Lesson 10: Deployment

Shawn Chai

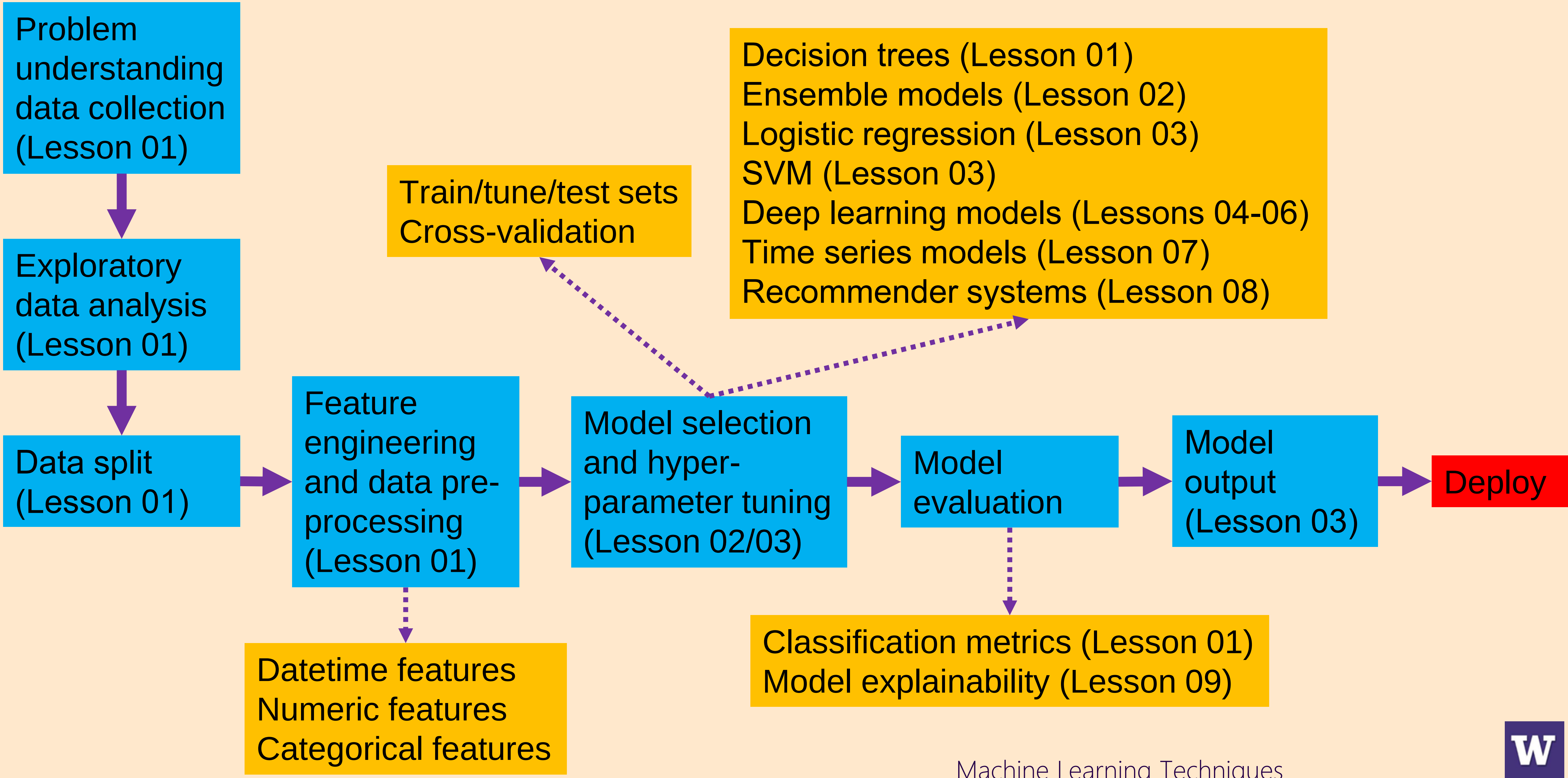
UNIVERSITY *of* WASHINGTON

