to manage data over periods of time which may involve sampling or aggregating old data

- Visualization
  - Monitoring systems typically provide visualizations of data, metrics can be display

- Alerting
  - Monitoring system is typically as a platform for defining and activating alerts.

Key Monitoring Principles for ML

- Monitor Model Predictions
  - Check for skew, bias and other quality indicators
- Computational performance
  - Monitor system training speed, serving latency..

Monitoring Metrics for ML

- Metrics represent the raw measurements of resource usage and behaviour that can be observed and collected throughout your systems.
- It can be low level usage summaries provided by the OS or they can be higher level types of data tied to specific functionality or works of a component such as requests per second or outputs from a particular function.

ML Metrics

- Operational – is it working?
  - Latencies
  - Memory size
  - CPU usage
- Are the predictions accurate?
  - Model output
- Is the data what is expected?
  - Model inputs

Summary

- We introduced different stages in Machine Learning lifecycle. Model deployment and model monitoring are two important stages, we need to deploy ML models in production environment and monitor its performance throughout the model lifetime.
- We discussed different challenges of machine learning deployment.
- We introduced FOUR types of machine learning deployment systems.
- We discussed the motivations of model monitoring.
- We also introduced what is data drift and FOUR different types of concept drifts as well as how to detect data drift.

References

- "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction" (2017) Breck *et al*. *IEEE International Conference on Big Data* (Google)
- "Software Engineering for Machine Learning: A Case Study" (2019) Amershi et al. (Microsoft)
- D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in Machine learning systems. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15). MIT Press, Cambridge, MA, USA, 2503–2511.
- https://towardsdatascience.com/4-machine-learning-system-architectures-e65e33481970
- Webb, G. I., Lee, L. K., Goethals, B., & Petitjean, F. (2018). Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, *32*(5), 1179-1199.
- [1] Quionero-Candela, Joaquin, et al. Dataset shift in machine learning. The MIT Press, 2009.