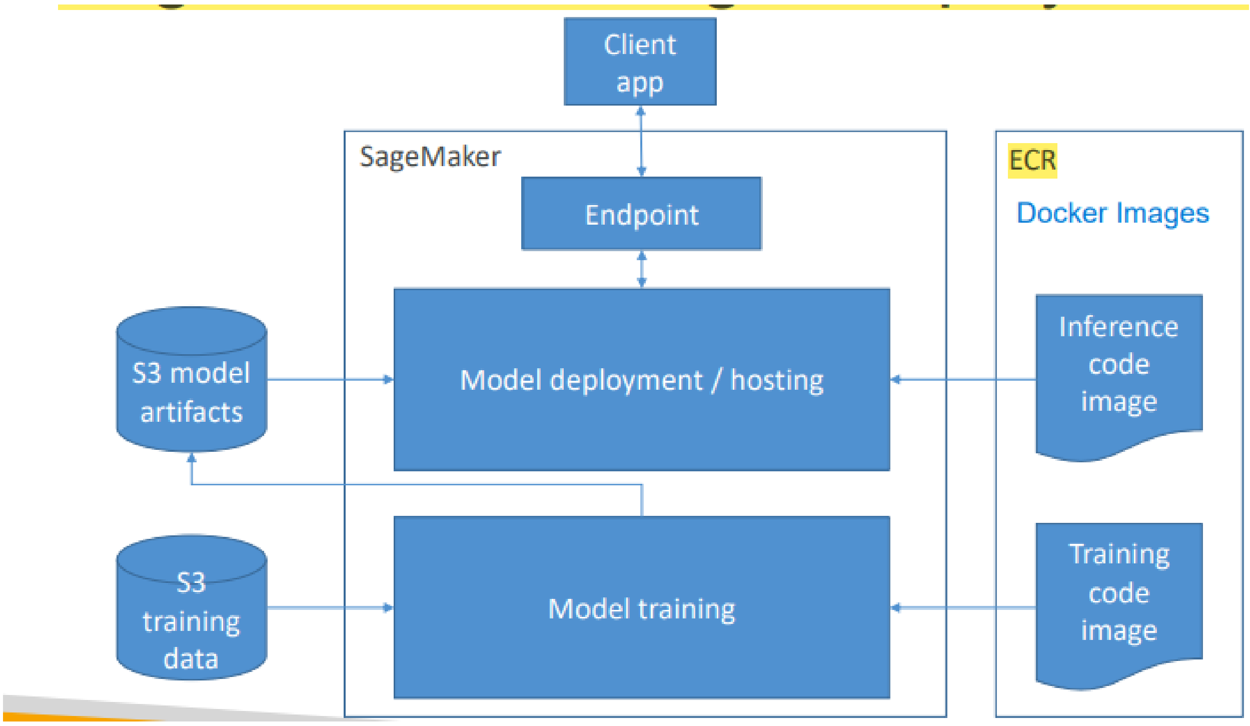
Section 5. Modeling -- Amazon SageMaker

`SageMaker` is built to handle the entire machine learning workflow

1️⃣ SageMaker Deployment

- `SageMaker Notebook` & `SageMaker Console` is built to run and monitor the workflow
- The data in the data processing usually comes from `S3`, so does the output processed data
- The training process
  - Load data from `S3`
  - Load training code from the `Docker Image` that contains the training code and environment



(Image Retrieved from [1])

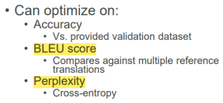
- Trained models can be deployed in two ways:
  - Persistent `endpoint` for making individual predictions on demand
  - `SageMaker Batch Transform` to get predictions for an entire dataset
- Lots of cool options:

`Inference Pipelines`	`SageMaker Neo`	`Elastic Inference`	`Automatic Scaling`	`Shadow Testing`
For more complex processing	For deploying to edge devices	For accelerating deep learning models	Increase the number of endpoints as needed	Evaluate new models against currently deployed model to catch errors

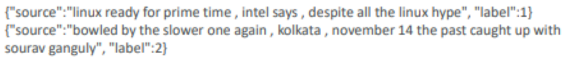
2️⃣ SageMaker's Built-In Algorithms

- `File Mode`: Copy all the training data over as a **single file** at once to all your training instances
- `Pipe Mode`: Pipe and stream data from `S3` as needed, which is more efficient, especially with larger training sets
- `Multi-GPU`: Multiple GPUs on one machine
- `Multi-Machine GPU`: GPUs on multiple machines
- `Serialization`: Converting the `state of an object` into a `byte stream` (`string`)
- `Deserialization`: Converting `byte stream` to the actual `object`