Agenda

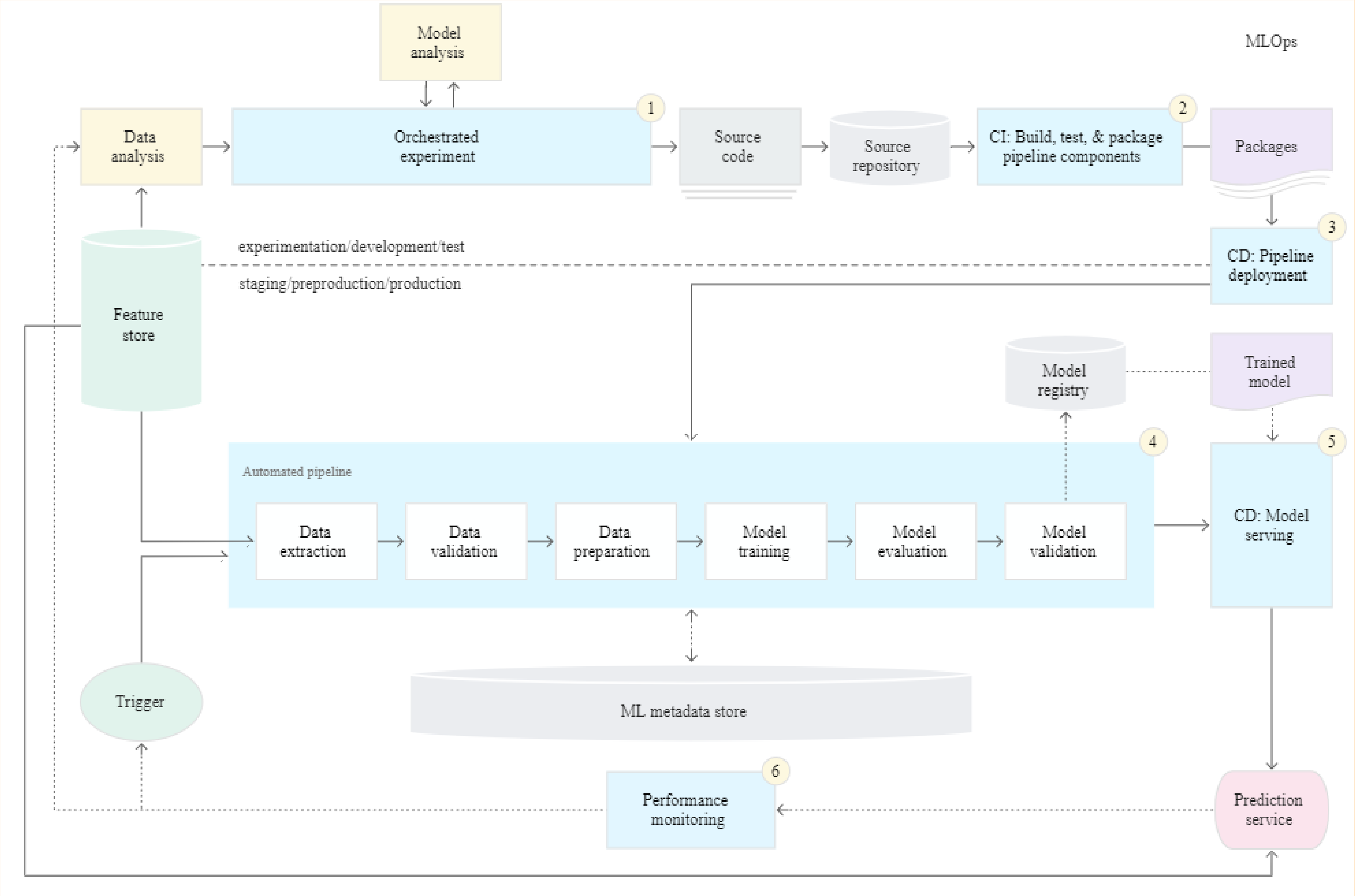
- Machine learning lifecycle.
- Course summary.

