

## AWS ML - Modeling

Training on SageMaker

Deploying Trained Models

Linear learner :

Ensemble Methods

Bagging

Boosting

XGBoost

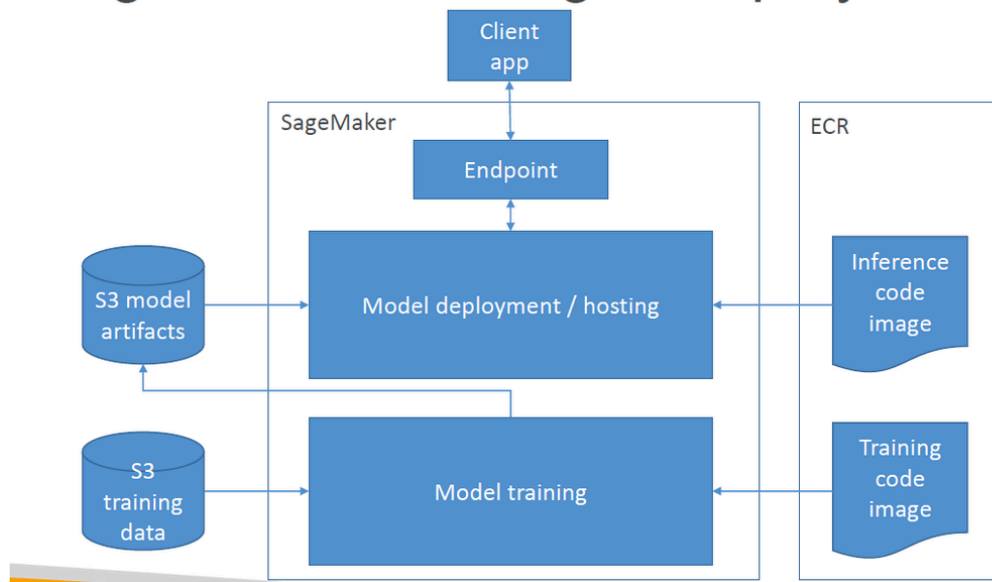
Semantic Segmentation

Seq2Seq

BlazingText

Time Series

## SageMaker Training & Deployment



### Training on SageMaker [↗](#)

Create a training job

- URL of S3 bucket with training data
- ML compute resources
- URL of S3 bucket for output
- ECR path to training code

### Deploying Trained Models [↗](#)

Save your trained model to S3

Can deploy two ways:

- Persistent endpoint for making individual predictions on demand
- SageMaker Batch Transform to get predictions for an entire dataset

## Linear learner : [↗](#)

- regression & classification
- input : csv(first column is the label or RecordIO wrapped protobuf(32 float)
- file mode : takes the entire dataset / Pipe mode: takes batches of dataset (efficient for large training dataset)
- Training data must be **normalized** (so all features are weighted the same). Linear Learner can do this for you automatically
- Uses stochastic gradient descent

Logistic Regression : Binary classification

Decision tree : classification or regression

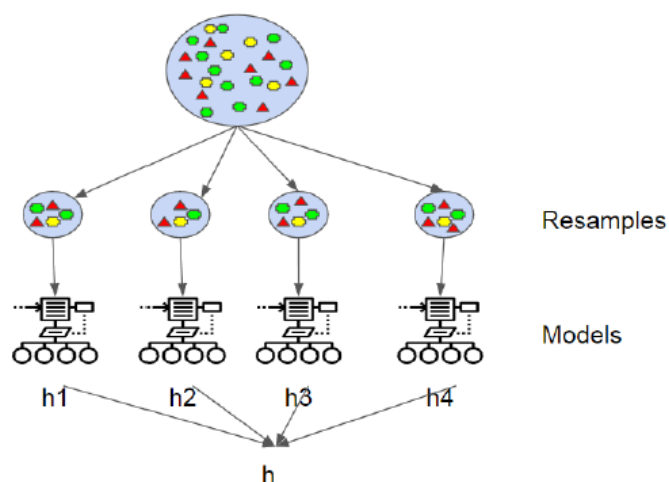
Naive Bayes: classification

multivariate Regression : predicting a dependent variable based on independant variables

## Ensemble Methods [↗](#)

An ensemble methos takes multiple models and they might be a variation of the same model( different set of training data, ...) and lets them all vote on a final result.

### Bagging [↗](#)

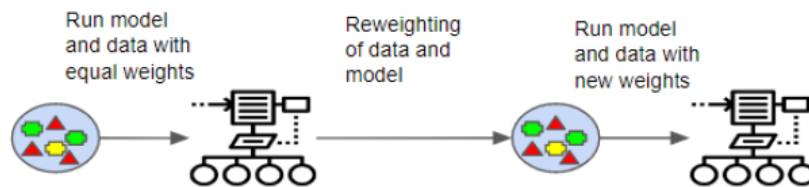


Bagging would generate multiple training sets by **random samling with replacement**. Each resampled model can be trained in parallel.

Random Forest

### Boosting [↗](#)

Training is sequential; each classifier takes into account the previous one's success. Observations are weighted



Boosting generally yields better accuracy

Bagging avoids overfittings

## XGBoost [↗](#)

- eXtreme Gradient Boosting
- Boosted group of decision trees
- New trees made to correct the errors of previous trees
- Uses gradient descent to minimize loss
- M5 is a good choice as instance type
- **Hyperparameters**
  - **Subsample**
    - Prevents overfitting
  - **Eta**
    - Step size shrinkage, prevents 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**
    - Optimize on AUC, error, rmse...
    - For example, if you care about false positives more than accuracy, you might use AUC here
  - **scale\_pos\_weight**
    - Adjusts balance of positive and negative weights
    - Helpful for unbalanced classes
    - Might set to  $\text{sum}(\text{negative cases}) / \text{sum}(\text{positive cases})$
  - **max\_depth**
    - Max depth of the tree
    - Too high and you may overfit

## Semantic Segmentation [↗](#)

Pixel-level object classification

Different from image classification – that assigns labels to whole images

Different from object detection – that assigns labels to bounding boxes

### What training input does it expect?

JPG Images and PNG annotations For both training and validation

JPG images accepted for inference

GPU instances for training (ml.p2, p3, g4dn, g5) Multi-GPU and multi-machine OK.

CPU or GPU for inference (m5, p2, p3, g4dn, g5)

## Seq2Seq [↗](#)

- Translation, summarization..
- , speech to text
- implemented with RNN or CNN with attention
- **What training input does it expect?** : RecordIO-Protobuf (integers)
- must provide vocabulary files

## BlazingText [↗](#)

- Text Classification : use with sentences not entire documents
- **What training input does it expect?** One sentence per line
- Word2vec
- **What training input does it expect?** a text file with one training sentence per line

## Time Series [↗](#)

- **Additive Model:** Seasonality with constant seasonal variation + Trends+Noise
- **Multiplicative model:** seasonal variation increases as the trend increases