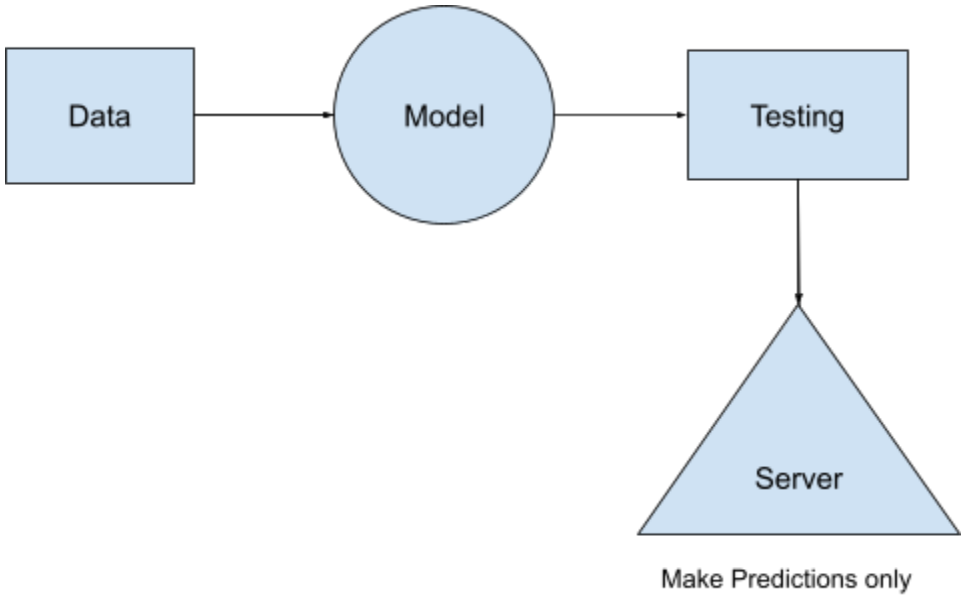
Batch and Online Machine Learning

1- Batch/Offline ML

→ This is the conventional way of training ML models.
→ In Batch ML, we train the model on the entire dataset at once.
→ There is no incremental learning, the model learns from all the data in one go.
→ The process looks like this: we take the full dataset, train the model offline, and once training is complete, we deploy the model to the server.



Problem With Batch ML

→ In Batch ML, the model is static.
→ It doesn't learn anything new after training, it only performs exactly as it was trained on the old data.
→ once the model is deployed on a server, it only makes predictions. To update the model, we need to take it offline, retrain it with new data, and redeploy it.
→ This makes Batch ML unsuitable for applications that need to adapt to dynamic or real-time changes.
→ For example, if we build a movie recommendation system using Batch ML, it won't be able to recommend the latest movies until we retrain the model with updated data.

Disadvantages of Batch ML

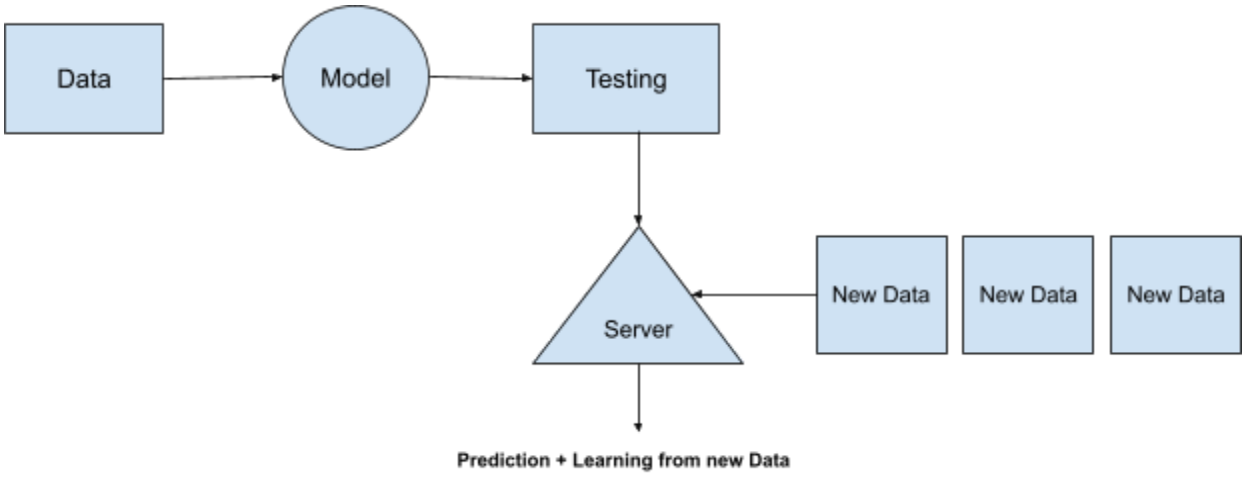
→ **Lots of Data:** If the dataset is very large and keeps growing rapidly, it may become impossible to train the model all at once, since available resources might not handle such massive data.

→ **Hardware Limitations:** Batch ML often requires high memory and heavy computational resources, especially for large datasets.
→ **Updating & Availability:** Updating the model with new data takes significant time and effort, and Batch ML lacks flexibility to adapt to real-time or fast-changing data.