Machine Learning Lifecycle

Recap: ML steps covered so far



MLOps



Source:
https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

ML deployment

- After we have built an ML model and validated it, we would like to let users interact with it in a scalable way.

- But before deployment, there are a lot of considerations:
  - Are we allowed to continuously collect ML features from users and use your model to make predictions?
  - Can our model create potential ethical issues?
  - Do we have a safeguard in case your model goes wrong?
  - Can someone interject our model and misuse it?
  - Do we need to get approvals from legal team in the company?
  - Etc.

ML deployment options

- Cloud deployment
- Client deployment
- Hybrid deployment

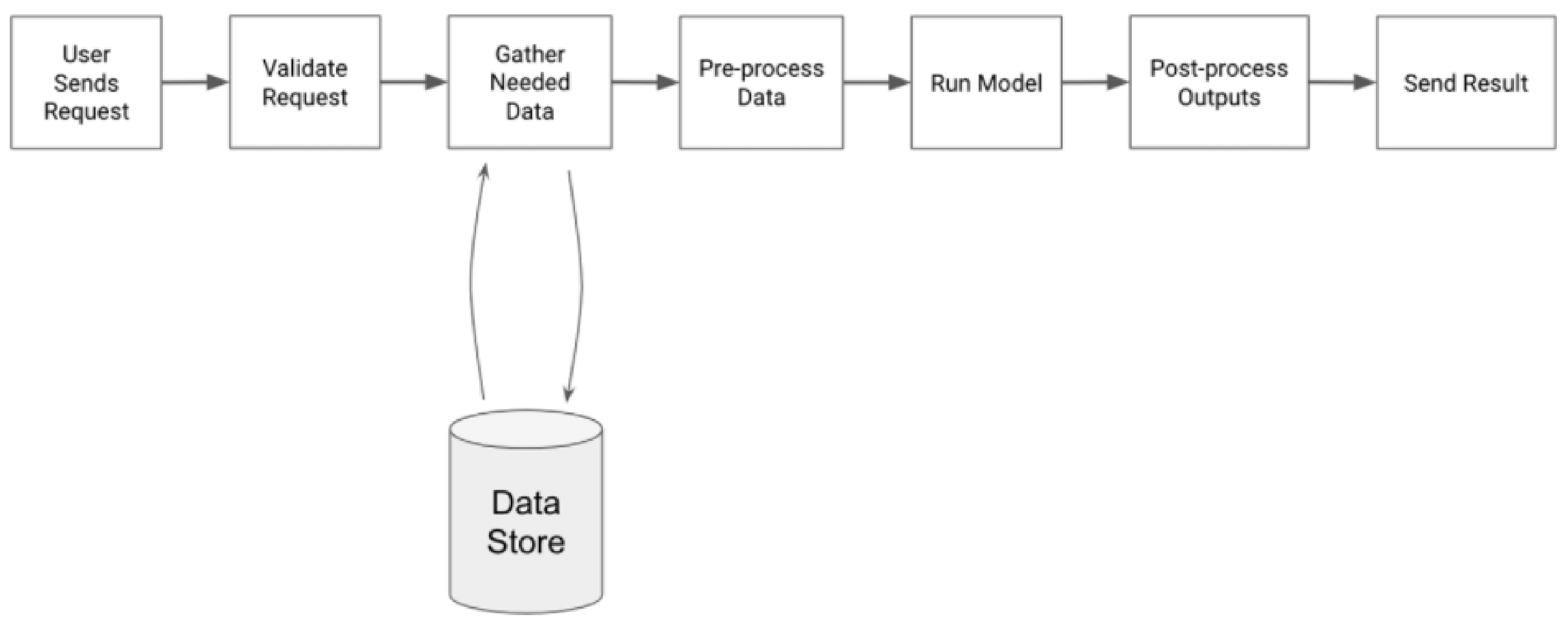
Cloud deployment

- Set up a web server that can accept requests from clients, run them through an inference pipeline, and return the prediction results.

