**AWS Sagemaker- Detracting the ML heavy lifting**

Nowadays, the focus on machine learning in cloud has risen due to several advantages like pay-per-use model, high computation power and availability. The best part about ML in cloud is that we can begin testing with smaller projects and then scale up in case of high demand, this prevents starting off in full-fledged mode and keeps the window to experiment open. Here is where Amazon Sagemaker comes into picture, it is a fully-managed service that enables developers and data scientists to quickly and easily build, train, and deploy machine learning models at any scale.

**What is Sagemaker?**

AWS Sagemaker is service belonging to the ML stack, it is fully managed service which allows development and training of ML models in production-ready environment. The need of server management is diminished as it has an integrated Jupyter notebook which is used to explore and analyse datasets. Popular ML algorithms are optimized to handle heavy workloads, also bring-your-own-algorithms are well supported by offering customisable distributions options. Let’s explore below that how AWS Sagemaker makes it work end-to-end.

* **Exploration and pre-processing of data**

It involves creation of a Jupyter notebook on Sagemaker instance which should be suitable to begin with creation of model training jobs, transformation of data and further testing and validation. This notebook will also be used to deploy through Sagemaker hosting.

* **Training ML models**

After cleaning and pre-processing of data, the next step is training the model for which a training job has to be created. This training job is a script which contains URL or S3 location of the folder where training data resides, the compute specifications of instances that are required and also the registry path of the training code

* **Deploying the ML model**

Coming to the deployment part, there are two options for this. The first one is creating a persistent endpoint which obtains single prediction at a time from Sagemaker hosting service. The other option is of batch transformation, this option is suitable for getting predictions for full dataset.

* **Monitoring the model deployed in production**

After deploying the model in the production environment, Sagemaker continuously monitors the quality of the deployed ML model in real time. There is model monitor in Amazon Cloudwatch which lets setting up of automated alert triggering system in case of any deviations beyond threshold like anomalies and data drifts. The log files are maintained and stored in S3 and setting up relevant triggers leads to quick resolutions.

**How local training & deployment waste time in ML related tasks**

* Setting up or maintaining notebook servers
* Bargaining with co-workers for compute resources
* Troubleshooting conflicting package installations
* Not knowing how your model is performing while training
* Stalling in front of your customers while your inferencing starts
* Not being able to replicate model training jobs
* Grid, random or manual hyperparameter searching or none at all
* Not having a continuous integration pipeline.

**Amazon SageMaker Features**

* **Amazon SageMaker Studio:** This feature provides an integrated ML environment for building, training, deploying and analysing of model in a single application.
* **Amazon SageMaker Ground Truth:** This feature is used to create labelled dataset with the help of workers, this dataset is then used for high quality training.
* **Amazon Augmented AI:** It includes Human-in-the-loop reviews which allow for manual intervention in cases where confidence score of the service falls below set threshold level.
* **Amazon SageMaker Studio Notebooks:** The latest version of Amazon SageMaker notebooks includes SSO (Single Sign-On) integration, faster start-up time and single-click sharing.
* **Amazon SageMaker Experiments:** It is used to experiment with management and tracking. The tracked data is used to reconstruct some experiment or incrementally build on experiments which are conducted by peers.
* **Amazon SageMaker Debugger:** It inspects training parameters and data all along the training and automatically detects and alerts users for common errors like parameter values going out of range.
* **Amazon SageMaker Autopilot:** This feature is for users who don’t have in-depth ML knowledge. It helps them to quickly build classification and regression models.
* **Batch Transform:** This feature is used for pre-processing the datasets and running the inference in case persistent endpoint is not needed.
* **Amazon SageMaker Model Monitor:** This is used to monitor and analyse production models or endpoints for detecting data drifts, deviations in quality of model.
* **Amazon SageMaker Neo:** This is used to train ML models once and then run them anywhere in the cloud and edge.
* **Amazon SageMaker Elastic Inference:** It speeds up the throughput and decreases the latency for real-time inferences.

**Sagemaker USP - a game changing solution for the enterprise**

* In-built Jupyter notebooks running R/Python kernels with a compute instance as per on demand data engineering requirements.
* Visualization, processing, cleaning and transforming of the data into required forms using Pandas, Matplotlib, R, ggplot2 etc.
* Post data engineering, the models are trained using a different compute instance on the basis of compute demand.
* Leverages smart default high-performance hyperparameter tuning setting.
* Leverage performance-optimized algorithms from AWS library.
* Supports bring our own algorithms through industry standard containers.
* Deploys the trained model as an API with a few lines of code
* Cost effective

**Best practices**

Whenever hosting ML models using hosting services of AWS SageMaker, the following should always be considered:

* Usually, the client application will send requests to AWS SageMaker HTTPS endpoint for obtaining any inferences from the deployed model. Send these requests to the endpoint from Jupyter notebook during instance during testing.
* Try deploying multiple variants of the model to the same AWS SageMaker HTTPS endpoint so that it is useful for testing variations of that model in production.
* For modifying any endpoint without hampering their production workloads, one can add new model variants by updating ML compute instance configurations for the new variants.
* For any changes or deletion of model artifacts after model deployment one has to modify the endpoint by creating a new endpoint configuration and then update the model artifacts which correspond to the old endpoint configuration.
* To get inferences on entire datasets, one should consider using the batch transform option as an alternative to hosting services.
* Use inference pipeline to define and deploy any combination of pretrained Amazon SageMaker built-in algorithms and custom algorithms packaged in Docker containers.
* Whenever using spot instances, one should keep in mind that these instances access spare capacity of Amazon EC2 and can be reclaimed within a few minutes of allocation. So, its important that we use automatic checkpointing for back-ups, also sync the same to S3.
* Keep a note of checkpoint frequency period for storing model parameters (in count of epochs).

Use cases of Sagemaker have a wide range of scope which is inclusive of supervised learning, unsupervised learning, textual analysis, image processing and analysis, performance analysis, forecasting, fraud and anomaly detection, predictive analysis. Some market alternatives for Sagemaker are IBM Watson Studio, Google Cloud AI Platform, Azure Machine Learning Studio, TensorFlow, Google Cloud AutoML, DataRobot, Dataiku, H2O. Among these, Sagemaker is highly flexible and cost-effective and it offers easy integration with MapReduce and other AWS services which makes it easier to deploy ML models as compared to open-source tools.