Imagine we have a social networking website that generates a personalized news feed. If the ML model behind it is trained with Batch ML and only updates every 24 hours, it will fail to capture sudden trends. For instance, if a viral news story breaks, the model won't show it in the feed until the next training cycle. By the time the model updates after 24 hours, people may have already lost interest, making the recommendations irrelevant.

2- Online Machine Learning

→ Unlike the Match ML, it is done by feeding data incrementally, processing one data point at a time (or in small batches)



→ In online learning, the model updates itself dynamically on the server as new data comes in, makes predictions and learns from the new data at the same time.

Where to use it?

→ data keeps coming continuously
→ you want the model to learn on-the-go
→ data patterns change over time (drift)
→ dataset is too large to fit in memory
→ you need real-time predictions and updates
→ you want frequent model updates without retraining

Learning Rate in Online Learning

→ The learning rate decides how often we update the model as new data arrives.
→ Typically, we don't update the model on every single data point, because doing so may cause the model to forget what it learned from older data.
→ Choosing the right learning rate is both tricky and important, since it needs to be tuned carefully based on the problem.

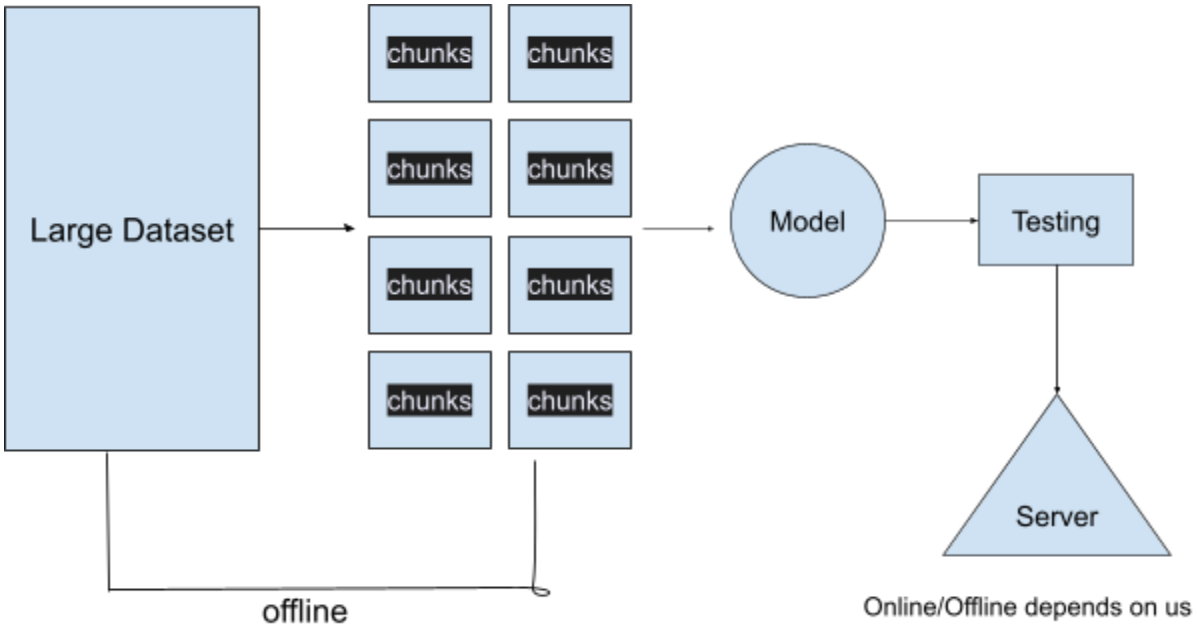
Disadvantages of Online Learning:

→ **Tricky to Use:** Online learning is not straightforward, handling continuous incoming data properly is crucial, and even small mistakes can lead to poor model performance.

→ **Risky:** The model's behavior heavily depends on the quality of the incoming data. If the new data is noisy, biased, or incorrect, the model may quickly learn the wrong patterns and become unreliable.

Out of Core Learning

→ It is a technique used to train a model on datasets that are too large to fit into a computer's main memory (RAM).
→ Instead of loading the entire dataset into memory, out-of-core learning processes data in smaller, manageable chunks or mini-batches.
→ The training itself is done offline (obviously we will not upload the entire big data on the server), but the technique behind it falls under Online Learning.



Batch/Offline vs Online Learning

Offline Learning	Features	Online Learning
Less complex as model is constant	**Complexity**	Dynamic complexity as the model keeps evolving over time
Fewer computations, single time batch-based training	**Computational Power**	Continuous data ingestions result in consequent model refinement computations
Easier to implement	**Use in Production**	Difficult to implement and manage
Image Classification or anything related to Machine Learning - where data patterns remains constant without sudden concept drifts	**Applications**	Used in finance, economics, heath where new data patterns are constantly emerging
Industry proven tools. E.g. Sci-kit, TensorFlow, Pytorch, Keras, Spark Mlib	**Tools**	Active research/New project tools: E.g. MOA, SAMOA, scikit-multiflow, streamDM

@iamahsanalirajpoot