- Two common workflows:
  - Streaming API: accept prediction requests as they come and process them immediately.
  - Batch: process a large number of prediction requests all at once.

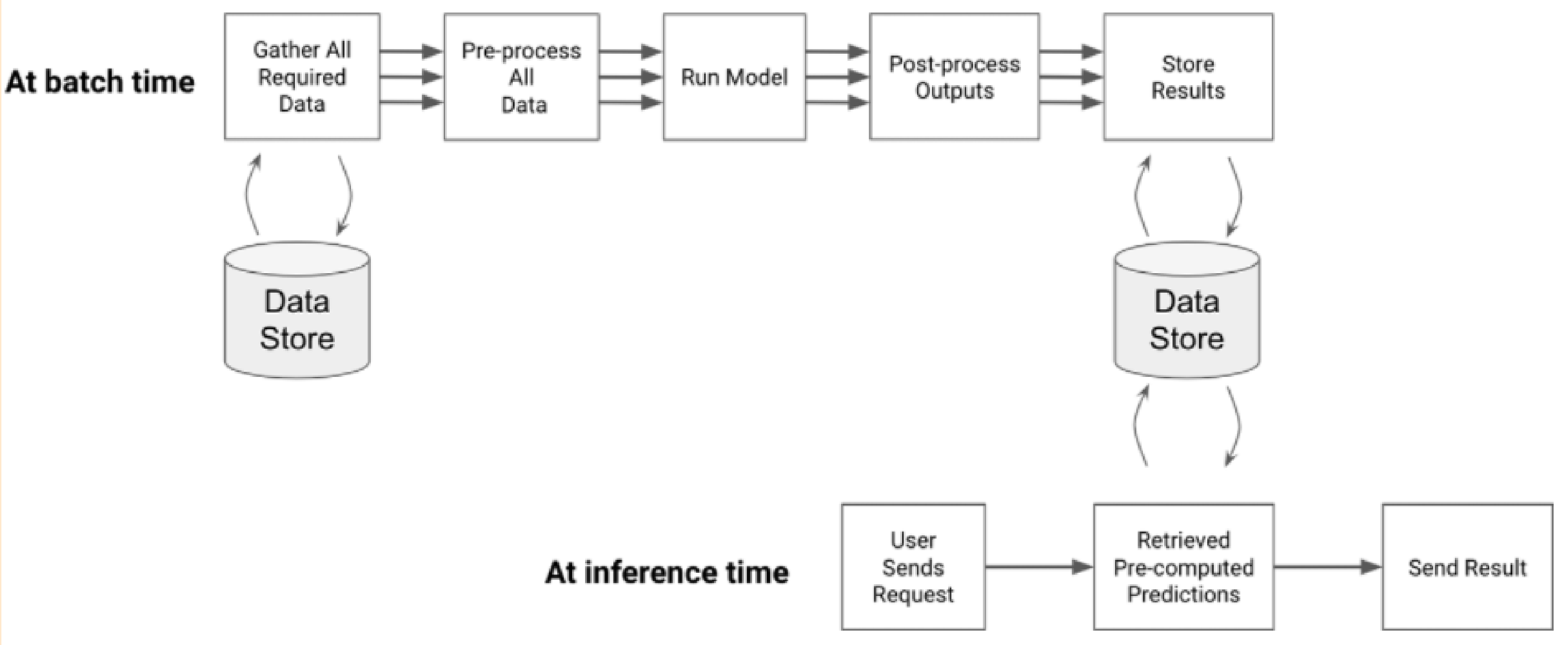
Cloud deployment – streaming API

- Streaming API workflow is good for scenarios that requires ML model's predictions to be available with low latency. Think about search engines.
- Common steps:



