MACHINE LEARNING DEPLOYMENT MODEL WITH IBM WATSON STUDIO CODE

Phase:4 Development Part-2

The Development and Training of Machine Learning Models

In ML, no model means no deployment, and the development of every model starts with the need to solve a business need/question. Knowing this business question helps align and choose an algorithm that fits the business use case. Then follows data collection, transformation and pre-processing steps, like cleaning and data wrangling. At this stage, an enterprise data integration tool like StreamSets can step in to ensure data from across your organization is pulled together, cleaned, transformed and ready for modeling.

After achieving excellent data quality, model selection and training begins. The model training step is the most crucial and determines the performance and accuracy of your final ML model. The model training step involves selecting an algorithm that fits your business use case and training the algorithm on your training data set until it achieves a particular level where further training decreases accuracy or increases validation error. The model training process is iterative and experimental and involves a training loop of experimental design, model structure formulation, parameter estimation, and model validation until models achieve a minimum validation error. The iterative model training process also involves optimizing and testing model algorithms to ensure models perform optimally in production after deployment.

How Machine Learning Models Are Deployed

Once an ML model is ready, the next step is to make it available for users to make predictions. Placing an ML model in an environment for users to interact with and use for decision making refers to model deployment.

For example, a loan application might use a model to predict a credit score for users using a set of values. Predicting the credit score can be as easy as calling this function: Prediction = classifier.predict(INPUT DATASET).

The Process of Deploying Machine Learning Models

ML model deployment involves the following steps:

1. **Develop, create, and test the model in a training environment:** This step requires rigorous training, testing, and optimization of the model to ensure high performance in production. The model training step determines how models perform in production. ML teams must collaborate to optimize, clean, test, and retest model code.
2. **Movement of models to deployment environment:** After rigorous testing and optimizing model code, the top-performing models undergo preparation for deployment. Models need a deployment environment that contains all the hardware resources and data required to make the model perform optimally. Different deployment environments include:
    - **Containers:** Most teams use a container deploying environment because containers are reproducible, predictable, and easy to modify and update, making collaboration among engineers easy. Containers encompass all the hardware, configurations and dependencies necessary to deploy the model, improving consistency among ML teams.
    - **Notebooks:** Jupyter and AWS Sagemaker are common notebooks used by data scientists for experimentation in the ML lifecycle. However, notebooks present difficulties like reproducibility and testing for teams. To efficiently use notebooks in the production workflow, teams should consider code organization, reusability, and dependencies, among other factors.
    - **In-App environments:** This environment works when certain limitations or constraints exist around using data outside the application.
3. **Making models available for end users:** ML teams must choose how to make models available for their users. Most can be available on demand or deployed to edge devices.
4. **Monitoring:** The ML lifecycle continues after deployment. Deployed models must undergo constant monitoring to evaluate the performance and accuracy of models over time. Because data is in a continual state of

motion and change, model degradation may occur. In this case, automating the ML workflow to monitor and retrain models constantly helps ensure the longevity of models.

-

Four Ways You Can Deploy ML Models Into Production

Deploying ML models into production can follow various methods, each with its merits. Let's explore some popular ML deployment methods:

- **On-demand prediction mode:** This deployment mode means users offer input on a model and receive predictions immediately, in real-time. Although this deployment method offers prediction on demand, it confers on prediction results an inherent latency, which limits the type of models it can utilize. This inherent latency means real-time prediction deployment can't deploy complex models.
- **Batch prediction:** Also referred to as offline model deployment, this deployment method runs periodically and returns results only for the new data generated since the previous run. Most batch predictions use ETL processes to fetch pre-calculated features from feature stores to use an input for the prediction model. It works for cases where real-time predictions aren't a priority. The batch method offers the advantage of its ability to perform more complex predictions and handle a high volume of instances. It also eliminates the constant worry of scaling or managing servers to handle peak demands, as seen in real-time prediction.
- **Deployment using a web service:** This method is the simplest and deploys the model as a web service, by building a REST API and using the API in mobile or web applications for users. Deployment as web services mainly serves ML teams with multiple interfaces like web, mobile, and desktop. Standard technologies that power web-service prediction models include AWS Lambda and Google cloud functions, docker containers, or notebooks like Databricks.
- **Deploying on edge devices as embedded models:** Edge computing has recently become popular for its improved latency and reduced user bandwidth. This improved performance results from placing compute resources and

devices close to the user. However, because of the limited size of hardware, compute power, and storage capacity of most edge-computing devices, deploying ML models directly on devices isn't practical. However, aggregation and quantization processes in tools like the TensorFlow Lite library help simplify the models for successful deployment on edge and IoT devices.

Considerations for Deploying Machine Learning Models

ML teams must carefully consider these factors before deciding on a deployment model that works:

- **Frequency of predictions:** knowing how often your model will generate predictions helps allocate resources like computing and storage to serve the model.
- **Time-consciousness of prediction results:** This helps decide between a real-time or batch method. For example, some services, like logistics or estimated delivery time for food, will benefit from a real-time prediction model, which improves the customer experience. Other services, like equipment maintenance, can use a batch prediction method whereby ML models batch maintenance metrics to inform on faults or those needing servicing.
- **Computational power and load implications:** Real-time predictions require setting up a plan to handle peak load while ensuring optimum performance while in use and achieving the set SLA. Preparing for peak loads may involve acquiring additional compute resources, which means more cost. On the other hand, handling load is more flexible for batch predictions as models compute predictions over a time batch.
- **Hardware and infrastructural responsibility:** Real-time prediction deployments create additional responsibility for ML teams as models need constant monitoring for performance checks to detect any drop in performance or issues. On the other hand, although batch predictions still need continuous monitoring, there is less responsibility, as there is less urgency in fixing any problems.
- **The complexity of the model:** Regression algorithms like linear and logistic regression are not complex to execute and do not require significant computing power, so using them for real-time predictions work. For more complex models like neural networks that require considerable time and computing power

to churn predictions, deploying real-time models would mean more cost and increased latency.

ML Model Deployment and the Cloud

Machine learning uses a significant amount of computing power, storage, and multiple servers for training and churning out predictions, which can be expensive to set up. However, the flexibility of the cloud makes ML adoption easier because organizations can quickly spin up compute and server resources to train, experiment, and deploy models—and shut these resources down when they're not in use.