	Linear Learner	**XGBoost**	**Seq2Seq**
`Description and Used for`	• `Linear Regression`: Fit a line to your training data • Can handle both `regression` predictions and `classification` predictions (if a linear threshold function is used)	• eXtreme Gradient Boosting • New trees made to correct the errors of previous trees • Can be used for `classification` and `regression` (using regression trees)	• Both input and output are a sequence of tokens • Implemented with `RNN` and `CNN` with `attention` • Machine Translation, Text Summarization, Speech to Text
`Expected Input`	• `RecordIO-wrapped protobuf` (most performant option and `float 32` data only) • `CSV` (First column assumed to be the lable) • `File` or `Pipe` mode both supported	• `CSV`, `libsvm`, `RecordIO-Protobuf`, and `Parquet`	• `RecordIO-Protobuf` -- tokenized text files • Must provide `data` and `vocabulary files`
`How Is It Used`	• Training data must be **normalized** manually or automatically by `Linear Learner`, so that all features are weighted the same • Training data should also be shuffled	• Models are serialized / deserialized with `Pickle`	• Training for machine translation can take days, even on `SageMaker` • Therefore, pre-trained models and public training dataset are more feasible

	Linear Learner	**XGBoost**	**Seq2Seq**
`Hyperparameters`	- `Balance_multiclass_weights` - `Learning_rate` & `Batch_size` - `L1` & `L2` regularization	- `Subsample`: Prevent overfitting - `Eta`: Step size and prevent overfitting - `Gamma`: Minimum loss reduction to create a partition; larger = more conservative - `Alpha`: L1 regularization term; larger = more conservative - `Lambda`: L2 regularization term; larger = more conservative - `eval_metric` - `scale_pos_weight`: Adjust balance of positive and negative weights and helpful for unbalanced classes - `max_depth`: Max depth of the tree	- `Batch_size` - `Optimizer_type` - `Learning_rate` - `Num_layers_encoder` - `Num_layers_decoder`  • Can optimize on: • Accuracy • Vs. provided validation dataset • BLEU score • Compares against multiple reference translations • Perplexity • Cross-entropy
`Instance Types`	- Training on single or multi-machine CPU or GPU	- CPU for multiple instances - Single-instance GPU training	- Can only use GPU instance types and single machine for training

	DeepAR	**BlazingText**	**Object2Vec**
`Description and Used for`	- Implement `RNN` - Forecast one-dimensional time series data	- Intended for use with **sentences**, **not entire documents** - For `text classification` and `Word2vec` (`Word Embedding`)	

	DeepAR	BlazingText	Object2Vec
`Expected Input`	• `JSON` lines format	• `Text Classification`: one sentence per line and first word in the sentence is the string `__label__` followed by the label **OR** `augmented manifest text format`:  {"source":"linux ready for prime time , intel says , despite all the linux hype", "label":1} {"source":"bowled by the slower one again , kolkata , november 14 the past caught up with sourav ganguly", "label":2} • `Word2vec`: text file with one training sentence per line	
`How Is It Used`	• Always include entire time series for training, testing, and inference • Train on many time series and not just one when possible		
`Hyperparameters`	• `Context_length` • `Epochs` • `Batch_size` • `Learning_rate` • `Num_cells`		
`Instance Types`	• **Training**: CPU or GPU, single or multi machine • **Inference**: CPU only		

	Object Detection	Image Classification	Semantic Segmentation
`Description and Used for`			
`Expected Input`			
`How Is It Used`			
`Hyperparameters`			
`Instance Types`			

	Random Cut Forest	Neural Topic Model	LDA
`Description and Used for`			
`Expected Input`			

	Random Cut Forest	Neural Topic Model	LDA
`How Is It Used`			
`Hyperparameters`			
`Instance Types`			

	KNN	K-Means	PCA
`Description and Used for`			
`Expected Input`			
`How Is It Used`			
`Hyperparameters`			
`Instance Types`			

	Factorization Machines	IP Insights
`Description and Used for`		
`Expected Input`		
`How Is It Used`		
`Hyperparameters`		
`Instance Types`		

(Images Retrieved from [1])

🐠 Reinforcement Learning

3️⃣ Automatic Model Tuning

🐟 Best Practices

4️⃣ SageMaker and Spark

5️⃣ Modern SageMaker

	Linear Learner	XGBoost	Seq2Seq
`Description and Used for`			
`Expected Input`			
`How Is It Used`			

	Linear Learner	XGBoost	Seq2Seq
`Hyperparameters`			
`Instance Types`			

📚 References

[1] "AWS Certified Machine Learning - Course Materials," Sundog Education with Frank Kane. https://www.sundog-education.com/aws-certified-machine-learning-course-materials/ (accessed Jul. 23, 2023).