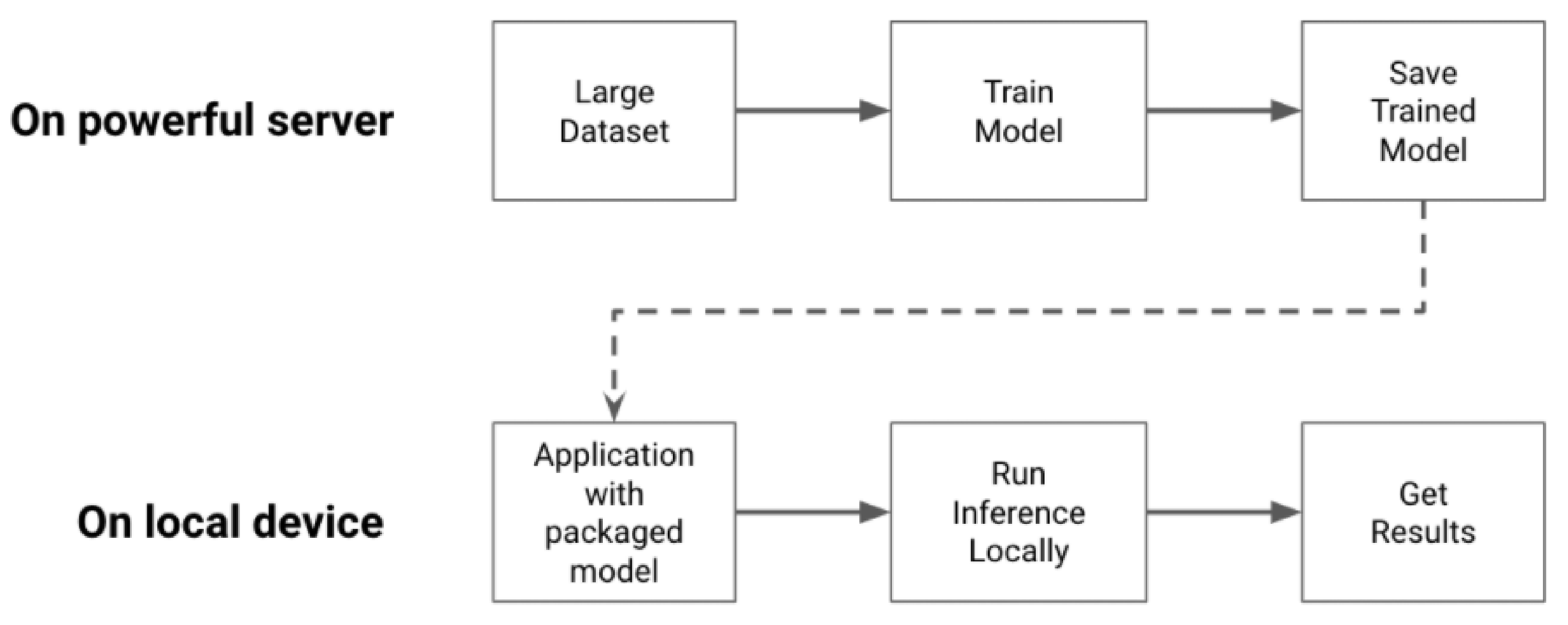
Cloud deployment – batch

- Batch workflow is good for scenarios when we have access to model features before the model prediction is required. Think about house price prediction model.
- Common steps:



Client deployment

- Run all model predictions on the client, e.g. computers, tablets, smartphones, IoT devices.



Client deployment

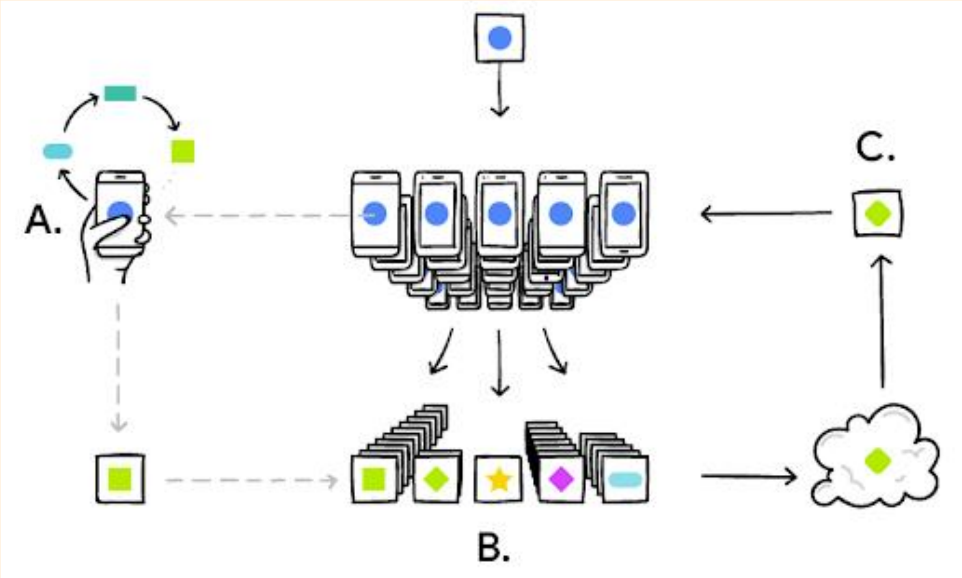
- Benefits:
  - No data transmission back to the cloud.
  - Very low latency time of making a prediction.
  - Work even without internet access.
  - Privacy protection of users' sensitive data.

- A #shamelessplug of me presenting this topic at the International Conference on Software Engineering (ICSE) ☺ https://youtu.be/k743aJAM5hk

Hybrid deployment – federated learning

- A seamless integration between cloud and client deployment:
    - Each client trains its own model locally based on their user's data, and send aggregated updates to the cloud.
    - The cloud server improves its global model based on individual updates, and push the new model back to each client.

- An exciting direction that powers Google's mobile keyboard predictions: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html



Lab

- ML lifecycle using mlflow.

Couse Summary

Recap: ML steps covered so far

Problem understanding data collection (Lesson 01)

Exploratory data analysis (Lesson 01)

Data split (Lesson 01)

Feature engineering and data pre-processing (Lesson 01)

- Datetime features
- Numeric features
- Categorical features

Model selection and hyper-parameter tuning (Lesson 02/03)

- Train/tune/test sets
- Cross-validation

- Decision trees (Lesson 01)
- Ensemble models (Lesson 02)
- Logistic regression (Lesson 03)
- SVM (Lesson 03)
- Deep learning models (Lessons 04-06)
- Time series models (Lesson 07)
- Recommender systems (Lesson 08)

Model evaluation

- Classification metrics (Lesson 01)
- Model explainability (Lesson 09)

Model output (Lesson 03)

Deploy Lesson 10

General advices for ML practice

- Do you build an ML model for people, or for machines?
- 80/20 rule: grasp deep knowledge to solve 80% of all problems.
- Human judgement is influential.
  - When you have a problem at hand, always start thinking how a human domain expert would solve it manually.
  - What is the human error rate (especially for tasks in image, text, etc.)?
- Simpler is better.
  - Do you really need a deep learning model?
  - Do you really need an ensemble model?

Tip #1: Does the problem even need an ML solution?

- Start with a non-ML solution, which works better in the real world than we thought. For example,
  - Random guessing baseline.
  - Simple heuristic: recommend items based on most recent/frequently used ones.
  - Zero rule baseline: predict majority case all the time.
  - Human baseline: what's the human error rate is.
  - Existing solutions based on domain knowledge.

- Stay tuned on the last tip about how to justify ML's "lift" comparing to simple heuristics.

Tip #2: Do you have **representative and good quality** training data?

- In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to.

- If your training data is full of errors, outliers, and noise, it will make it harder for the system to detect the underlying patterns.

- Data scientists spend a significant amount of time on cleaning and exploring data.

Tip #3: Avoid the state-of-the-art trap

- It’s essential to stay up to date on state-of-the-art technologies for your own career and your business.

- On the other hand, if possible, solve your real problems with simpler and cheaper solutions.
  - Most state-of-the-art technologies are proposed by evaluating on some static datasets.
  - It does not mean it will also perform well on your data.
  - It does not mean it will be fast/cheap enough for production implementation.

Tip #4: Don’t overlook the power of simpler models

- They are easier to deploy, which allows you to understand, validate and debug production pipeline early.
- They can serve as a baseline on top of which you can build more complex models.
- Industry examples:
  - Facebook (ads click prediction): https://research.fb.com/publications/practical-lessons-from-predicting-clicks-on-ads-at-facebook/
  - Google (recommender system at Google Play store): https://arxiv.org/abs/1606.07792
- Ensemble models are less favored in production.

Tip #5: Improve performance with more feature engineering

- Feature engineering, together with data pre-processing are more important than sophisticated ML models.

- In practice, start by trying “two extremes” as baselines:
  - “Performance floor”: fit a naïve linear model.
  - “Performance ceiling”: fit a complex model like ensemble learning.

- Perform more feature engineering to improve model performance.

Tip #6: Model interpretability is important

- In the field of statistics, model interpretability is never a problem because it's required ☺

- In the field of machine learning, model interpretability is also required but often overlooked ☹
  - Interpretability builds trust with stakeholders.
  - Interpretability speeds up model debugging.

Tip #7: Error analysis is often overlooked

- An overall measure of accuracy often hides the details.
- Error analysis: on what population the model makes more errors?
- Example: face detection comparison across APIs, conducted by Inioluwa Deborah Raji et al (as of August 2018).

	Overall	Darker Female	Darker Male	Lighter Female	Lighter Male
Microsoft	99.52%	98.48%	99.67%	99.66%	100%
Face++	98.40%	95.90%	98.70%	99%	99.50%
IBM	95.59%	83.03%	99.37%	97.63%	99.74%
Kairos	93.40%	77.50%	98.70%	93.60%	100%
Amazon	91.34%	68.63%	98.74%	92.88%	100%

Tip #8: Combine offline and online A/B testing evaluations

- Offline evaluations are helpful based on static test set.
  - Always have a baseline efficacy to compare with, e.g. comparing with simple heuristics without ML.
- Online evaluations via A/B testing are essential.
  - A/B testing is used in industry to establish “causal impact” of your ML model.
  - It gives model efficacy based on real users' data.
  - It checks reliability and latency of your model based on real users' situations.
- In either case, connect ML metrics with business goals.

Ethics of Artificial Intelligence (AI)

- Microsoft: https://www.microsoft.com/en-us/ai/responsible-ai
- Google: https://ai.google/responsibilities/
- IBM: https://www.ibm.com/watson/ai-ethics/
- PwC: https://www.pwc.com/gx/en/issues/data-and-analytics/artificial-intelligence/what-is-responsible-ai.html

Farewell, and stay connected ☺

- Thank you for all your patience and feedback!
- Please complete the end course survey.
- Farewell, and stay connected ☺ Shawn's LinkedIn: https://www.linkedin.com/in/xchai/

References

- Mitchell, T. (1997). *Machine Learning*. New York, NY: McGraw-Hill.
- Hastie, T., Tibshirani, R., & Friedman, J. H. (2013). *The elements of statistical learning: Data mining, inference, and prediction* (Second edition). New York, NY: Springer.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning: With applications in R*. New York, NY: Springer.
- Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling.* New York, NY: Springer.

References

- Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (Second edition). Sebastopol, CA: O'Reilly Media.
- Grus, J. (2019). *Data science from scratch: First principles with Python* (Second edition). Sebastopol, CA: O'Reilly Media.
- Vanderplas, J. (2016). *Python data science handbook: Essential tools for working with data* (First edition). Sebastopol, CA: O'Reilly Media.

Lesson 10: Deployment